Tampereen teknillinen yliopisto. Tietotekniikan laitos.
Tampere University of Technology. Department of Pervasive Computing.

Matti Vuori

Changing testing and quality assurance competence needs in Finnish software product development

Tampere University of Technology. Department of Pervasive Computing
Tampere 2017

Abstract

Testing and quality assurance are important activities in the development and deployment of all technological systems. While the world is indeed becoming more global, unique competences and capabilities are as important as ever – or even more so, because of the fierce competitions. Due to various factors, Finland cannot be similar to others. One part of the competitiveness is our ability to create for the global market good, marketable products and systems that have the qualities required and desired. By looking at things from Finland's perspective, we will gain a unique insight into ways of doing things in the development and the related competences.

The desired competences support the success factors of our unique Finland, such as the competences and fearless relation towards innovation and built upon the unique good qualities of Finland, such as good education.

This dissertation aims to see the changes in our environment and to find the testing related competences that should be actively supported during the coming years.

The research was qualitative and theoretical by nature. Research was based in modelling the product development environment on many levels at global phenomena, national changes, changes in companies, changes in software development and in testing. The changes analysed were concrete and selected subjectively, based on how they represent important more abstract phenomena, while trying to avoid possibly short-lived common media hype. That required some informal interpolation by the researcher as the information in media reflects the visible manifestations of phenomena, which are not sufficient to see as real "changes". For example, instead of the rise on "hackathons", it makes more sense to analyse the exploration culture.

The changes were condensed into change-competence snippets that describe the change and its associated competence needs and the links between different changes, thus producing a network or changes and competences. There were 62 such snippets. For differentiating types competences, a level system was used that consists of orientation, understanding and the ability to actually do the quality-related actions. For a basis of the analysis there was first a thorough analysis of what testing is currently. Another type of analysis was done using the activity system triangle model from action research, where the essential competences were identified against the actors themselves, system under test, development goals, organisation, teamwork, processes and tools and methods.

That analysis emphasises how the viewpoint into testing was organisational and cultural: testing is seen as an activity in an organisation that responds to its need, is done within the organisational structures and culture. Responding to pure engineering problems with engineering solutions is not viable, even though some testing has a technological basis.

A small survey was made to Finnish testing community, which produced understanding of the essential competences and their relations.

All approaches produced, as was expected, somewhat different views into the competences. The analysis of changes in the contexts of various levels produces more task-oriented views to competence, whereas the activity system-based analysis gives deeper insight on the local activity system as whole. Both complement each other.

The synthesis of the findings was made by grouping the most essential competences into phases of product development, the elements of the activity system triangle, and "competence lumps". Competence lumps are concepts than combine related competences at various levels (orientation, understanding, ability) into independent groups that can be used as guidance for role definitions, training and education.

There is a need to revise the old stereotypes related to testing. It is usually perceived as functional testing during the implementation of software, but we need to consider all experimental assessments as a holistic toolbox, starting with concept level experiments, validations in lean startup -type of development and user experience testing. Testing evolves in practice, but to be effective we need to have mindsets that capture the rich whole.

Security and user experience are essential success factors for any new product development and assessment and testing of those adds to the core of testing competences. The new society in general requires everyone involved in product development to have some business orientation and understanding of it, to be able to support better the goals of their organisation. Obviously, the complex systems require good technical skills and test automation skills. The needs for competences are so wide and diverse, that any one person definitely cannot possess all of them. This is where the competence lumps help as they provide a view to full sets of competences that a person can realistically have.

But the situations change all the time and the changes change! For this reason, this dissertation should, in a longer term, be seen more as a methodological work that presents approaches for identifying competences on most any changing domain.

Keywords Quality, software testing, quality assurance, software testing, Finland, competence, competence models. skills, knowledge, product development, software development, changes, future

Tiivistelmä

Testaus ja laadunvarmistus ovat tärkeitä aktiviteetteja kaikenlaisten teknologisten järjestelmien kehittämisessä. Vaikka maailmasta on tosiaankin tulossa aina vain globaalimpi, uniikit kyvykkyydet ovat yhtä tärkeitä kuin aina ennenkin – tai jopa tärkeämpiä kuin koskaan aikaisemmin, koska kilpailu on kovaa ja siinä erottuminen vaikeaa. Suomi ei voi – useista seikoista johtuen – olla samanlainen kuin muut maat. Yksi osa kilpailukykyämme on kykymme luoda globaaleille markkinoille hyviä, kaupaksi meneviä tuotteita ja järjestelmiä, joilla on kaikki vaadittavat ja myös ihastusta tuottavat ominaisuudet. Katsomalla asioita Suomen perspektiivistä, voimme saada uusia näkymiä tuotekehityksen toimintamalleihin ja niissä tarvittaviin kyvykkyyksiin. Toivotut kyvykkyydet tukevat meidän erityisen Suomemme menestystekijöitä – kuten hyvää osaamista, pelotonta suhtautumista innovointiin ja niiden pohja on Suomen muista maista erottavissa tekijöistä, kuten lavean osaamispohjan luovassa koulutusjärjestelmässä.

Tämä väitöskirja pyrkii tunnistamaan muutoksia ympäristössämme ja näkemään testaamiseen liittyviä kyvykkyyksiä, joita pitäisi aktiivisesti tukea tulevina vuosina.

Tutkimus oli laadullinen ja teoreettinen. Se perustui tuotekehitysympäristön mallintamiseen monilla tasoilla kattaen globaalit ilmiöt, kansalliset muutokset, muutokset yrityksissä, ja muutokset ohjelmistokehityksessä sekä testauksessa. Analysoitavat muutostekijät olivat konkreettisia ja ne valittiin subjektiivisesti, perustuen siihen, miten ne edustavat keskeisiä abstraktimpia muutoksia. Valinnoissa pyrittiin välttämään lyhytaikaisia hype-ilmiöitä. Tämä edellytti tutkijalta tiettyä epäformaalia tulkintaa, sillä mediatiedot heijastelevat aina ilmiöiden näkyviä piirteitä, jotka eivät kuitenkaan kerro riittävästä ilmiöiden edustamasta todellisesta muutoksesta. Esimerkki tästä on viimeaikainen hackathon-ilmiö, joka on vain kokeilukulttuurin ilmentymä.

Muutokset tiivistettiin muutos-kyvykkyys pilkkeisiin (change-competence snippets), jotka kuvaavat muutosta siihen liittyviä kyvykkyystarpeita ja kytkevät kyvykkyyksiä yhteen luoden verkon muutoksista ja kyvykkyyksistä. Pilkkeitä tunnistettiin 62 kappaletta. Erityyppisten kyvykkyyksien erottamiseksi luotiin tasojärjestelmä, joka eritteli laatuun liittyviin asioihin orientoivat, niiden ymmärtämiseen liittyvät ja asioiden tekemiseen liittyvät kyvykkyydet. Analyysin pohjaksi tehtiin ensin perinpohjainen analyysi testauksen nykytilanteesta, käsityksistä ja toimintamalleista. Toisen tyyppinen analyysi perustui toimintajärjestelmän kolmiomalliin, jota sovelletaan toimintatutkimuksessa. Siinä oleelliset kyvykkyydet tunnistettiin liittyen yksilöihin itseensä, testattavaan järjestelmään, tuotekehityksen tavoitteisiin, organisaatioon,

tiimityöhön, prosesseihin sekä välineisiin ja menetelmiin. Se analyysi korostaa sitä, miten näkökulma testaukseen oli organisationaalinen ja kulttuurinen: testaus nähtiin aktiviteettina, jolla organisaatio vastaa tarpeisiinsa, jota tehdään organisaation rakenteissa ja kulttuurissa. Reagointi pelkkiin puhtaisiin teknisiin ongelmiin insinööriratkaisuilla ei ole kestävää, vaikka osalla testauksella onkin teknologinen perusta.

Suomalaiselle testausyhteisölle tehtiin pieni kyselytutkimus, joka tuotti myös näkemyksiä keskeistä kyvykkyyksistä ja niiden välisistä suhteista. Eri lähestymistavat tuottivat, kuten saattoi odottaa, hieman erilaisia näkymiä kyvykkyyksiin. Muutostekijöiden analysointi eritasoisissa konteksteissa tuottaa enemmän tehtävä-orientoituneita näkyviä, kun taas toimintajärjestelmän tarkastelu syventää paikallisen toiminnan ymmärtämistä, Kumpikin täydentää toisiaan,

Löydösten synteesi tehtiin ryhmittämällä keskeisimmät kyvykkyydet tuotekehityksen vaiheisiin, liittämällä ne toimintajärjestelmän elementteihin ja luomalla kyvykkyyskimpaleita (competence lumps). Ne ovat konsepteja, jotka liittävät yhteen kuuluvia kyvykkyyksiä toisistaan riippumattomiin kokonaisuuksiin, joita voidaan käyttäen apuna luotaessa roolikuvauksia, koulutussisältöjä ja opetusta.

On tarve uudistaa vanhat testaukseen liittyvät stereotypiat. Testaus on yleensä koettu toiminnallisena testauksena tuotteen implementointivaiheessa, mutta käsitteen piiriin on otettava vahvemmin kaikki kokeelliset arvioinnit. Siten voidaan muodostaa kokonaisvaltainen työkalupakki, joka alkaa konseptitason kokeille, validoinneilla lean startup -tyylisessä kehittämisessä ja käyttäjäkokemuksen testauksella. Testaus kehittyy koko ajan käytännössä, mutta ollakseen tehokasta on löydettävä mentaalimalleja, jotka kattavat sen rikkaan kokonaisuuden.

Tietoturvallisuus ja käyttäjäkokemus ovat keskeisiä menestystekijöitä kaikessa uusien tuotteiden kehittämisessä ja niiden arviointi ja testaus ovat uusia testauksen ydinosaamisalueita. Uusi yhteiskunta ylipäätään edellyttää kaikilla tuotekehitykseen osallistuvilla olevan jonkinlaista liiketoimintaorientaatiota ja sen ymmärtämistä, jotta he voivat tukea paremmin organisaationsa tavoitteita. Monimutkaiset tekniset järjestelmät edellyttävät tietenkin hyviä teknisiä taitoja ja testiautomaatio-osaamista. Kyvykkyystarpeet ovat niin laajoja ja monimuotoisia, että yksittäiset henkilöt eivät voi hallita niitä kaikkia. Kyvykkyyskimpaleet auttavat tässä, sillä ne tarjoavat näkymän kyvykkyyksien sellaisiin kokonaisuuksiin, jotka ovat yksilölle realistisia. Mutta tilanteet muuttuvat jatkuvasti ja muutoksetkin muuttuvat! Siitä syystä tämä väitöskirja kannattaakin pidemmällä aikavälillä nähdä metodologisena työnä, joka esittää näkökulmia osaamistarpeiden tunnistamiseen erilaisissa muuttuvissa olosuhteissa.

Avainsanat Laatu, ohjelmistotestaus, laadunvarmistus, Suomi, kyvykkyys, kyvykkyysmallit, taidot, tiedot, tuotekehitys, ohjelmistokehitys, muutokset, tulevaisuus

Preface

This dissertation was made during August 2011 - December 2016 (some documenting details added in September 2017).

The work was on the side of teaching and various projects when there was some spare time form those. Well, actually it was started a year or two before that, when I, after working some years in quality consulting, started making inventory of what I understood about testing and how it had changed during decades.

Thanks to my supervisor, professor Hannu-Matti Järvinen for guidance for writing of the thesis. Thanks for early commenting also to associate professor Mika Katara who started as my supervisor, to professor Kari Systä and professor Petri Nokelainen (methodological critique only).

Tampere 11.9.2017

Matti Vuori

Contents

APPENDIX 5: Changes ranked by their effect on product development performance factors

List of figures

List of tables

Terms and abbreviations as used in this thesis

#A: Tag for ability to do the required actions. Type of competence used in this dissertation.

Activity system: An organisational context where humans work on an object, for outcomes, under rules, with division of labour, in a work community, using methods and tools.

Activity system triangle: A model of activity system elements as presented by Engeström (1999).

AI: Artificial Intelligence.

Best practice: A practice that is generally seen as valuable and suitable for use in any context, such as unit testing or continuous integration.

Change-competence snippet: Pattern-like structure that links together changes and related competence needs.

Community: A relationship system of several people who perceive themselves as members of the system and have a role in it. Example: the professional community of some profession, open source development community of one product, social media site's community. A company's personnel is a community, but not often referred to as one.

Competence: The ability to be able to produce desired results in a given context. Definition used in this dissertation.

Competence level: A measure of the possibility of capability usage for a task often in the cognitive dimension (from knowing to being able to apply knowledge in required action).

Competence lump: Concepts than combines related competences at various levels (orientation, understanding, ability) into an independent group that can be used as guidance for role definitions, training and education. Exist as sets that differentiate various lumps.

Competence model: System of differentiating different types or levels of competences or the elements of some type of competence.

Context: An action system with unique elements. It has unique principles and rules for activity. It has various states and situations, in which the interactions between acting elements differ. It may change to another context by some transformation caused by for example it having reached its temporary goals. It is a true system where every element needs to be present in order for it to work (whether the elements are explicitly defined or not).

Core competence: One competence a set of similar, which all define the competences expected from a person of a profession in all contexts.

Craftsman: An identity, mostly found in programmers, which emphasises quality of code and focuses all development efforts on working with code.

Cynefin: A system of differentiating contexts / domain based on their nature related to order and how well they are understood.

Domain: a) a broad context, b) an area of organisational life, c) a business area.

Ecosystem: A system where the participants are in economic relations with each other. May exist for example in a nation, a branch or a product platform.

Experiment: A procedure where a hypothesis is validated by testing, or one where testing is carried out to produce information for understanding a phenomenon.

Experimental culture: Culture that emphasises experiments above careful pre-planning as means for design.

Finland: A country in Northern Europe.

Hype: Idea or a paradigm that has popularity and publicity but which lacks critical review and which can be suspected to not be as solid as claimed.

ISTQB: International Software Testing Qualifications Board. International tester certificate provider. Currently the most widely used certification system. Handled by national organisations, in Finland Finnish Software Testing Board, FiSTB.

Lean: Originally refers to the manufacturing paradigm at Toyota, but often (perhaps incorrectly) means just something done incrementally and with low resources.

Lean Startup: Product development methodology where products are built from testing hypotheses with minimum versions of the concept and thus learning the customer preferences and behaviour. Despite the name, is not particularly based on Lean or suitable only for startups.

Maturity: The level of competence and internal standardisation of how relevant practices (such as testing-related practices) are executed in an organisation.

Maturity model: Model of maturity levels and the factors that position an organisation or person at given level. Maturity models exist for software development, testing, management and other areas.

MVP: Minimum viable product. Product version that offers minimal functionality or features that is used to test customer’s preferences and behaviour. A term brought into wide use by the Lean Startup methodology.

#O: Tag for Orientation to an area of activity, thinking or discipline. Type of competence used in this dissertation.

OWASP: Open Web Application Security Project. Project for providing guidance about security risks and their testing. Also offers similar guidance for mobile application development as a sub-project.

Pattern: Recurring structure or relationship between system elements or in an activity. Usually means something that has been demonstrated to happen, but may also refer to potential activity.

PESTLE: Method for analysing changes in futures research. PESTLE refer to Political, Economic, Social, Technical, Legal and Environmental factors. Variations include PEST.

Product development: Set of activities aiming at producing a product to market, including analysis of needs and concept development, but not deployment.

QA: Quality Assurance. Activity that aims at assuring that the product and its creation process are of sufficient, including required, quality.

Quality (of a product to someone): The measure of how well a product, or its characteristic, has value, from some perspective, to someone.

Ripple: Something that appears to be a relevant change in the environment or industrial practices, but which turns out to be a short-term phenomenon and is evened out by a long-term trend.

Role (of a person in an activity): The set or shared expectations toward a person in a position in a team or group. Can be assigned or evolve dynamically due to team dynamics.

Role (of a thing to a person or role): How a thing directly or indirectly influences a person's actions, thoughts or beliefs in a context.

Safety-critical system: A system that has safety hazards that needs to be developed and tested accordingly.

Scripted testing: Testing which is determined by a pre-written test cases and test procedures. Refers often to such manual testing whereas similar test automation is automated testing.

Scripting: Creating simple test execution automation by writing small programs in a scripting language.

Sensemaking: The process of understanding the concept, principles and rough level functioning of a new system or phenomenon.

SET: Software Engineer in Test. A role or even a paradigm that refers to people in development that create test-enabling infrastructure, tools, test designs and so on.

Smart creative: A term coined by Google to refer to an ideal ICT worker in their context.

SME: Small or Medium-size Enterprise.

Software development: Set of activities aiming at producing software. Not a synonym of product development, but can be a subset of it.

Tester: Anyone who tests at one particular time. Does not imply occupation, static organisational role or identity. Thus, tester can, besides a professional tester, a marketing manager, a software developer or a CEO.

Testing: Empirical activity that produces information about the quality of a system, the system being a concept, prototype, implementation, competitor product or any else.

T-shaped professional / T-shaped competence profile: A person has one main competence area, which is complemented by other, secondary areas. Related types include pi-shaped and H-shaped, and dash-shaped (-) which refers to generalists. Those are rarely seen in general use.

#U: Tag for understanding of a need to do something regarding an area of activity, thinking or discipline. Type of competence used in this dissertation.

UX: User experience.

V&V: Verification & Validation.

Validation testing: Testing for assessing how well a given artefact meets its expectations or real-life requirements.

Verification testing: Testing for assessing how well a given artefact meets its specified characteristics.

Viewpoint: A set of paradigms, assumptions and conditions that guide an actor in focusing on relevant issues. Does not imply valuation.

Virtualisation: Executing workflows and testing in virtual computer environments, which may be created rapidly for just one execution of a test.

1 Introduction

1.1 Background

The readers of this dissertation know that testing and quality assurance are important activities in the development and deployment of all technological systems. They are the means by which systems of at least sufficient quality can be created and successful business build around those systems. The activities consume much time and effort and require people with many types of skills and knowledge to carry out.

The national challenges in Finland relate to our global competitiveness. While the world is indeed becoming more global, unique competences and capabilities are as important as ever – or even more so, because of the fierce competitions. Due to various factors, we cannot be similar to others.

One part of the competitiveness is our ability to create for the global market good, marketable systems that have the qualities required. Sometimes the qualities are a customer centric criteria and sometimes given by laws and standards, for example with safety-critical products there are plenty of very demanding requirements for how their safety and quality in general should be assured during their development. This is by no means easy. Technological systems tend to become more and more complex and at the same time the customer's expectations for quality have risen. Expectations for any area of quality are higher, but at the same time the expected areas of quality are expanding. Where at the beginning of the software engineering just functional correctness and reliability were the most important factors, now we need to consider usability, user experience, security and interoperability, and the list of the factors keep growing as the culture of technology evolves.

But at the same time we see that the resources for all this do not grow at the same rate. For example, complexity of the systems may grow exponentially, but new systems often need to be created with smaller organisations in less time than before. Some may even think that this situation is impossible and the software crisis is insolvable, but that kind of conclusion is not something that we should accept. Instead, we need to think how we could do things differently, more efficiently and more smartly. Many domains face these challenges in different ways. The following are some examples.

The developers of safety-critical systems, such as machinery and automation systems must use the best technologies but develop the systems rapidly to the market, while filling all safety requirements and all customer requirements, providing some unique value that is implemented in an excellent way.

The developers of machinery, for example moving work machines, face the larger problem of changing the whole approach of product development from making machines made of steel into complex computing and communications platforms. There are the changes of managing the development and how to make the software systems safe and reliable.

The developers of mass market products need to bring some excellent value to their products, often using new and immature platform technologies. The lean organisations work in a demanding situation where they need to beat the competitors, reach important time frames for product launches and win over the millions of potential new customers.

The startup companies build their practices from scratch. They need to be very agile in the first phases of the company's life and after that stabilize the practices that best suit the company's business goals, while still keeping lean and adaptive.

Developers of information systems face the problems in the size of the system and in integrating even tens of separate systems to a well working whole that can tolerate the heavy loads from usage is a secure way that maximizes business process productivity and user satisfactions.

Of course, the development of competences has always been on the "national agenda". Traditional quality assurance and management has a long history and has been nicely complementing the engineering processes. However, software quality has a shorter history due to even the paradigms being quite new. Only in the 1990's was a comprehensive understanding created about how software quality should be managed in development processes and only in the early 2000's was testing education more widely introduced in software engineering education. For example, a study of Surakka (2005) in 2004-2005 on needs assessment of software systems graduates only listed testing as competence area, but no relevant courses for it. In this context it is sufficient to assume that it had some role in software engineering courses. However, in Finland perhaps the first software testing course was in Helsinki University of Technology in 1998 and TUT started in 2002 their course where testing was tightly linked to the software engineering process and the software development lifecycle.

But shared competence development does not need to be based on the actions of the education system. The Finnish testing community has seen the need to build and share the competences and the key tester society TestausOSY – Finnish Association of Software Testing – has for a decade been very active in arranging practitioner level seminars (peer conferences) and similar.

Testing has always been a reflection of the constructive engineering principles and practices. Because of that, the common thinking about testing still reflects on the days of earlier, waterfall type processes and practices. When agile development was introduced, the testing community was left to wonder how testing should be done in the new context. This vacuum of lacking understanding caused many repercussions such as over-emphasis on lower level testing and lacking system level testing. Now a balance is finally becoming found, but this shows how fragile cultures are in this regard, when deep understanding about things is missing. The surest and most stable thing in our environment is constant change, and quality and testing should or could be the approaches that help us through any transformations, not things that make the transformations more difficult.

From the domain of organizational development, we also know that any practices the companies apply should optimally be selected so that they suit the company's goals, culture and unique ways of acting best. There have been and continue to be approaches for standardizing testing to a one-size-fits-all mode, but it should be clear that a startup and a large established company need different testing practices.

It is essential that when any paradigm changes occur, Finland is the first one to tackle them properly and perhaps is the one competent enough to build business around them globally. Some of the bases for that include our strict professional attitude about quality and project work in general, not to mention high technological engineering competence and the ability to implement new ideas and concepts. This can also be the basis on which testing and quality assurance competencies could be built on in a unique way.

Sometimes it is thought that testing competencies could be outsourced and many companies have tried that. That was seen as possible during the days when software development was more based on waterfall model or slower development cycles with low amount of test automation. In those circumstances, it made some sense to think that a software version would be sent to another company every month or so for a team of testers to test (between the testing round the testers might do nothing!). Today, agile development has given us more rapid development and testing activities are mostly carried out in the development teams. If special testing competencies are needed, the ease of communication makes companies favour services that are physically and culturally close to the development activity. The outsourcing today will more often be about testing infrastructure – such as a mobile phone farm – than actual testing work.

So there are many challenges related to the changing environment, technologies, expanding areas of quality and so on. Competence is the key factor in overcoming the current challenges and the future challenges – which we don't even know yet.

But when it is a question of competences, are they not similar in all countries? Are not the practices in engineering and product development similar globally and we should concentrate on global understanding and just apply the best practices in Finland? Organisational development experts usually emphasise that there are no best practices. The ways of doing things should be developed based on the local needs and culture. It is a mistake to do otherwise. Because we think that Finland in special, we need to base the research on that idea. It is a bit of a paradox that in this age of globalisation we should look into uniqueness as strength. By looking at things from Finland's perspective, we will gain a unique insight into ways of doing things and the related competences that:

- Support the success factors of our unique Finland, such as the competences and fearless relation towards innovation.
- Are supported by the unique good qualities of Finland, such as good education and thus opportunities for building competences in technology, and management.
- Suit our other characteristics, be they whatever they might be, such as the often seen supposed dominantly silent and introvert character of the people,
- Bring extra added value to everything that we do.

In general, the unique characteristics can be found in national culture, prevailing organisational culture, the structures of the industry, economic constraints, geological location, size of the home market, and so on. Some of them are currently as they have developed during decades, but some may be things that we wish to actively develop. In chapter 2 we will look in more detail to the ways that Finland is unique and also what the wishes for the development of Finland are.

What those competencies might be is the focus of this dissertation. When we understand the competences better than now, we can both apply and develop them better than we do currently. However, that is the focus of other research. Now we just try to identify the competences and understand them as fully as possible.

1.2 The research frame, hypotheses and goals

The target environment is software production: Development of software products, information systems and programs embedded in products, and quality assurance in that context carried out by testing and other means, such as reviews. This activity is essential for the quality of the product developed and a requirement for marketing of the products. Quality assurance is a central activity of software development, into which often tens of per cent of the development budget are spent. Often it also has a great influence on the speed of product development and the ability to bring products into the market at a right time. The branches that the research is targeted to are the product development branches, including development of applications, digital devices and machinery.

The research is guided by this hypothesis, or perhaps expectation, as it is often emphasised that qualitative, theoretical research should formulate the hypotheses during the research (e.g. Alasuutari, 2011):

> "It is possible to assess the current views regarding personal and organisational competence and to formulate a new view that gives an improved insight to the current and future needs for competence and capability. That view will thus give guidance to the improvement of the competencies and capabilities. That is turn will lead to better performance of testing and quality assurance in software development."

Of course, any such views and architectures do not help us much. That's why there is a practical goal of finding, with some reliability, what the necessary personnel competencies and organisational capabilities that makes it possible to:

- Produce in Finland testing that fully meets the needs of highly demanding system and product development.
- Produces global competitiveness by which testing services can be created, utilizing unique competencies. Those services can be internal, national or exported.
- Support bringing testing back close to the new generation of innovative ICT development action. There was and still is an era when testing services were outsourced to other countries based on cost. There is a need to improve quality of services to help keep the services close, thus improving collaboration in systems development and agile support for innovation and business.
- Integrating the automated, model-based and manual exploratory testing approaches and increasing the understanding about their complementary benefits.

"Some reliability" is emphasised here, because all possible futures are just scenarios and one should lock into them too tightly. Similarly, any activities, be they creation or validation, and their competence needs, depend on what's really going to happen. There is a need for "loosely coupled" thinking that enables adapting to whatever will happen.

Traditionally, it has been noted that the requirements for testing may be different in product business and in developing tailored information systems on project basis, and in offering testing services. In this dissertation, a sharp distinction is not made. The differences are analysed in chapters about those areas when the changes in the operating environment of testing are assessed.

The goal is not to find a stable “grand vision”, but more to learn how to find a good vision that evolves when the environment again changes. Here, the viewpoint to testing is that it is not about processes and tools, but how people and organization think and act and can advance their thinking and actions. There is plenty of hype around about organizations, but hype is not something that top companies should trust. “Hype keywords” change every year, because consultants make money on them – not on the long-term success of companies. Best practices are how everyone else does things. To be better than the rest, we need to find something unique, something that brings out the best in us.

What will be found, will be a hypothesis in nature. Qualitative research produces hypotheses and that is exactly what is needed. We also need new ideas. Research (with the exception of constructive research) is often criticised for just reporting the status quo. This kind of research should be geared for producing fresh ideas for the future.

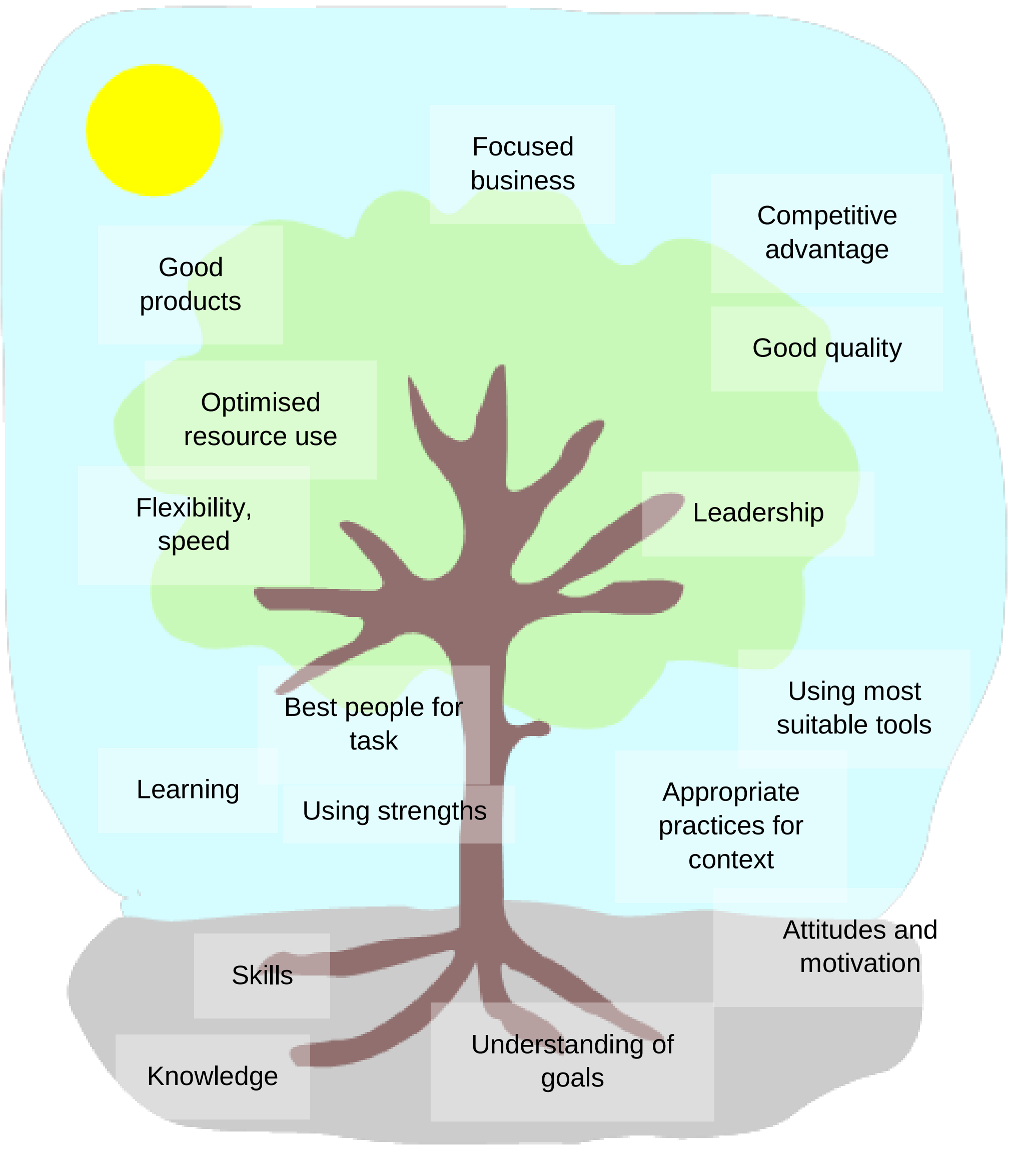


Figure 1. Competences bring benefits (illustrative figure).

The scope raised some criticism in its early phase for its largish scope and large issue at hand. The author thinks about this in the same way as famous management and strategy researcher Mintzberg (2005): "Doctoral students are told in their research to address some manageable issue, take a small piece of a large issue. I disagree. The really interesting dissertations address really big issues". We have some large issues in Finland that need to be tackled and they are issues that require many viewpoints and approaches that suit the problem scope[1].

[1] It also so happens that management and testing have some similarities in their nature so we may have other learning from Mintzberg – written out in this dissertation or not.

1.3 Research approach

First, let's typify the research based on goals, as that lays ground to the methodological choices. Runeson & Hölst (2009) present these types of research:

- Exploratory – finding out what is happening, seeking new insights and generating ideas and hypotheses for new research.
- Descriptive – portraying a situation or phenomenon.
- Explanatory – seeking an explanation of a situation or a problem, mostly but not necessary in the form of a causal relationship.
- Improving – trying to improve a certain aspect of the studied phenomenon.

Which type is this dissertation? It is exploratory as it analyses the changes in various activities and as theoretical provides ideas for development and empirical validation. It is also descriptive, as it presents the current situation and the changes. It is improving, as it outlines ideas for how the challenges related to changes could be tackled and new opportunities to be utilised. Therefore, it is also explanatory as it shows the relationships between issues.

The research was qualitative and theoretical by nature. The research utilised the research strategies of Grounded Theory methodology by Corbin & Strauss (2008). That method’s world view is that the world is very complex. Capturing of the complexity in research is essential. There are no simple explanations for things. That is why this dissertation “dissects” many issues quite thoroughly. Otherwise there is a danger of falling in the trap of simplifications and thus making false conclusions.

The method emphasises intuition and experience. The author has experience and draws from that. Of course, all experiences and intuitions are personal and the method description even goes as far as to imply that objectivity is a myth. Objectivity is a myth and one should instead focus on sensitivity: How the researcher is "tuned in" to the data, the context. Puusa (2011) sees that two researchers would never end up with the same conclusions even if they had the same qualitative materials. People don't perceive things in the same way, or interpret them the same way. But the iterative nature of research should help researchers narrow their thinking towards at least "possible interpretations".

But one must keep track of the data. The dissertation focuses on areas that don’t have much applicable “hard” data. Instead the data is spread in the communication channels (like social media services), books and magazines. Those could be a target of systematic coding, but that was seen to be too resource-consuming in this case. Instead they are seen as material for "observing". Where in the industrial age, actors were observed in their physical space, now social media, for example Twitter, can be used to observe how people "think out loud" and express and form the culture with their use language. The author has for several years now been observing tens of Twitter members that are relevant for this dissertation. Of course, Twitter can form a very biased view and is prone for various hypes. But that can be seen as a good thing too: it can expose the hypes, make them visible and thus object for scrutiny.

Grounded theory emphasises freedom from preoccupations, but generally most qualitative research is started with the researcher's preliminary understanding of the issues, which is used as tool to get started (Puusa & Juuti, 2011), but those should be turned into new hypothesis during the cyclic rounds of research. This is the case in this dissertation, where the researcher has plenty of preliminary understanding and needs to reflect any new finding against those, in search of new hypotheses.

One research strategy of the method is making comparisons. The dissertation looks into many fundamental issues from various perspectives, but sometimes does not make a summary of the “truth” of the issue. That is because there are none. The method is based on a postmodern reality where there may be many truths. This is often illustrated in figures such as Figure 2[2].

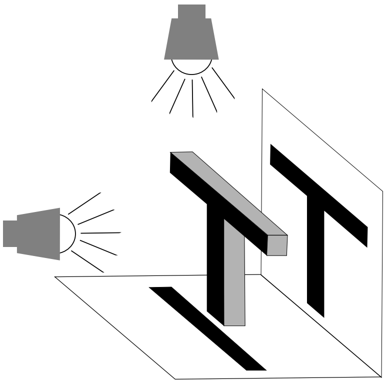


Figure 2. Different viewpoints can show a very different reality.

There is a general hazard of research that researchers only use one viewpoint into a rich reality. One viewpoint is easy to manage and validate, but as a result there will be an illusion of validity. That does not match the principles of qualitative research.

The Grounded Theory method, as many qualitative methods, emphasises using diagrams to show relations between things. Those have been used widely, as have been analyses of issues, reported in listings and tables. Those are essential for capturing the complexity of the issues at hand.

The world according to the method is contextual and that is why much of the analyses are done on various contexts, presenting them, dissecting them and trying to find their

[2] Such figures may claim inspiration from the cover of cover of Douglas R. Hofstadter’s book Gödel, Escher, Bach: An Eternal Golden Braid from 1979.

essentials. The context models are sometimes practical and sometimes theoretical. There was an aim of keeping those views in balance.

As a summary of what the method requires from the researcher, the author concluded in his review of the Grounded Theory methodology (Vuori, 2012) these important qualities that also should show themselves in this dissertation:

- Intellectualism. Understanding of multiple paradigms in the core issues. Multi-disciplinary approach. Many abstraction levels of action; various viewpoints; no (big) black areas.
- Understanding of psychology and culture – or at least inclination to understand.
- Understanding of language, what the “texts” around us express.
- Freedom of thought.
- Tolerance of uncertainty – must let the facts and stories lead where-ever they might.
- Ethics – as many stories may be built from data, one needs to remain reflective and not promote own agenda.

The “black holes” are a problem in research. A common problem in software engineering research is the researcher closing eyes from issues that have not been described in literature or that have been only covered in scientific literature. That causes the research to be based on a too narrow set of ideas. The author has tried to avoid that, but it is obviously a two-edged sword: it is necessary to rely on own intuition and experience that may lead to other problems. To handle that, the dissertation presents details and analyses as rationales for its conclusions. The research was iterative by nature. To reduce the danger of being too attached to current ideas and structures, the research used a “breadth first” exploratory strategy, where the whole domain of testing, its environment and changes was covered lightly and after that iteratively more in detail. That allowed for new phenomena to emerge in the discussion and to become objects of analysis. This is visualised in Figure 3, Figure 4 and Figure 5.

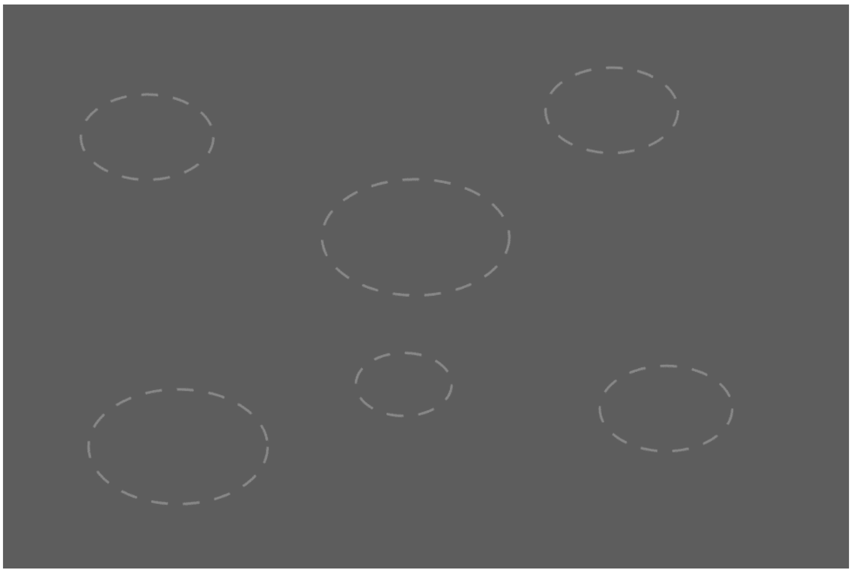


Figure 3. Open setting in the beginning.

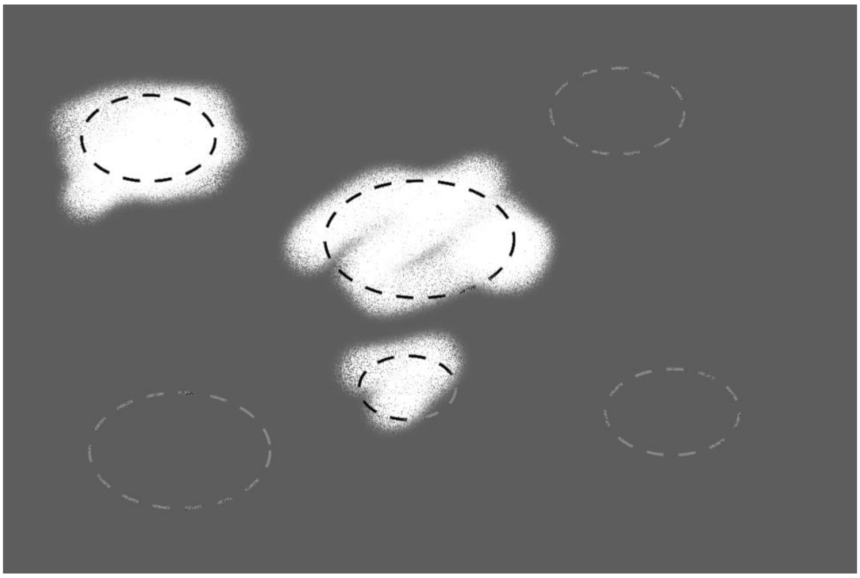


Figure 4. Analysis proceeds and issues emerge.

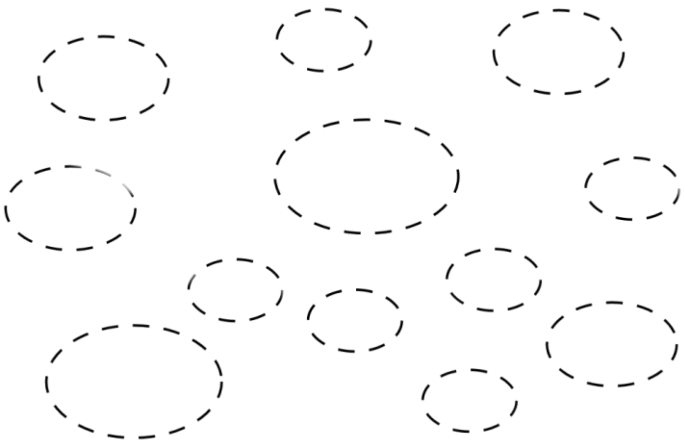


Figure 5. Overall picture becomes clear.

Another challenge is the richness of the target domain. When we look into quality and testing from top down (and in this case the top is higher than usual), new worlds emerge almost in fractal manner. Details will be handled in a case by case manner, as deeply as any topic seems to require for understanding it – both by the author and by the readers. But the world really is rich and complex, as the philosophy of grounded theory reminds us. The analyses, in sometimes detailed way, are done in order to expose the richness, which is always in danger to be hidden behind coarse generalisations. That exposure is a key element in qualitative research and also allows the reader to assess the assumptions behind any conclusions by the author.

How the richness is turned into a network of changes and synthesis of relevant competences is an important form of analysis, and like traditional “coding” of texts in grounded theory, it finds order, connections and practical meanings in the sometimes chaotic reality under analysis.

As the research is theoretical in nature, it is, however, based on analysis of ideas more than analysis of what is happening right now. It analyses real and potential changes in the environments where testing is done, but testing always follow by some delay its drivers and therefore this kind of research must provide hypotheses and “problems” that the practice – or experimental research – will later solve.

All in all, this dissertation follows the philosophy of grounded theory more than its common “recipes”.

The concrete flow of the research was as follows:

- Literature surveys about competence.

- Analysis and description of the current state of testing, including the tradition and many different aspects of testing.
- Creation of the context models for the environment where testing is done, including a layered model ranging from global issues to low-level work at workplaces and selection of models that describe the activity system of someone who tests.
- First round of analysis of the changes in the environment in order to start identifying the issues that need analysis. This was based on personal views and earlier collection of changes (Vuori, 2014d, originally from 2010). Reasons for that are twofold. First, there are no reliable sources in this kind time of change. Issues need to be extracted from observations. Second, this is a form of modelling and models are always the modeller’s an interpretation of the reality – both in science and in testing. Models are also always incomplete and flawed. Cilliers (2006) expresses it this way: "No matter how we construct the model, it will be flawed, and what is more, we do not know in which way it is flawed." Note that incompleteness is a good thing both in research and in testing, as it makes model manageable in size and in complexity.
- There was one survey to testing experts in Finland. The research is avoiding the problems of using large volume surveys, because that will turn the results into a description of the mediocrity and because people usually see the opportunities of the future as answers to problems they had a couple of years ago. That is because they only understand properly the past, and can see resolutions only in what they understand[3]. Instead, we need to rely on the researcher’s analysis and vision. The survey was used to in deductive mode to gain insight of testing experts’ thinking and to deductively construct small models of their views, but this was only to see if there is any new “weak signals”, not to create any real valid basis as such. The survey is described more in a chapter 6.
- Second phase of analysis of the changes.
- Analysis of the competences implied by the changes in relation to a model of the activity system at the workplace.
- Creation of the concept of “change-competence snippet”, a pattern language that helps in extracting the competences implied by the changes in compact form and linking the changes – which changes have an effect with what others.
- Analysis round of the changes using the change-competence snippets and creation of visualisations based on those.

[3] This very same effect shows in the world of technology. Industrial Internet is a new paradigm that offers a huge number of opportunities, but even knowledgeable people have presented their wishes for it to be in getting big data out of devices – a need from decades ago.

- Forming of a synthesis of the whole.
- Of course, during the whole period there have been observations about the testing culture by media and in the related research projects the researcher participated in.

There is in general a danger of using various models and structures to rigidly in this kind of research, to make it “more scientific”. We do use models, but understand that they are simplifications, so they need to be applied in a free form in this kind of research.

1.4 Elements of subjectivity

Every research, theoretical or not, has some subjective assumptions by the researcher, which will influence the stories created. The following ones are what the author has identified. They might have solid proof available, which would make them "facts", but that is beside the point here. Many of these issues will no doubt be revisited later in the text.

It is assumed that testing and quality assurance are good for products, businesses and the society. While there is some fluctuation in the attitudes of companies, the amount they are practiced increases continuously. That is necessary as the complexity of systems increases and the expected quality (including reliability and security) rises each year. Thus, the quality practices also form an important basis for risk management.

Therefore, there is a need for more professional competences, while usage of those will and should vary based on the business processes, business priorities and the lifecycle phases of companies.

There is also "meta-ignorance" about testing, that is, lack of understanding it importance and its relation with product development activities, and the general ideas and possibilities of it. Part of this is caused by people's mindsets being formed in history, in simple situations during an era, when testing too was seemingly simple because of the conditions (closed, focused context with narrow approach to testing).

Testing is primarily not engineering even in an engineering context, but organisational human activity and needs to be treated as such. It is also part of a systemic whole of activities in organisations and need to avoid local optimisation, need to be considered in contexts, as elements of the overall system. That is why, when researching and/or developing testing, much of the analysis needs to be done on the activities surrounding it. That is absolutely critical.

Thus, research of testing is generally not technical research, or computing science research, but organisational research that has as its subject and context the activity or product development.

Due to historical reasons, much of the thinking about testing considers it as part of the software engineering process, not the higher level of product development process. That needs to change.

1.5 Quality criteria of the research

There is a tradition of looking into the validity and reliability of research at the end of a dissertation. But as Aaltio & Puusa (2011) noted, the concepts are designed for quantitative research, they are not terms that work well for qualitative research. The science and the reader are better served by using more concrete criteria developed for this particular context (note the analogue with testing). The author thinks that the following criteria are most relevant in the evaluation of this dissertation:

- The rationale, basis for the competences architectures used. Are they founded in a solid thinking?
- How well do the competence architectures help us understand the issues of competence in this context?
- How sensitive is the analysis regarding how the world changes? After all, this is about the future and that is generally somewhat unknown.
- Quality of the analysis of the selected changes.
- How well does the work overall help us understand the contexts in which testing is done, their characteristics and what testing could be like in them?
- Reusability of the methods used How well can the resulting views to the new competences be used in education, training or in competence development in companies?

A research can't fill all its expectations. There are some inherent limitations in any research. One of them is Thorngate's postulate of commensurate complexity, which states that:

> *"The impostulate of theoretical simplicity dictates that we shall never see a general, simple, accurate theory of social behaviour. In order to increase both generality and accuracy, the complexity of our theories must necessarily be increased." (Weick, 1999).*

Weick (1979) has transformed the postulate to apply to the results of research and used a clock as an analogy to visualize that research necessarily needs to focus on some quality factors at the cost of others. If the research is good in two respects, it will be lacking in the third. Now, let's have a clock where there is one hand showing the goals of research based on generality, accuracy and simplicity, see Figure 6.

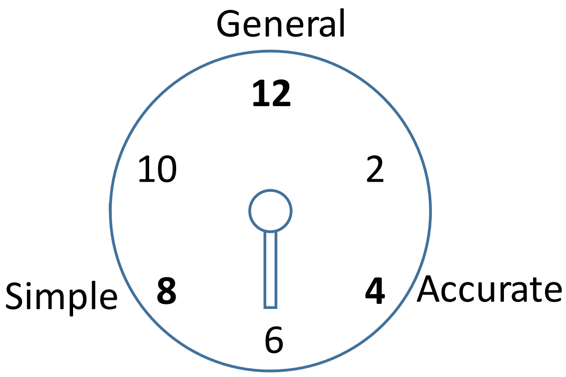


Figure 6. Weick's (1979) clock showing the trade-offs in research.

The clock visualizes that if research that aims to be accurate and simple, the hand pointing at 6 o'clock, results would not be generally applicable. The simplicity would reduce the results to apply only to some situations and would not reflect the various contexts we live and work in. In any research that is about organisations, including research of testing, accuracy and simplicity are not possibly to find. Qualitative research needs to stay at somewhat abstracted level. If research that aims to be general and simple, the hand pointing at 10 o'clock, results would not be accurate. Accuracy requires some complexity. Complexity also results from broadening the scope of research, as it brings in more contexts with their detail. If they were to be included in a formal model of things, the results would be so complex that it could not be used. This dissertation tries to reduce complexity by having loosely connected viewpoints and trying to find a suitable level of formalism. If research that aims to be general and accurate, the hand pointing 2 o'clock, results would not be simple any more. This would be the case of having one large model of how organisations work, how the contexts change and how that reflects into competences. It was already mentioned that it would be problematic.[4]

So, there are compromises to be accepted, resulting from the nature of the world and the science.

[4] Scientific researchers don't often note such inherent limitations, for various reasons, but in the fields of software engineering and project managements various triangles of compromises are often seen.

1.6 Related work – or the lack of it

It seems that there are no similar studies. Obviously, there are lots of studies about the future of testing technologies e.g. for automated functional testing (such as Bertolino, 2007, but that are has little relevance here. There are generic futures studies about Finland and general competence needs in occupations, but those are done in the context of national policy development or the context of regional development and are consultative in nature. Reflections on the competences of testers are found in the consulting domain (a defining mark of a consultant is that she outlines the future in an industrial meeting), and those are not "related work" to build on, but anecdotic raw material about our culture. The future of organisations is naturally a common research theme and an important area for this research too, and we will refer to that area as appropriate. Historical studies are always relevant when assessing the future. Pohjalainen (2007) writes about the history of software testing Finland in 1950-2000, mostly based on experts' interviews.

The status of related scientific work in this area is understandable when we consider some factors.

- The fragmentation of research. Testing researchers would reach outside their primary domain if they would analyse the contextual issues in product development or organisational development in a way that this dissertation does. And even the research in testing is divided. Researches feel comfortable only in their own are, be it test automation, testing or quality assurance processes or exploratory testing. There are few researchers who are specialised in generalism...
- Testing has been seen as methodological problem and method develop by innovation – some company or consultant thinks up a new method which is published; or an existing phenomenon is turned into a named practice (such as exploratory testing). Thus, new testing paradigms enter the focus of research only after they have been taken into use.
- Development of testing has also always been adaptation to the product or development technology and to the product development practices (model-based development forming basis for model-based testing; the radical team-organisational change brought new practices also for manual testing).

There is a tendency to understand issues only when they are a solid practice or solidifying into one and that applies to research too. That is why any future-reaching studies are so valuable, if they are done well and not just extrapolating one isolated viewpoint or paradigm,

In general, this research seems to be loosely linked into many research areas and will need to integrate their findings into a new setting instead of carrying directly on what others have achieved.

1.7 Personal motivation and background

A common question during defences seem lately to be, why you picked that subject? So, let's get that in writing too, as in should interest the readers and in this case it has even relevance to the subject of the dissertation.

This work is about developing the quality of how we do things at various levels and that has interested me during my working life, having worked on workplace design principles, risk management, product usability and later with software processes and testing. There is also the element of looking just a bit into the future, which is also a part of my past (the development of future products was one research topic many years ago) and a very interesting are. I have had a hobby of reading about organisational strategy and this work is very strategical from many viewpoints. Those are the interest areas that overlap right here in the focus of this work.

I have been involved in the field of testing for more than a decade and watched how the software development world has been very bad at utilising the competences available and regressed into practices that are not very helpful for any or us. Understanding why the situation is like that and what could be done about is something I have been doing every now and then. In this regards this work is a continuation of a "personal knowledge management" process. When my work in a company ended I begun to summarise what I had learned during that phase and self-published on my web site many analyses of testing. Some of those "white papers" are used as raw material in this book. In a way, this work closes one period on personal growth in knowledge and continues the aim and process of sharing the knowledge.

So, there was heart-felt motivation and an opportunity to apply many familiar disciplines that I like for an interesting task.

During and at the end of the dissertation we shall meet various ideas about the larger context of quality management and assessment and due to that it should at this point be noted that the author has experiences from many different related contexts besides software testing, among others the following:

- Safety assessments of production and power plants (work).
- Ergonomic assessment of power tools – which are in today's terminology "non-digital mobile devices" (research).
- Usability assessments of software products (research and service provision, training).
- Development methodology for future products (research, toolbox production).
- Human errors and accident investigation (research and investigation work).
- Quality manager's occupation (work).

- Software development process assessment for quality management or maturity systems (work).
- Assessment of quality management systems and tool development for that (work).
- Project crisis auditing (work).
- Risk management from many perspectives (research, toolbox production).
- Development of safety-critical systems (research).
- Requirement specification (research, assessment, training).
- Ethics in information systems development (research, training).
- Software development at startup companies (research).
- Development of software products and information systems (work and hobby).

Those experiences provide vast practical knowledge about the issues presented in this work.

1.8 Structure of this dissertation

In this dissertation, we start building the story by first defining some key concepts in Chapter 2 in order to "set the stage" for later analyses. The ideas about competence and testing are presented and also crucially the concept of concept, which will be used as a main tool in the research. Also, Chapter 2 looks into what we assume about Finland and what are the special characteristics of the nation and should influence testing.

Chapter 3 lays out the competence model used in the work. How competences relate to the changes in the environment and how this relation will be used in the work. Also, the competence level framework used will be introduced.

Future is always built on top of history and today and those need to be understood properly. That is why Chapter 4 presents a through look into testing as it currently. The chapter looks into testing from many viewpoints aiming to reveal that testing is not a simple task, but a complex weave of thinking patterns and cultures.

The idea in this work is that the environment is changing and testing needs to reflect that. Chapter 5 contains the analysis of changes at various levels from global issues to technology and software development processes, not to forget the testing itself. The chapter is structured on a layered way in a top-down fashion. The chapter includes the change-competence snippets that link together changes and related competence needs. The chapter aims to produce a view of what is happening and what will happen and what kind of competences respond to that.

Chapter 6 reports the short survey made to Finnish testing community. It contains various graphs that visualise the competence chains that will for basis for action and actually produce business results.

Finally, in Chapter 7, we get to evaluating of what we have collected. The chapter first takes a summarising look into what kind of activity testing is and into the main new competence needs in product development. Then it takes the alternative view of structuring competences around the action research triangle model of a work system. Next, the synthesised competence lumps are presented, which summarise manageable groups of competences that could form basis for future professionals. After that, a reflective look is taken to the “core competences” in testing and finally the dissertation itself is assessed.

There are several appendices.

- Appendix 1 lists the contents of the foundation level ISTQB syllabus, as it is the baseline for traditional testing thinking.
- Appending 2 lists the raw answers to the tester community survey, which were analysed in Chapter 6.
- Appendix 3 lists the change-competence snippets from Chapter 5.
- Appendix 4 lists the competences referenced in change-competence snippets
- Appendix 5 takes the same change-competence snippets and reflects how they relate to various performance factors of product development.

2 Setting the stage

2.1 General

Research like this is a process and it should also create a process in the mind of the reader of the dissertation. For both of them to happen, we need some stable points from where we start to grow our understanding. We need to define some concept that we can start working with. The definition can change later, and perhaps should change later – if they don't we may have not shed sufficient new light to them and the context we are using them. We need to outline the essence of testing and quality. Because this is about Finland, we also need to write about how we see Finland progressing in general, during the coming years.

2.2 What do we mean by competence?

2.2.1 What does the literature say about it?

First a note about how the term is written: It exists in two forms, competence and competency, of which the first one is preferred. Sometimes the two words mean different things, as we shall see later. Still, mostly they are synonyms.

The concept of competence was used first in the 1970's but we will look here at its more recent usage. The term was made widely used by Hamel & Prahalad (1992) in management and industrial cultures. According to them, competence is defined as the ability to do a certain work task with the help of means and support provided by the organization. Competence is proved by practical achievements. This is an important concept, as previously "skills" of "knowledge" were mostly used as a person's measure, but now there was a new term that combined many elements that lead to performance – including experience and values. Hamel & Prahalad were mostly interested in the term as a basis for "core competence", which would provide companies an edge in competition, which is very much the question in this research too – companies providing testing services need to be able to do it better than others.

Woodruffe (1993) differentiates the meaning of the terms “competency” and “competence”. For competency they give a definition: “A competency is the set of behaviour patterns that the incumbent needs to bring to a position in order to perform its tasks and functions with competence”. Yet, he fails to define competence. He shows lists of various competencies and criteria for the terms used for particular competencies – after all, they are critical terms with which people communicate about abilities and skills.

Moore et al. (2002) look into the usage of the terms, note confusion in the usage, and give a suggestion that competence is an area of work, competency is the behaviour(s) supporting the area of work and competencies are the attributes contributing that behaviour. About that we take a practical stance that the concepts just relate to different aspects of work – from goals and tasks to subtasks and the abilities involved. That is the approach used later in this research and we will clarify that later in a model that is focused for this context.

Athey & Orth (1999) define competency as “a set of observable performance dimensions, including individual knowledge, skills, attitudes, and behaviours, as well as collective team, process, and organizational capabilities that are linked to high performance, and provide the organization with sustainable competitive advantage.” And thus, according to them, based on this definition, competencies may include a wide range of capabilities of an individual, team or an organizational that include knowledge or skills associated with current job performance, emerging knowledge or skills required for future success, intellectual or behavioural best practices of high performing people or teams, process capabilities that enhance organizational or business performance, and new ways of thinking or behaving that provide distinctive competitive advantage. They also emphasise that in competency the core idea is to actually be able to do so and verify that, instead of relying to assumptions. The article also includes a list of trends in the evolution of competence methods, showing where the competence thinking was directed at that time:

- Trend 1: Demand for More Participative Competency Approaches
- Trend 2: Shift Toward Short-Cycle Competency Methods
- Trend 3: Increasing Emphasis on Emerging Future Competencies
- Trend 4: Increasing Focus on Team and Process Competencies
- Trend 5: Transition to an Organizational Learning Perspective

These all seem very valid even nowadays.

The concepts used by Athey & Orth (1999) are not sufficient terms to describe the phenomena. In order to use the appropriate terms in the right places, we need to remember the concepts such as “ability”, “capability”, “skills”, “knowhow” etc. The problem is that each of the terms has various meanings and there are no standard references to define them. Indeed, Helakorpi (2005) refers to the world of competence-related terms as a “jungle”. His book does its best to describe the jungle from various viewpoints, integrating competence thinking to the Nordic organisational research among others. Another paper that discusses the confusion of terms is Ash et al. (2000), which links that discussion into the areas of job analysis and competency modelling in that context. Coming to the latest publications, Mulder (2011) acknowledges the history of the term “competence” and states that “the meaning of the concept is mostly defined as being able to perform effectively” and provides a definition of competence to be:

- The set of integrated capabilities.
- …which consist of content-related clusters of knowledge, skills, and attitudes,
- …which are conditional for sustainable effective performance (including problem solving, realizing innovation, and creating transformation)
- …in a certain context, profession, organization, job, role and situation.

On a personal level of a tester, one critical question of competence is this: Does the Individual Matter in Software Testing? Merkel & Kanij (2010) tackle that very issue in a report of the same name. They made a survey to testing professionals and the results highlight domain knowledge as essential, which is usually linked with sufficient experience. Intelligence and dedication were considered important. Also, various other personality traits were mentioned. Communication skills and interpersonal skills were mentioned as important. Authors note that none of the skills are testing-specific, but rather relate to the generic good characteristics of an IT worker. It should be remembered here that the core occupational skills are so obvious that the most important skills do not get mentioned!

De Coi et al. (2007) present a general view to competence profiles as being divided into two classes: 1) Required Competence Profile which specifies the requirements (in terms of competences) to be fulfilled by an applicant and 2) Acquired Competence Profile which specifies the accomplishments (in terms of competences) of employees and learners and shows which competences have been acquired or to represent the expected accomplishment after a successful completion of a programme. This is the idea in competence gaps: the difference between what is available and what is required. The concept is even defined in standards. ISO standard on certification bodies (SFS-EN ISO/IEC 17024, 2012) defines competence as “ability to apply knowledge and skills to achieve intended results”. That is a very pragmatic definition.

Now, we need a typology of how the concept of competence relates to lower level concepts. Winterton, Delamare-Le Deist, & Stringfellow (2005) present one such typology in a report of European competence and qualification principles. It is shown in Figure 7.

	Occupational	Personal
Conceptual	Cognitive competence (knowledge)	Meta-competence (facilitating learning)
Operations	Functional competence (skills)	Social competence (attitudes and behaviours)

Figure 7. Unified knowledge-skills-competence typology matrix (redrawn from Winterton, Delamare-Le Deist, & Stringfellow, 2005).

This is an example typology that could form a template for a typology to be used in the testing and quality assurance context.

2.2.2 The European e-Competence Framework

There is a European e-competence framework, in 2015 at version 3.0 (the first version was published in 2008), that is designed to provide an orientation for companies and other organizations for managing competences for various occupations and ICT tasks. Its main documentation is in two parts, the Framework document (European e-Competence Framework, 2014) and its user guide (European e-Competence Framework, 2014a).

First, the framework's user guide (European e-Competence Framework, 2014a) defines the competence related concepts used:

- Competence is defined as "a demonstrated ability to apply knowledge, skills and attitudes for achieving observable results". Consequently, the related e-Competence descriptions embed and integrate knowledge, skills and attitudes.
- Skill is defined as "ability to carry out managerial or technical tasks". Managerial and technical skills are the components of competences and specify some core abilities which form a competence.
- Attitude means in this context the "cognitive and relational capacity" (e.g. analysis capacity, synthesis capacity, flexibility, pragmatism). If skills and knowledge are the components, attitudes are the glue, which keeps them together.
- Knowledge represents the "set of know-what" (e.g. programming languages, design tools) and can be described by operational descriptions as well.

The choice of wording is curious in the case of skills: how can tasks be divided into "managerial" and "technical"? That seems like a very harsh simplification. Linking capacity into “attitude” does not sound logical. Those definitions are listed here just for reference purposes, and to aid in our analysis or to be a target of it.

The framework is built on following structure:

- The main areas of activity, or competence area, related to ICT systems, which are plan, build, run, enable and manage.
- Under those are work profiles, or competences, which directly relate to people's roles. For example, the build activity has profiles such as: Application Development, Component Integration, Testing, Solution Deployment, Documentation Production, and Systems Engineering.
- There are five proficiency levels available and a set of those are associated for each task.

The proficiency levels are, starting from the highest level:

1. Associate: Able to apply knowledge and skills to solve straight forward problems; responsible for own actions; operating in a stable environment.
2. Professional: Operates with capability and independence in specified boundaries and may supervise others in this environment; conceptual and abstract model building using creative thinking; uses theoretical knowledge and practical skills to solve complex problems within a predictable and sometimes unpredictable context.
3. Senior Professional/Manager: Respected for innovative methods and use of initiative in specific technical or business areas; providing leadership and taking responsibility for team performances and development in unpredictable environments.
4. Lead Professional/Senior Manager: Extensive scope of responsibilities deploying specialised integration capability in complex environments; full responsibility for strategic development of staff working in unfamiliar and unpredictable situations.
5. Principal: Overall accountability and responsibility; recognised inside and outside the organisation for innovative solutions and for shaping the future using outstanding leading edge thinking and knowledge.

For testing and quality assurance, all levels except e-5 are applicable.

Let’s look into the descriptions for testing in the framework. First the overall descriptive text for the profile:

> "Constructs and executes systematic test procedures for ICT systems or customer usability requirements to establish compliance with design specifications. Ensures that new or revised components or systems perform to expectation. Ensures meeting of internal external, national and international standards, including health and safety usability performance, reliability or compatibility. Produces documents and reports to evidence certification requirements."

The proficiency levels are specified like this:

1. Performs simple tests in strict compliance with detailed instructions.
2. Organises test programmes and builds scripts to stress test potential vulnerabilities. Records and reports outcomes providing analysis of results.
3. Exploits specialist knowledge to supervise complex testing programmes. Ensures tests and results are documented to provide input to subsequent process owners such as designers, users or maintainers. Accountable for compliance with testing procedures including a documented audit trail.
4. Exploits wide ranging specialist knowledge to create a process for the entire testing activity, including the establishment of internal standard of practices. Provides expert guidance and advice to the testing team

The framework document gives "knowledge examples", which are not tied to the proficiency levels.

- Techniques, infrastructure and tools to be used in the testing process.
- The lifecycle of a testing process.
- The different sorts of tests (functional, integration, performance, usability, stress etc.).
- National and international standards defining quality criteria for testing.
- Web, cloud and mobile technologies and environmental requirements.

It also offers skill examples, again not tied to the proficiency levels or the skills:

- Create and manage a test plan.
- Manage and evaluate the test process
- Design tests of ICT systems
- Prepare and conduct tests of ICT systems
- Report and document tests and results

How about other quality related profiles? There are three ways for it:

- There are profiles for ITC Quality Strategy Development and ITC Quality Management.
- Some relevant quality management tasks are integrated into other tasks, for example the competence of producing quality plans is included in example skills for product / service planning, as they should be.

Actual work profiles for a domain can be formed by combining existing profiles fully or partially. The user guide notes:

> "A competence can be a component of a job role, but it cannot be used as a substitute for similarly named job titles, for example; the competence, D.7. ‘Sales Management’ does not represent the complete content of a ‘Sales Manager’ job

> role. Competences can be aggregated, as required, to represent the essential content of a job role or profile. On the other hand, one single competence may be assigned to a number of different job profiles."

The user guide also presents Quality Assurance Manager as an example. That profile is composed by defining it and then selecting e-competences from the ones in the framework on some defined level. In the example, they are: ICT Quality Strategy Development (levels 4 and 5), Risk Management (level 3), Process Improvement (level 3) and ICT Quality Management (level 4).

All in all, this framework provides a general taxonomy of task based competences, but does not add to our understanding about competences or testing. The framework documentation seems to have the concept of competences quite near the concept of a task profile and that masks the idea of the true competences needed in tasks.

Frameworks like this are mostly designed to give a common framework for nations and thus help co-operation across the EU. They may also pose a danger of being turned into formal qualifications systems. Indeed, the proficiency levels are mapped into the levels of the European Qualifications Framework (European Qualifications Framework, 2015). The mapping is found in the framework document.

All in all, frameworks like this necessarily look into things from above, as a global consensus and thus necessarily fail to even attempt to take into consideration any unique circumstances or emerging issues, which why they have in this kind of dissertation only an anecdotic the role. This framework does notice that. On the national level its user guide advices to first consider whether this framework would be of help for the local uses and then to consider the typical processes of local companies with the e-CF categories (plan, build, run, enable and manage) and to look into the areas the companies operate (the work profiles) and to consider the national, local, economic, social or cultural characteristics that would cause a need to modify the level descriptions. The framework’s site even has available a tool for building local profiles, so as long as the general idea of the framework is applicable for a purpose, the substance can be tailored, which is all good.

2.2.3 How this dissertation sees competence?

We have seen that there are many definitions for competence. Competence is clearly the most critical concept here and we need a clear specification for it that describes what is meant by in this dissertation. Is it skills, talent, knowledge or what? We use the following definition synthetized from the references and tailored to the context and purposes of the dissertation:

"Competence is the ability to be able to produce desired results in a given context".

Competence is about being able to do things that matter, things that are relevant and produce results. No other kinds of things matter. Part of competence is clearly the ability understand what matters and then acting on it. In product development, it means understanding what matters in the product and in the processes of creating and selling it so that the end result contains desired value and competitive edge.

In general, competence is an enabler. There are conditions in our environment and activities that demand or favour certain actions or a way of carrying out the actions or provide us with new opportunities. Competence is the thing that enables us to do the favourable things – of course we also need some resources for that too. The mechanisms in this are on a rough level visualised in Figure 8.

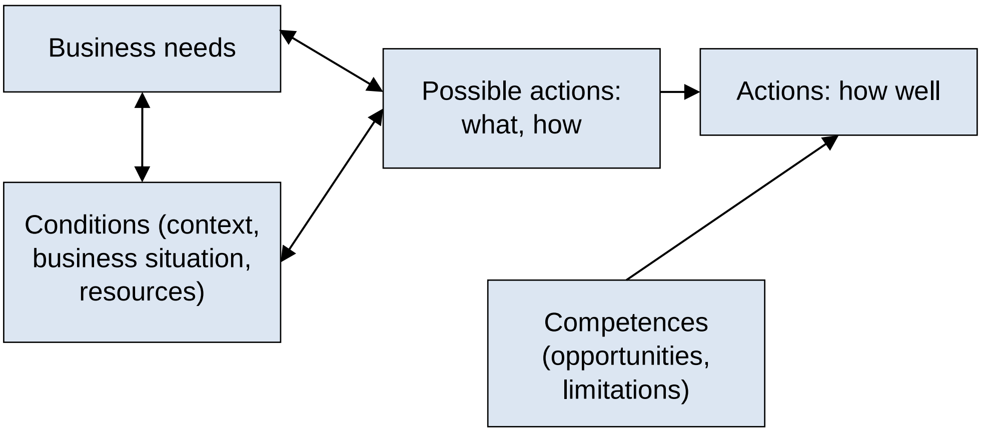

Business needs
Possible actions: what, how
Actions: how well
Conditions (context, business situation, resources)
Competences (opportunities, limitations)

Figure 8. Competence as an enabler.

The business needs give activity goals and scope. The organisation knows what should and could be done to fulfil the needs. The organisation lives and operates in a context – in a business environment, in some economic situation, with certain assets and so on. Those frame the action that could be taken. Only when the competences at disposal are considered, the company can see what actions could actually be taken, can estimate their success and related risks of the operations. So, the competences are the “final enabler” for doing things successfully.

Competence is used on the personal level. On the organisational level the same thing is often called "capability". The factors that influence competence include:

- Knowledge.
- Skills, including cognitive skills.
- Understanding about the task and the desired outcome, requirements and its quality-related aspects.
- Understanding about the context.
- The tools the person has in his/her disposal.

The context is understood to be dynamic and having elements of various scopes: working environment, project, technology, goals, tasks and the culture – among others.

2.3 What is quality?

When we talk about testing and quality assurance, we need to first assess the question, what is “quality”, because that very much defines what kind of information we are looking to produce by testing and other activities – and it is an area where many have rather mechanistic views. Let’s look into quality at various angles.

The traditional view to the quality of a software system is captured in the software system characteristics list of ISO/IEC standard 25010 (2011) and condensed in Table 1.

Table 1. Quality characteristics of ISO/IEC 25010 (ISO/IEC, 2010)

Characteristic	Sub-characteristics
Functional suitability – characteristics about the set of functions and their specified properties that satisfy stated or implied needs.	Functional completeness Functional correctness Functional appropriateness
Performance efficiency – characteristics about the relationship between the level of performance of the software and the amount of resources used.	Time behaviour Resource utilization Capacity
Compatibility – the degree to which two or more systems or components can exchange information and/or perform their required functions while sharing the same hardware or software environment.	Co-existence Interoperability
Usability – characteristics about the effort needed for use, and on the individual assessment of such use, by users.	Appropriateness recognisability Learnability Operability User error protection User interface aesthetics Accessibility
Reliability – characteristics about the capability of software to maintain its level of performance for a period of time.	Maturity Availability Fault tolerance Recoverability

Characteristic	Sub-characteristics
Security – The degree of protection of information and data so that unauthorized persons or systems cannot read or modify them and authorized persons or systems are not denied access to them.	Confidentiality Integrity Non-repudiation Accountability Authenticity
Maintainability – characteristics about the effort needed to make specified modifications.	Modularity Reusability Analysability Modifiability Testability
Portability – characteristics about the ability of software to be transferred from one environment to another.	Adaptability Installability Replaceability

The standard's list is aimed to be used only as a baseline for creating any particular list of quality characteristics (or a "quality model"). So, the characteristics are just a baseline, but yet, those are the practical things that are tested and otherwise assured. This is a rather engineering style of classification that may not include the broader characteristics of products. Other standards may have more abstract views. ISO/IEC/IEEE 24765 Systems and software engineering – Vocabulary (2010) lists alternative meaning for quality collected from various sources:

1. The degree to which a system, component, or process meets specified requirements. IEEE Std 829-2008 IEEE Standard for Software and System Test Documentation.3.1.25.

2. Ability of a product, service, system, component, or process to meet customer or user needs, expectations, or requirements.

3. The totality of characteristics of an entity that bear on its ability to satisfy stated and implied needs. ISO/IEC 9126-1:2001, Software engineering — Product quality — Part 1: Quality model.B.21.

4. Conformity to user expectations, conformity to user requirements, customer satisfaction, reliability, and level of defects present. ISO/IEC 20926:2003, Software engineering – IFPUG 4.1 Unadjusted functional size measurement method – Counting practices manual

5. The degree to which a set of inherent characteristics fulfils requirements. A Guide to the Project Management Body of Knowledge (PMBOK® Guide) – Fourth Edition.

6. The degree to which a system, component, or process meets customer or user needs or expectations. IEEE Std 829-2008 IEEE Standard for Software and System Test Documentation.3.1.25.

As there are indeed many definitions, we clearly need to also check what the testing experts' community thing is the most appropriate. The testing glossary of International Software Testing Qualification Board, ISTQB (van Veenendaal, 2010) has selected this definition: "The degree to which a component, system or process meets specified requirements and/or user/customer needs and expectations. [After IEEE 610]". That is still a quite generic and common definition from a generic IEEE terminology standard (IEEE, 1990).

There are more interesting definitions. Gerald Weinberg is a highly respected expert in the testing world. He has a very contextual definition of quality (Weinberg, 1992): "Quality is value to some person". Keyword is here "some", as he also states: Every statement of quality is a statement about some person(s) and more quality to some person may mean less quality for another, so we need to think about whose opinion of quality is to count when making decisions. This is something that many companies have needed to tackle in their practical decision making in product business – which person or role is "right" about quality in any given situation. Things really are not that simple, when we have a non-trivial understanding about quality. This philosophy means that anyone can define quality and will be right about it. So we need to respect the opinions of those who matter. Ultimately, in the product business, they are mostly the customers (also in the role of the user), but that doesn't mean that other people's opinions would not matter. The customer is not always right about what is important for the company running the business. Other people will understand that, including management, quality people, developers etc. They may prioritise the customer and user-centred qualities, but also consider the differentiation on the market and technical characteristics that help in maintaining and evolving a product or a product portfolio.

Sometimes quality is mystified. The book "Zen and the art of motorcycle maintenance" (Pirsig, 1999 – first issued 1974) is often referenced when writing about quality. It offers among others this snippet:

> *"Quality—you know what it is, yet you don't know what it is. But that's self-contradictory. But some things are better than others, that is, they have more quality. But when you try to say what the quality is, apart from the things that have it, it all goes poof! There's nothing to talk about. But if you can't say what Quality is, how do you know what it is, or how do you know that it even exists? If no one knows what it is, then for all practical purposes it doesn't exist at all. But for all practical purposes it really does exist. What else are the grades based on? Why else would people pay fortunes for some things and throw others in the trash pile? Obviously some things are better than others—but what's the*

> *"betterness"? – So round and round you go, spinning mental wheels and nowhere finding anyplace to get traction. What the hell is Quality? What is it?"*

From today's perspective we understand that quality is not just technical or ergonomic, but depends on the values and culture where the user lives and works in. The current term that encapsulates many of the qualities at this level is "user experience". It highlights some of the characteristics of quality: we do not need to be able to measure it accurately or even understand its details. Sometimes it is enough that someone can say which alternative is better! This principle is today used in many tasks in business level testing of product concepts and implementations (such as A/B testing).

Those are qualities of a software system. What about the other views to the product? If we assume for a moment that some of the most critical products will be disruptive products that form at one stroke new product markets, what are the qualities of disruption? We could at this point formulate a list of potential qualities such as:

- The new concept / approach is immediately obvious.
- The new concept shows obvious benefits (perhaps in comparison with the old concept, but not necessarily).
- It hits hard a population that finds it desirable, making it possible to start the business.
- It is significantly better than the alternatives – being just a little better is not sufficient.
- It has sufficient technical quality (reliability etc.) to get business started.

These are elements of "quality for business", which is present at the level of product concepts, but also at product implementations. For example, at the user interface level, just usability is not sufficient for commercial web sites, where designs must lead the user to make purchases.

We need to remember that quality is not just the quality of the software component. The component may form just the "core product", but it is the quality of the overall product that matters. That is, the quality of everything that the customers and users deal with. That includes the product information, delivery mechanisms, the process of purchasing, the quality of any customer services and technical services, etc., during the whole lifespan of the product.

One important characteristic of quality is that it is not absolute, but relative. Quality is perceived in reference to expectations gained earlier (or manufacturer's promises) and in comparison to alternatives, such as competitor's products that the customers on the market have experiences. In that sense, quality is not even a thing as such, it is not an attribute of a product, but a relation between the product and the customer or user. That has been understood and emphasised in the usability context for long. Because of that, there is not any meaningful concept of a product without association with it the idea of who the users are, what are their ways of using the product, what are their preferences and expectations, and so on. The author has called this kind of product concept structure a "operational product concept" already in the 1990s while doing research on usability and future product development. That is in today's terminology a form of system thinking, where the single elements of a system are meaningless and undefined without the whole context.[5] The elements of the operational product concept are shown in Figure 9

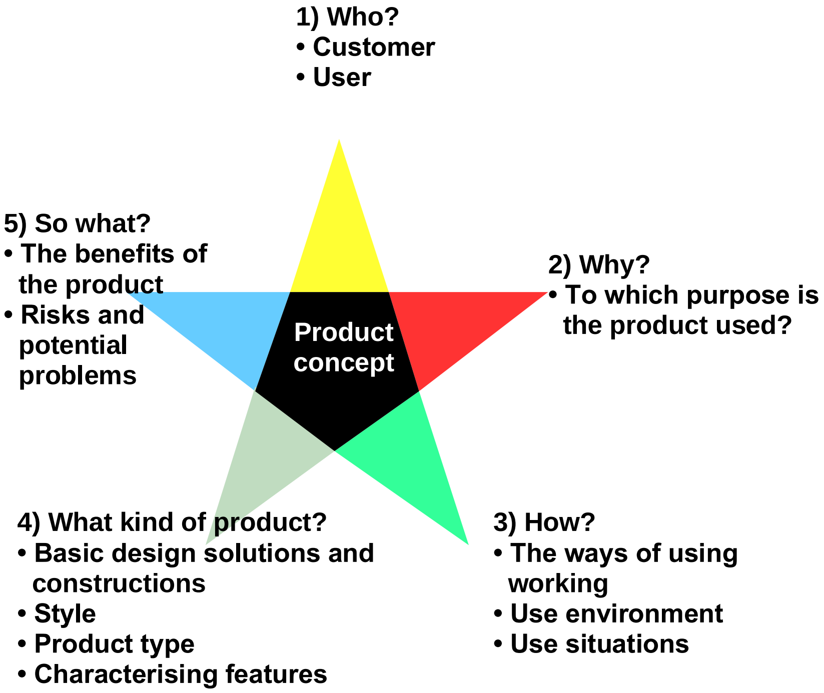

1) Who?
• Customer
• User
2) Why?
• To which purpose is the product used?
3) How?
• The ways of using working
• Use environment
• Use situations
4) What kind of product?
• Basic design solutions and constructions
• Style
• Product type
• Characterising features
5) So what?
• The benefits of the product
• Risks and potential problems
Product concept

Figure 9. Elements of the "operational product concept" (Vuori, Kivistö-Rahnasto & Toivonen 2001).

This thinking is very much in contrast with some software development cultures that still have a technical product paradigm where the product concept covers only the product itself, although those cultures acknowledge prioritisation of features based on customer needs and designing based on user stories, but the link is still vague.

To summarise the discussion about the quality factors, Table 2 presents the layers of a product and some of quality factors that visualise the quality at the layers.

[5] In fact, we shall later see contextual models for human activity that would be quite fitting for understanding about products, but introducing those now would be too soon and too distracting to the reader.

Table 2. Product layers and examples of quality factors.

Layer	Customer and user viewpoint	Manufacturer viewpoint
Product concept	Match with needs, values and desires	Clarity Desirability for target demographics Fit to brand
Overall product	Purchasibility Customer satisfaction Lifecycle costs Manageability	Manageability Support costs Compatibility and support for growth of ecosystem
Functional product	Functionality Usability User experience Efficiency of business processes Security	Market distinction Compared with competition Developability Meeting of standards Lifespan expectancy
Technical product (software system)	Reliability Compatibility	Developability Maintainability

The main lesson here is that the definition of quality depends on the context, the viewpoints chosen. One critical competence is clearly the ability of understanding the context and forming an understanding based on priorities. Because of this, the dissertation will be open in the definition.

When is quality sufficient? When have we found and corrected a sufficient number or problems so that the system is satisfactory? When does more testing and error correction provide benefits?

When we think of the relativeness of quality, the question depends on the viewpoints of all parties of product development and manufacturing. There are many issues that affect the question, like user satisfaction, safety and business risks to the customer, or risks for the manufacturer from recalls and product updates (which can be very costly even when delivered via the Internet). This is an area that includes so many issues of understanding businesses and users that we cannot afford tackling it in this review, but will assess some of these questions later. For now we just note that understanding that is a very important area of competences.

2.4 What is “testing”?

There are many meanings for testing, but if we wish to have an understanding about it, which is not constrained by today’s conventions, we need to accept that there are many views to that, each one giving us some important message. IEEE Std 829-2008 IEEE Standard for Software and System Test Documentation (IEEE, 2008) is the most widely used testing standard and it defines testing as: “[an] activity in which a system or component is executed under specified conditions, the results are observed or recorded, and an evaluation is made of some aspect of the system or component. The ISTQB tester certification system foundation level syllabus (ISTQB, 2011a) describes it being besides the activity of running a test, a range of activities before and after that, including planning and control, reporting and reviewing of plans. The reviewing of plans is a form of “static testing” and as it has no empirism, it is hard to call “testing”. Consider a product which has never been executed or even simulated, but the requirements of which have been reviewed. It would be misleading to say that it has been tested.

On a practical level, we could say that testing is something that is done empirically on a product to gain some quality related information for some purpose. This would obviously include executing the system with some defined test cases to see how it behaves and whether there are any errors, or having test users use a product and observing them how they behave and if they have any problems with the product, or delivering alternative versions of a web store to customers and measuring their use and making comparisons. The product versions used may be production versions, versions under development, prototypes, demonstrators and even paper prototypes. On the smallest scale, product icons can be tested for preference by users. Some common examples of testing at various stages of a system’s lifespan:

- During software development, the software system is tested to find out about its quality (correct functioning, usability, security, performance).
- Software is tested as part of the computing system, in test environment, consisting of computers, operating systems, communications systems etc.
- We may test the whole system or just an element of it, such as a unit, module or a class or some minor detail in the user interface.
- The software may be under various maturity levels when tested. It may be a prototype, with which we may try to find out the requirements for the actual system, or a version under development, or a fully implemented system that we wish to accept for use in an organisation, or to the market, by the information gained by testing.
- A system that has already been taken into production use may periodically be tested for changes in its performance or for diagnostic purposes.

- In a production version of a commercial web system, alternative user interface implementations may be tested and measured how they work for the user population.
- The system that we test may be an embedded system where software is just one element that provides the functionality for the user, or a system of systems where many systems interact.
- A production version may be comparison-tested against a new product version from a competitor.

Even though we think that the technical artefacts are tested, it is actually the overall business process that is often being tested, or the relation between user and the technical system. As in any activity, the core actions – the execution of some sorts – is surrounded by other activities, which actually consume most of the time. They include:

- Planning the overall testing activity.
- Arranging the test environments and tools.
- Designing the details of testing.
- Discussions and making contracts with other parties about the arrangements, timing and goals of testing.
- Assessment of the results.
- Reporting the new information to others and discussing about it.
- Managing assets, such as documents, test scripts and test data.

Like any human activity, testing is not only rational activity, but it has deep cultural meanings for a person and for an organisation. Schein (2004) presents the levels of organisational culture being (1) visible artefacts, (2) espoused beliefs, values, rules, and behavioural norms, and (3) tacit, taken-for-granted, basic underlying assumptions. The last ones are unconscious, yet the ultimate source of values and action.

On the first level, the presence of testing is a visible artefact that is an icon for quality and for the aims of reaching quality. As such, it represents and leads the quality culture in a company irrespective of how well it is done. On the second level are the actual behavioural patterns of activities related to testing and how they have guiding norms and instructions and shared expectations for the activities and their results. On the third level are the underlying assumptions of for example how software is built, how managed of chaotic the system is that produces the software and what is the nature of technology – and leading from those, the assumptions for the need and role of testing.

Note that the above is an organisational culture view. The other cultural viewpoint would be “quality culture”, which generally refers to “how well” quality issues are handled in the organisation and how much everyone participates in the quest for quality. How is quality valued and lead by top management? Are there practices in place, including testing, and how disciplined the actions are? Dellana & Hauser (1999) gives an example of statistically defining elements of quality culture. That view is usually closely related to the idea of maturity or an organisation, and an important one, but really a different viewpoint to the one that identifies elements testing activity and thinking as elements of organisational culture.

So, testing can be a very rich whole.

Besides what testing is, we need to understand what its purpose and goals are, as that is a requirement for finding new ways to do that. ISTQB (2011a) describes the purpose to be finding defects, gaining confidence about the level of quality, and providing information for decision-making, and preventing defects. This definition of the purpose does not take into account the context where testing is carried out. We will get into that later and offer extensions to the traditional definition of the purpose of testing.

2.5 Who does testing, who is the tester?

“Anyone, who does testing, does testing!” By this we mean that testing is an activity, a task and being a tester is a role that may be temporary or more static – someone “puts on a tester’s hat”.

In the early days of software engineering, the programmers were testers too. One person did all development tasks. In the case of embedded systems, there were obviously dedicated test engineers that tested the software as part of the overall system – just like they do nowadays. When software development evolved, it became clearer what kind of activities were needed in projects, especially large ones. Gradually was a perceived need to have a tester occupation emerged. That idea was supported by the growth of formal knowledge about testing and a growing process approach to software development. Still, low level testing was, and still is, the programmers’ task. So with “tester” we mean a person in a role where she, by that role, gives a value proposition to others that she is willing and available to do testing and that she aims to do testing properly. In that role, she can do other things as well, and can also have other roles in the development organisation or in the organisation that acquires software systems. In practice, testers are for example:

- Assigned testers in development teams.
- Testers in testing teams.
- Consulting testers who are used as needed, for example performance, security or usability testing experts.
- Software developers doing unit testing or integration testing.
- Software developers, when they otherwise have a “tester’s hat” on.
- Business users when they do testing.

- Everyone involved in the software project – including managers – when they have a testing task or participate in a “test bash”.
- User experience developers when they plan and execute A/B testing, where different versions (“A and B”) are produced and delivered to different user populations and their usage metrics compared.
- Independent testers doing validation of a safety-critical system.
- End users participating as a tester role in the development of an open source system or following the instructions of a beta test process.

But end users, when they just use a system; they are not testers.

2.6 Role of testing for an individual

We see that many people do testing, but the role of testing for an individual varies greatly. There are the “testers” (by any job title) whose identity is based on testing and who have a main role in the activities that primarily promises added value from the testing they do. They think and act like a testers and others expect that. That does not mean that would not do other things, including programming, as needed. But then there are people who just use testing as a tool when needed. Developers do a variety of tasks using various tools. They do programming using editors, compilers and version control tools and similarly they – in varying amount – use testing and testing tools as needed. Testing is in that case just one element in an integrated whole of a professional’s life. But a developer’s identity is a developer’s identity and her approach to the software is different than a tester’s. On a more abstract level, testing is a reflection base for one’s work – somethings that shows a mirror image of all tasks and tells about the integrity of the work. For some people testing is just a mandatory element in the software development or acquisition process. This may be a result of certain mindset (strong design-thinking) or a symptom on under-developed quality attitude. Managers, product planners and owners, and decision makers in general do not in that role do testing. They use information created by testing as an aid in understanding the products and work around them. For them, testing is a source of information – we could even call it a media! They may also see testing as an organisational process, not as an individual’s activity.

2.7 Testing is done in contexts

Testing and quality assurance are always done in some context. A context is understood in the context of this thesis as:

- An action system with unique elements.

- It has unique principles and rules for activity.
- It has various states and situations, in which the interactions between acting elements differ.
- It may change to another context by some transformation caused by for example it having reached its temporary goals.
- A true system where every element needs to be present in order for it to work (whether the elements are explicitly defined or not).

When we in this thesis discuss the changes in our environment, we do that by assessing changes in some context. There are various levels of context that surround each other in an onion-like way. See Figure 10.

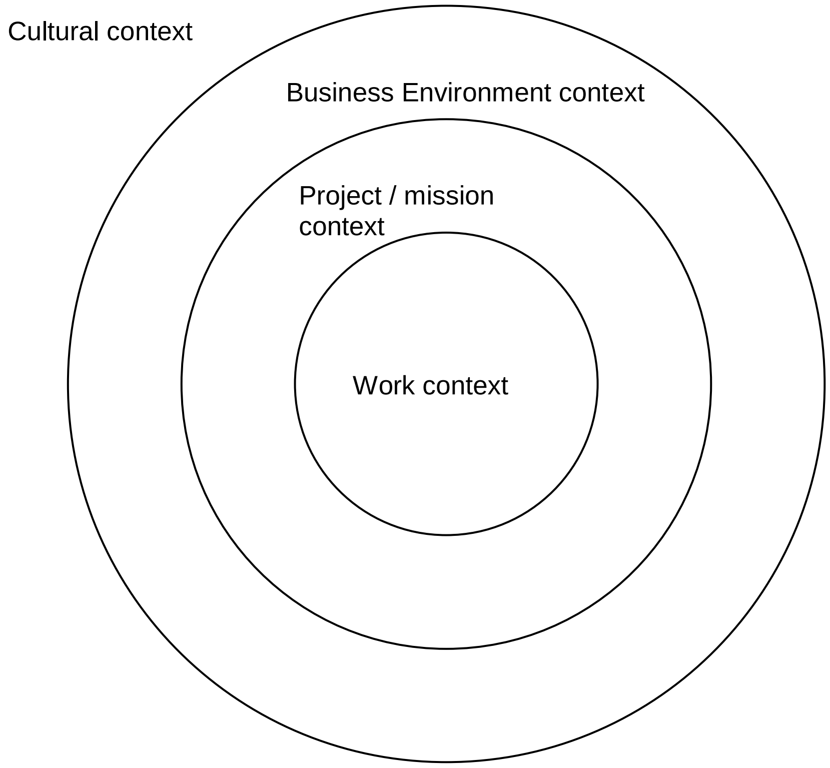


Figure 10. Different levels of contexts

The lowest (in the figure, the central level) is the context or work. That is where the tester does testing work on a product and its technology, or in general, some artefact, using some methods and tools, having some goals, etc. The next level is the project context. It has its own defining elements, such as timelines, project goals, participants, stakeholders and instructions. Projects are executed in some business environment, which may be a company, a government office, a community or something else. There are elements such as the company's organisation, the company culture and the domain it operates in, that need to be accounted for. Above that is the external context of the national culture, economy, politics, the general practices of working in occupations, ethics and so on.

Some of the contexts we can create – the “lower” level contexts are defined for example with software developers, when teams define how they work, when and how testing is done. Higher level contexts are largely given and can seldom be affected – testers or companies cannot influence how national culture is or how it develops.

Between those are the changing contexts that other actors of society can influence, for example the economy and the environments for companies and the educational system. The government and other such actors take care of those, hopefully successfully. Now, all these levels interact and overlap and in our rich world each one of those requires many models to describe sufficiently. That is why we don’t attempt to describe the elements of the contexts here any further, but leave that to later chapters.

Much of the research related to testing and in general, software development competences, is positioned at the level of engineering, software lifecycle processes, deployment engineering and similar. That fails to acknowledge the critical levels of real product development or that everything people do is done in an organisational context and besides having a varying element of engineering, is about humans interacting. That’s why research needs to span the contextual layers and be multi-disciplinary. Those readers coming from the software engineering culture may need to refocus their thoughts somewhat.[6]

2.8 Contexts are systems

All contexts are systems by their nature. According to Ackoff (1999) a system "is a whole consisting of two or more parts that satisfies the following five principles", which are:

1. The whole has one or more defining characteristics.
2. Each part in the set can affect the behaviour or properties of the whole.
3. There is a subset of parts that is sufficient in one or more environments for carrying out the defining function of the whole; each part is necessary but insufficient for carrying out this defining function.
4. The way that each essential part of a system affects its behaviour or properties depends on (the behaviour or properties of) at least one essential part of the system.

[6] It seems that even some s/w engineering research cultures that do constructive research have a hard time understanding that tools are used by humans and processing logic or logistics are not sufficient based for good tool design.

5. The effects of any subset of essential parts on the system as whole depends on the behaviour of at least one other such subset.

For example, in companies, testing practices form one subset in product development, but alone cannot produce anything meaningful. All the parts of organisational activity are needed, and they cannot be separated, and thus the whole of the organisational activity needs to be under scrutiny. Note that the approach here is broader than for example the common discussions about feedback loops and similar. Here it is the matter of all kinds of relations and interactions in operational, psychological and even symbolic level. Besides the implications for analysis, this also means that the optimal practices in any context depend on the other elements of the system. It is not possible to form an optimal testing practice without consideration for e.g. the management system, business idea, product technologies, organisational culture and so on.[7]

In software engineering, architectures are systems and it is understood that they need to be looked at from various viewpoints in order to understand them. Kruchten's (1995) is the most well-known manifestation of that and presents these views:

- Logical view, which concerned with the functionality that the system provides to end-users.
- Development view or implementation view, which illustrates a system from a programmer's perspective and is concerned with software management.
- Process view, which addresses the dynamic aspects of the system, explains the system processes and how they communicate, and focuses on the runtime behaviour of the system.
- Physical view, which depicts the system from a system engineer's point of view.
- Scenarios, which illustrate the architecture with a set of use cases, or scenarios.

Similarly, any context of human activity can and should be viewed from many perspectives. Traditionally, in software development, process flow views have been important in project activities (Gantt charts, flowcharts and others are used as presentation format). Those are close in nature to views of information flows. Organisation charts provide a structural view to an organisational context and in the case of distributed activity also describe the physical implementation. Those are just some that are used.

It is partly semantics whether the views are just views or models, in which case they represent a "real" alternative reality. That viewpoint is relevant, because different views can include very different system elements.

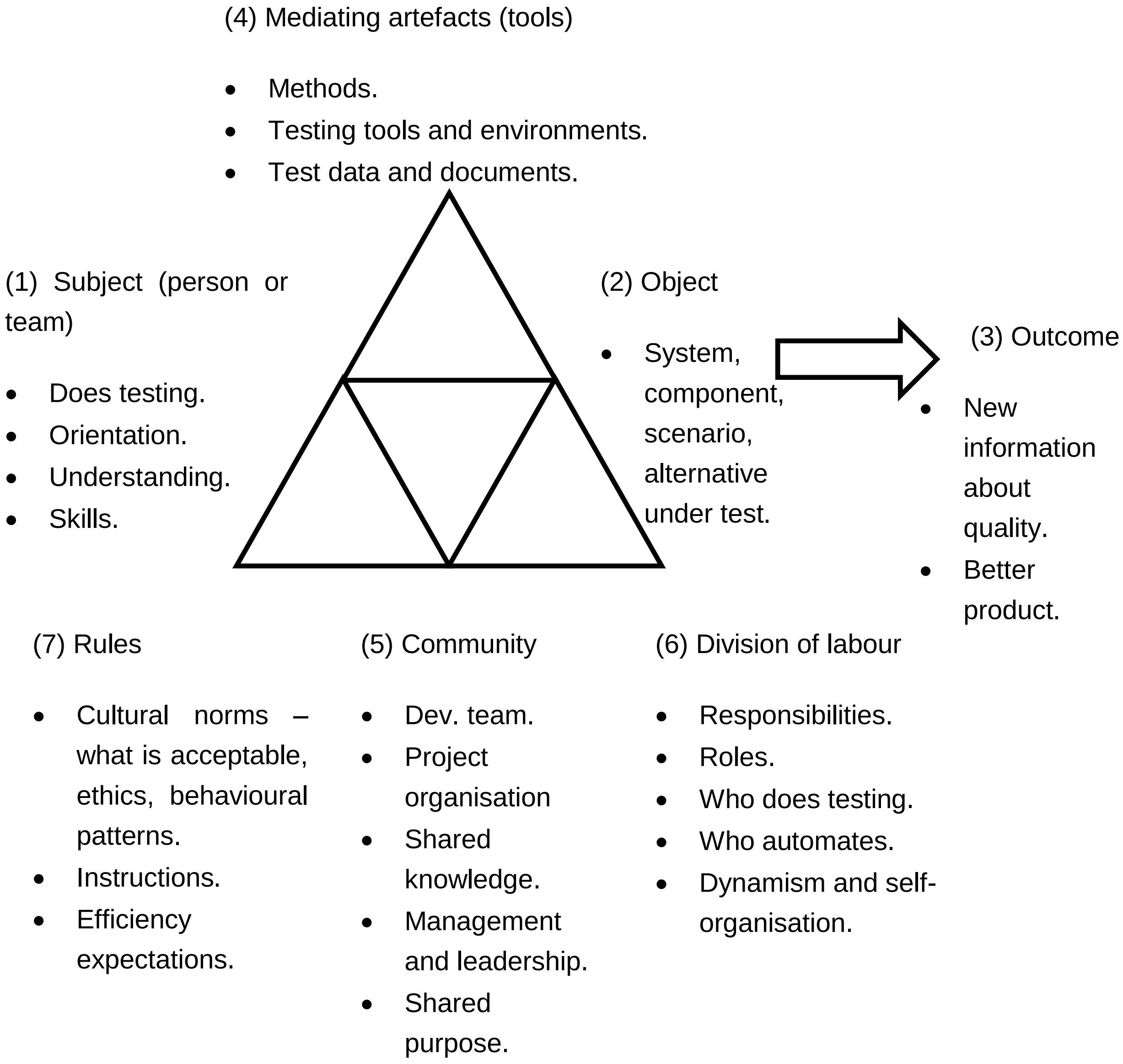
[7] This view also suspects that using any testing process standard as reflection for development of testing practices is suspect, while done e.g. by Kasurinen (2011).

The important thing here is that this dissertation uses different models / views during the analyses and in making the conclusions, in the hope of that way revealing the nature of the reality.

It needs to be noted that process models and similar show a current snapshot of what the organisation does, they are a creation of the organisation. It is more valuable to address the characteristics of the organisation that does the creation, as they are the ones that are more stable, respond to changes and create changes. The activity theory provides us one such model and that is the one that we'll look into next.

2.9 Activity system as contextual model

One view to the work context is to see it as an activity system. The activity theory is according to Barap et al (2004) a "psychological and multidisciplinary theory with a naturalistic emphasis that offers a framework for describing activity and provides a set of perspectives on practice that interlink individual and social levels". The most important element of the theory is the model of the activity system that presents the relations between the actor (e.g. the tester) and the objectives of the work and the working organisation. Action research has for some decades used the activity triangle of Figure 11 to model work systems.



(4) Mediating artefacts (tools)

- Methods.
- Testing tools and environments.
- Test data and documents.

(1) Subject (person or team)

- Does testing.
- Orientation.
- Understanding.
- Skills.

(2) Object

- System, component, scenario, alternative under test.

(3) Outcome

- New information about quality.
- Better product.

(7) Rules

- Cultural norms – what is acceptable, ethics, behavioural patterns.
- Instructions.
- Efficiency expectations.

(5) Community

- Dev. team.
- Project organisation
- Shared knowledge.
- Management and leadership.
- Shared purpose.

(6) Division of labour

- Responsibilities.
- Roles.
- Who does testing.
- Who automates.
- Dynamism and self-organisation.

Figure 11. The activity system triangle, adapted from Engeström (1999) with example content for testing.

The idea is briefly as follows. The (1) subject is the person who does the work, in this case the tester (of any occupation). She is a person with her own orientation to the work, her understanding and practical skills to do the expected things. The skills obviously need to be suited to the purpose – the system under test, the methods to be used and the workings of the work community.

She does the work on some (2) object, such as a system or a component under test. Sometimes the object is a user scenario or an alternative design or implementation of a product or its element. Again, it is understood that different objects (including their state) require different approaches. The object encompasses also the motives and meanings of the activity,

The work results in (3) outcomes, which often are information about quality or a “tested software version which may have become improved with the help of the new information.

She uses some (4) mediating artefacts, tools or (abstracted) instruments in that work. They can be testing tools, test systems, methods, data and documents. It was already noted that those need to fit the other elements of the task at hand.

The work is carried out in a (5) working community – a team, an organisation that shared its purpose, has a management system, elements of leadership and general culture. That provides the general approach for working, for creating and assessing quality and everything else must fit those basic premises.

In that community there is a (6) division of labour – a professional tester (if there is one) does some things in testing, the developers do other things and managers still something else. Expectations on that are based on their jobs and roles. The roles can be dynamic.

All this is guided by (7) rules, which include non-written organisational norms, process instructions and cultural conventions. Rules are created by the working community and may be based on analysis and decisions, or in the case of cultural norms, on the shared history.

Systems (and their expressions) are often expected to have some defined interaction types between their elements, but when it comes to an organisational system like this, they really are varied and what matters can be identified in the analysis of any particular context.

In an activity system there needs to be a balance between these elements and if one of those is changed – by process improvement or some other reason – the others need also be assessed to regain this balance. But yet, local optimisations are something to avoid. That is emphasised the by each node of the model having several connections. These are the very ideas in systems in general. This model has been found to work well in practice in showing some essential elements and the author has used it in many cases, for example in the 1990’s in analysing industrial assembly workplace designing (for example Vuori, 1994) and in analysing product development (Vuori, 1998).

We can already identify some essential competence-related issues with the model: The tester has individual competence, but the community obviously has “collective competence” and the competencies of each individual. One essential competence is to apply competencies in team work so that the overall competence is maximised. In order to reach the desired objectives, the tester must understand those. That understanding is required for selecting the most suitable tools to use. When working in any organisation, the rules must be known and adhered to in order to make the total operations as effective as possible. This means that the tester must know how to work in some context and know the applicable standards and regulation – and others' expectations for her work. One important element is also having shared values with the rest of the organisation. It should be noted that a team can also be considered a subject in this model, and when assessing teamwork in agile development, for example, the team may be the most important subject.

The main value of such model of a work context is that it presents all the elements and forces that influence work. It reminds us that we must not be naïve and expect just better tools to change things into better, but at the same time we must change other related things too.

We will use this triangle a couple of times as a tool of our analysis and explain it more in later chapters.

In practice, the importance of context is shown in these examples:

- Earlier it was noted that the quality factors vary at different layers of the product and so do the goals and practices and the culture of collaboration.
- Games are developed differently than safety-critical systems.
- Small startups have different needs and practices than mature, larger companies.

We will look into these later as needed.

2.10 Anatomy of change and how the contextual layers interact

As we noted, one view to our world is that it is composed of layers:

- The global world,
- Nation – Finland.
- Organisations.
- People.

All of those are in interaction. Upper layers have various kinds of influences and limitations to the lower levels and the lower levels in turn form the upper levels – such as the economic functions of companies and people forming up the thing we call "Finland". In fact, all global phenomena start somewhere in a local context and then spread. See Figure 12 that visualises this in conceptual form.

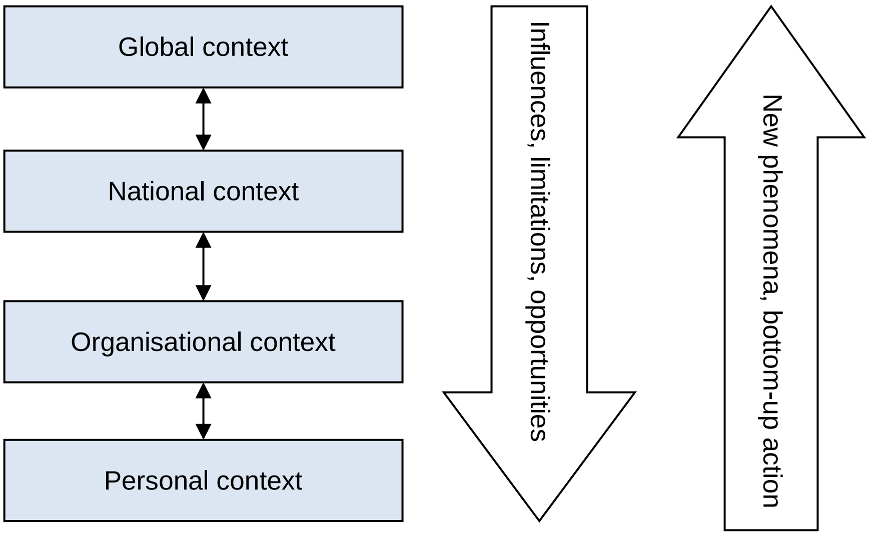

Global context
National context
Organisational context
Personal context
Influences, limitations, opportunities
New phenomena, bottom-up action

Figure 12. Layers of context are in interaction.

For example, the global level interacts with national level in many ways. There may be changes in focus due to global megatrends and paradigms. Sometimes they are so strong that we may think of our world view changing and us getting a new set of assumptions about things. Examples of this have been multiculturalism, ecological thinking and the wide introduction of agile thinking. These bring with them a changing culture. Similarly, due to global changes we may get new priorities. Security was not a priority some years ago in many domains, but now it is a critical system characteristic everywhere.

Along with the changes in thinking we gain new working styles and processes. Where projects were traditionally carried out with generic project management practices, software and product development implements the agile thinking in processes that are tailored to that work.

The global market sees changes all the time. During every period there are countries that lead innovation and are a model for others and there are countries that are profiles as providers of low cost services. In our context, those services include software implementation and testing services. The location of such countries “travel” around the globe. So the roles of countries and their cultures change and the same happens to Finland every now and then. Global political climate also affects the collaboration between countries. Sometimes relations between countries may become just a bit tense and the market will be careful to buy communications and hosting services or product development services from certain countries.

The population’s role changes. Once the citizens may have been thought only as passive consumers, but due to the community cultures in for example open source software development and open innovation, they have become active parties in the development of systems.

The volumes change. In the days when software applications were sold on physical media in stores, they were and rare. Now that software is sold and distributed online in the internet and application stores, for example games may sell in millions.

The above is related to change in costs. Games that would sell for 50 euro – not uncommon previously – would not sell such volumes. If they cost 2 euro, they can sell a lot – but most don’t sell but a few. In many domains, customers are getting used to software costing nothing. Open source software is simply downloaded and used. This means that some companies need to think of other ways of making money. Tailored development is still needed and will be needed in the future; integration services and hosting services will be needed always. However, it will be more difficult to sell (conceptually) “shrink wrapped” software.

International laws and standards may change. There may be international agreements about trade, new safety standards and similar.

Basic technological development is global in nature. Things like Internet, cloud computing, mobile technologies are global in nature once they got in wide use so that they formed paradigms and viable environments.

When companies work globally, they see and meet global changes everywhere. The changes also diffuse into the local domain. To generalise the global changes, they may bring us various type changes in our environment:

- New culture.
- New ways of working.
- New technology.
- New opportunities.
- New limitations.
- Change of goals and focus.

In response to the upper level changes, there must be changes in the lower level context. The change may be organic and unconscious, but there must be conscious changes with defined goals too, because we are all so attached to the old ways of things. There is great inertia. Inertia is good, because it keeps us going when there are small ripples around us – like a boat or an automobile’s engine would soon stop without inertia. When things change permanently, inertia slows us down. We may not be able to make a turn when the road makes it. Because of that we need to look into the changes find out what reactions are needed on any level and on any element of our activity system. On general level the responses may include:

- New awareness.
- Organic changes.
- Cultural changes and new set of assumptions.
- New focus areas for businesses.
- Rethinking of practices.
- New strategies.
- Building of new competences.
- New national strategies.

Of course, some changes are more relevant than others. Some are passing ripples, some are just manifestations of a bit deeper level of change and some are about the very basis of how we as humans behave and understand our behaviours. The daily media presents us the ripples and passing manifestations of something deeper. For example, we may read that a company has had a hackathon, but that is not very interesting as such. But it is interesting in the sense that the hackathon is an icon of something that is happening in the ways the companies do their product development and technology management. That level is the interesting one: what kind of changes are seen in that? More focus on experimentation? Quest for more speed? More flexible ways of organising activities? Relying on small companies instead of a huge R&D centre?

All this is not that simple, because the world is not built like a simple mechanism. Instead, it consists partly of good old known things, partly things that are somewhat familiar, but not yet that well known and things that may at least look like they are a complete chaos, which brings us to the topic of the next chapter.

2.11 Making sense of changes – chaos, complexity and Cynefin

Today it seems common to say that things are chaotic, but different domains (or contexts) and situations vary greatly in this regard. Similarly, any new phenomena will look like a chaos and too complicated to understand. It would be good to have a framework for this that would allow positioning things on some dimensions to make them easier to assess. Cynefin[8] (Kurtz & Snowden, 2003) is a framework designed just for that. The framework divides our world into five different domains categories, see

[8] A Welsh word, pronounced /ˈkʌnɨvɪn/.

Figure 13. Each new situation, phenomena, activity, practice or technology can be positioned into one of those.

Complex

- Cause and effect are only coherent in retrospect and do not repeat
- Pattern management
- Perspective filters
- Complex adaptive systems
- Probe-Sense-Respond

Complicated (Knowable)

- Cause and effect separated over time and space
- Analytical/Reductionist
- Scenario planning
- Systems thinking
- Sense-Analyse-Respond

Disorder

- Destructive state where the domain is not known

Chaos

- No cause and effect relationships perceivable
- Stability-focused intervention
- Enactment tools
- Crisis management
- Act-Sense-Respond

Simple (Known)

- Cause and effect relations repeatable, perceivable and predictable
- Legitimate best practice
- Standard operating procedures
- Process re-engineering
- Sense-Categorize-Respond

Figure 13. The domain types in Cynefin (based on Kurtz & Snowden, 2003). The labels are as they are called by Snowden and other later and in parenthesis the original names, which are more meaningful in the knowledge management context where the model was first developed.

On the right side are the “ordered” domains: The simple domain is the traditional worldview where everything is predictable. Engineering should be like that, and engineering type testing too, at least in principle. It is assumed that there are best practices for any task and that we should use them. This is the world of bureaucracy. Simplistic science is also often like this.

The complicated domain is something that we do not understand, but could learn to once we analyse and study it more. Customers’ and users’ world is to us often like that. We think that there is some order and sense, but we just need to clarify it by drafting scenarios and analysing them or by doing preference tests and similar.

On the left side are the “unordered” domains. The complex domain is something that relationships between cause and effect cannot be seen beforehand, but can be seen in retrospect. So, there is some kind of order, but it is not similar to the order in the simpler domains. Customers’ (collective) market behaviour can be like this. We can only afterwards make complete sense why some product platform took off and another didn’t. We may have made the mistake of thinking that we can understand it beforehand![9]

The fourth domain is the chaos; of which we cannot make sense even afterwards. That domain is something that we have no control of. Politics in some cultures might look like this.

There is also the disorder state, between all the others. It is a “destructive” state, where the domain is not known. Perhaps it is not known yet, but will be clearer once some time passes. Because the domains really exist in the mind of the observer as much as in the reality, this state needs to be guided into one of the others. That is an important element of the model: we can move things from one domain into another by analysing them. We can turn a new, complicated thing into simple once we understand its logic better and gain experience of it. Alternatively, we can decide that the best way to deal with a context is to think of it as complex one and not even try to understand it fully (and thus fool oneself with “knowledge” that has no real basis). We can even make sense of something that looks like a chaos, determine that it is instead complex in nature and later on turn it into something simple and define more and more proven tactics and tools for handling it. Hasan & Katzlauskas (2009) provide some examples of domain transformations in IT, see Table 3. In general, things that started as chaos turned in some ways into the known territory.

Table 3. Some computing issues positioned into the Cynefin domains by Hasan & Katzlauskas (2009), moving from chaos into known during the decades (from top to the bottom).

	Area		
Cynefin domain	Computer programming issues – progress in history	Information system development – progress in history	Selection of issues in 2009 mapped into the domains
Chaos	Assembly languages Individual experts Benevolent hackers	Local in-house solutions Legacy systems The productivity paradox	Network centric advocacy Cloud computing Social technologies

[9] More discussion about the differences of complex and complicated domains – a very common discussion topic – can be found in Sargut & McGrath (2011).

	Area		
Cynefin domain	Computer programming issues – progress in history	Information system development – progress in history	Selection of issues in 2009 mapped into the domains
Complex	Open source Agile programming Various 3GLs emerge	Emergent SAD (Supply And Demand) technologies Outsourcing SSM (System Software Management)	Web 2.0 Convergence Tacit knowledge management Network-centric configurations Communities of interest and practice
Complicated (knowable)	3GLs standardized Software engineering Structured programming	SAD/SDLC (Software Development Life Cycle) research Standardised approaches SAD methods UML	Communities of interest and practice [overlaps complex and knowable] Explicit knowledge management E-commerce IS development BPR (Business Process Re-engineering) Data warehousing Databases
Simple (known)	4GLs Code generators Wizards	ERPs Formal methods CASE tools Legacy systems [overlaps chaos and known]	Hierarchies Websites – Web 1.0 ERPS Command and control
Disorder	Spaghetti code	IS project failures	

All this is important in product and system development. How we classify project domains should define our actions. Analysis and testing can be used to make sense of the situation and make it more controllable. Sometimes we don’t want control, but a reflection point. That situation is at the early stages of product development where chaos can be positive and we don’t want to turn that phase into a mechanistic process, but testing can provide us with glimpses of sense that help us in moving forward.

Now, for the sake of conversation we could map some testing and quality related phenomena into the Cynefin domains. The mapping in Table 4 is just illustrative and in no way scientific or meant to be reliable.

Table 4. Some testing and quality related issues positioned into the Cynefin domains.

	Area	
Cynefin domain	Quality related issues	Testing practice related issues
Chaos	Success factor of disruptive products Behaviour of AI systems and their interactions	
Complex	Functioning of Internet of Things systems and systems of systems Security of systems Socio-technical behaviour of robot-human collaboration in non-limited workspaces Reliable updating of the interlinked infrastructures	Testing of complex dynamic systems Reliability analysis of dynamic complex systems Automated usability testing Testing of AI systems Testing of human-like robots Handling complexity explosion in testing
Complicated (knowable)	Usability criteria for wearables Security of individual application Customer experience criteria Problems with disruptive products Expanded digitalization	Testing of simple dynamic systems Testing of large and complex static systems Overall testing process in continuous deployment Testing in many startup contexts Industrial-grade exploratory testing Quality management in agile and lean development Application of robotics in testing How the actor roles in testing should be divided
Simple (known)	Usability criteria for products Reliability of physical components Mechanisms of human error	Traditional testing and test design techniques Testing in waterfall projects Scientific test design principles Reliability analysis of simple systems
Disorder	Understanding the context	Experiment design

Many of the issues in that table will be discussed in the further chapters.

2.12 General vision of Finland 2030

This dissertation is about analysing the competences. To lay out the basis for discussing those, let’s take a view into the national responses in the form of how Finnish experts see the future of Finland, what it needs to be like in 2030.

As was noted earlier, there are many layers in any activity. The product development companies are where the actual activity happens that this dissertation analyses, and we will spend plenty of words on that later. They work in broader context, the nation, and that in turn operates in the global context. All the levels are in interaction, producing forces and requirements from the higher levels to the ones below. The lower levels do actions that need to fit the environments of the higher levels. The level of nation is critical here, because it provides organisations the general operating environment that they can use for their benefit and what they can in turn produce benefit to – success in business, jobs, satisfaction and well-being for the people of the nation.

That is why we need to at this point take a look into what kind of Finland there might be in the coming decades. We don’t know it for sure, because the future is impossible to know. We can reflect on what knowledgeable people assume it could be, under various forces and with the help of shared will and goal-driven national actions.

In 2012, a foresight process was carried out in Finland consisting of theme group works, national debate, questionnaires and analysis of the information. The process was done in collaboration between the Prime Minister’s Office, the Finnish Innovation Fund Sitra, the Academy of Finland, and the Finnish Funding Agency for Technology and Innovation, Tekes. (Foresight 2030, 2013) Representatives of research, businesses and citizens’ organisations participated in the process. The main result of this foresight process is the definition of what kind of Finland we want to create for 2030. This question is divided into two main levels:

1) What kind of Finland do we aspire to in 2030?
2) What is the aim of the change?

The first level will produce the second level, which reflects the real needs of the society. The work produced analyses around several themes. The relevant ones for this dissertation are “Working life in the future” and “Business regeneration”.

Here, we take a look into what the author thinks are the most relevant aspects in the report. First, some goals for Finland of 2030 include the following.

Finland is a winner in the global share of labour when we concentrate on highly value adding competences on our value network that spread over multiple lines of business, rather than narrow domains that are more fragile, anchoring those to Finland. ICT solutions and product development related competences are clearly among those. In those, Finland “will be” among the best. So, we will have the best companies doing that and those companies will necessarily have the best possibilities of doing excellent work, including good people with suitable competences, practices, management and so on.

Finland is agile and flexible. Changes in the world are hard to foresee. Because of that, we need to be agile and flexible to survive. Related to that are constant renewal and diversity. Parts of this are strong competences that are not tied to certain technologies, but allow refocusing immediately when the need arises. Of course, foresight improves agility; when we see weak and strong signals of emerging changes we can react. An example of this is the studying of new technologies in some volume before they need to be rushed to market.

Finland has an excellent digital economy. Value addition in the digital domain will equal that of the physical domain. We need to be experts in the creation of digital systems of all kinds – from information systems to robotics and everything that links those together. This requires leveraging the already good ICT skills. Obviously, the world will also be more critically dependent on digital systems, and the skills must include the ability to produce reliable and safe systems. We need to be able to design those from concept to details of implementation and to validate their quality efficiently and effectively – a task that is traditionally time consuming and process intensive due to the strictness and variety of the practices required.

There is strong social learning and alliance of learning and work. Competence requirements are constantly changing and we face more complex and multidisciplinary issues. That is why the current educational system or training systems in organisations will not be sufficient. We need to become social learners and be able to tap into each other’s knowledge and approaches. The educational system will be changed, but as importantly, continuous, life lasting learning will finally be obvious to the professionals. This is related to closer relations between elements of competence and knowledge and their application – science and research, education, understanding about working life in broader scopes, business and entrepreneurship more in focus. All this is essential for the development of rich products and for working at the domain of products instead of traditional engineering level, in which the Finns have excelled for decades.

Lower level goals that help in achieving the higher level goals include the following.

Individualised working life. For the adaptation and learning, work needs to be customised for each age, situations in life and other factors. Studying after youth should be more common than now. People could gain deep expertise in a new domain in middle-age and later! New paradigms and changes in technology will require deeper changes. We can learn new technologies, about cultures and approaches at any age, but starting to do so may require some mental and cultural adjustments.

The best education system. This is the traditional value in Finland and other Nordic countries and needs to be maintained and evolved. The changing of competence needs in industry, including various meta-competences, will require changes to the style of education, including more emphasis on experimental learning (such as learning how to develop and test products).

Multi-skilled experts. The future needs multi-skilled experts and development of those is one goal of all educational systems and other learning systems, including the personal ones. One simple example is that in the future it is not sufficient for testers to only excel at the core testing skills, but they need to be able to professionally participate in other activities too; but what they are, will depend on the context.

The best operating environment for companies that lead the way. Finland should provide good environment for the companies and people to work in, including cultural atmosphere that will boost business, product development and attitudes for quality and risk management, which are dearly needed in the more and more digital world.

The world’s best management of global value networks. Finland could be the global leader in chosen value chains. That requires very high and diverse competences. The traditional Finnish assets of discipline and trustworthiness are also important here. Good project management is one of our key “selling points” in this regard.

Entrepreneurship is easy, appreciated and popular. Entrepreneurship has been at least mentally hard. Making it easy will produce more agile business environment and more changes in the workplaces. This dynamism will aid in the general agility and flexibility of the society, but again emphasises those characteristics, and high competences, on the personal level. However, becoming an entrepreneur will require a different competence profile than the engineering occupations.

Some of the interactions between the goals are illustrated in Figure 14.

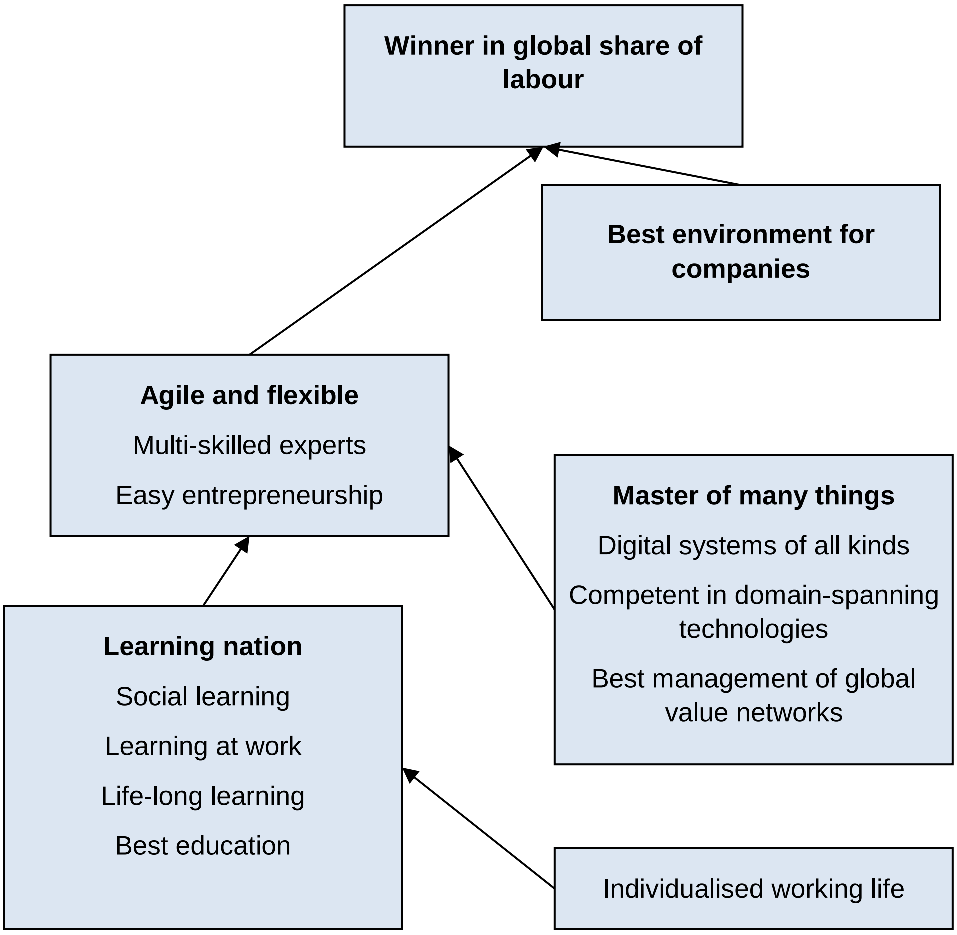

Winner in global share of labour
Best environment for companies
Agile and flexible
Multi-skilled experts
Easy entrepreneurship
Master of many things
Digital systems of all kinds
Competent in domain-spanning technologies
Best management of global value networks
Learning nation
Social learning
Learning at work
Life-long learning
Best education
Individualised working life

Figure 14. How do we create Finland of 2030?

So this is the general nature of Finland that the society has some consensus of creating. Of course, consensus is often prone to thinking. The general idea seems to be that Finland is unique and competent and that it has to become even more unique and competent in order to survive and to succeed. That means that we need to make choices on what to focus in all areas of life and business.

From a risk management perspective, we must remember that the above is just one possible scenario for Finland. In futures research (or futures studies) one common method is to create scenarios[10] (for the history of the approach, see Bradfied et.al, 2005, more about techniques Amer et.al, 2013, and for some applications Meristö et.al,

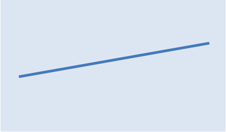
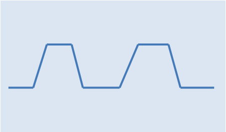
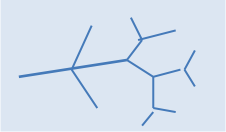
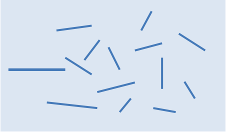
[10] The author participated in the development of a toolbox for product development that had as one element the use of scenarios in the concept development. It is only available in Finnish (Vuori & Kivistö-Rahnasto, 2000). The scenarios made in that process are for nation, product culture, technology and the domain where the product will be used.

2012 & Hellsten, 2007), where the critical variables of the context are identified, their possible values are identified and the combinations of those are made so that alternative views to the future can be crafted. The variables must be independent of each other so that combinations of the values can be made freely. One of the views, the scenarios, is the favourable one, one is the most probable and one is an unwanted one. When we can affect the future, the favourable one is the most important (just like in product development a company would study the product in its best form). Here, we see a scenario with variables such as global positioning, business environment, operating style, education, competences and learning. Even though the variables here are clearly not independent, we can think of this as a good scenario. The risk here is that perhaps the variables will not get the values we wish? What if the education system fails or we cannot create the good environment for the companies, or cannot change the culture to support life-long learning? This is just a side note for the reader, as we are not going to make a risk analysis for Finland here.

It should also be noted that every scenario and visioning process has its own problems, no matter how objective it aims to be. Often it becomes a response to current issues that are acute in the collective mind. When for example the nationally dominant (in cultural, technological and economic sense) mobile phone industry went into problems due to – perceived – too deep focus on one technical platform, it is only natural that more generic competences are sought out. That makes a lot of sense. Still, the coming years may bring other problems and the problems vary in each domain and each company. The age of generic, common truths is over. Still, visions are important for showing us vision, even if they are not correct! One lesson of the futures research is that we need to look into the situation periodically or when we see “serious change”. Then it is time to check whether we are still on the road that leads to the chosen scenario or is the road leading to some other scenario? How should we adjust our thinking and the actions that we do to create the desired future?

Perhaps more important is the dynamism of the world. It is often said that the world is no more linear, leading nicely to some future state (perhaps fluctuating a bit on the way, due to depression periods and such), but more a chaotic one, which is discontinuous and turbulent (see visualisations in Table 5 for a visualisation of this).

Table 5. How the nature of change has changed during decades (After Sydänmaalakka, 2014).

Decade	Characteristic of change	Visualisation
1970s	Stable	
1980s	Periodical	
1990s	Complex	
2000->	Discontinuous: full of surprises, fragmented, chaotic, turbulent, uncontrollable	

The characterisation is not scientific in any meaning of the word (it is quite the opposite), but rough visualisations of how our environment is often perceived. There are two main messages here:

1. The progress is not linear and the end state will be different than we thought.
2. When the environment is fragmented, our analysis of it needs to look at the fragments in the system and be ready to plug them together in various ways, rather than use big models (even alternative ones) of the whole as structure.

The issues related to complexity are discussed later a bit more.

Coming back to the vision of Finland. It is a declaration of goals and wishful thinking and we need to pay more attention to the issues in separate analyses on later chapters, some of them as in separate analyses and some by reflecting on them when discussing other issues. Before that, let’s tackle one essential question. If the goals and visions make sense, don’t they make sense for every country? Or is there something that would make the visions appropriate for us more than for other and help us execute them better than others? There may be some factors and one of them is our “human capital”.

The World Economic Forum is an independent international organization working globally on public-private co-operation. It produces reports about various global issues and one of them is The Human Capital Report (World Economic Forum, 2015). The report has its main tool the Human Capital Index that

"(...) quantifies how countries are developing and deploying their human capital and tracks progress over time. It takes a life-course approach to human capital, evaluating the levels of education, skills and employment available to people in five distinct age groups, starting from under 15s to the over 65s. The Index covers 124 countries, representing between them 92% of the world's people and 98% of its GDP. It measures present performance against an ideal benchmark, and offers insight into how well a country is positioned for deploying talent in the future."

The index considers two main areas: Learning and employment and assesses those for various age groups. For example, for the age group 25-54 it considers for the factors listed in Table 6.

Table 6. Elements of the Human Capital Index (World Economic Forum, 2015a).

	15-24 age groups	25-54 age group
Learning	Enrollment in education • Tertiary education attainment rate • Vocational education attainment rate Educational attainment • Primary education attainment rate • Secondary education attainment rate Quality of education • Quality of education system • Youth literacy rate	Educational attainment • Primary education attainment rate • Secondary education attainment rate • Tertiary education attainment rate Workplace learning • Staff training services • Economic complexity
Employment	Economic participation • Labour force participation rate • Unemployment rate • Underemployment rate • Not in employment, education or training rate • Long term unemployment rate Skills • Incidence of overeducation • Incidence of undereducation • Skill diversity	Economic participation • Labour force participation rate • Unemployment rate • Underemployment rate • Employment gender gap, female-over-male ratio Skills • High-skilled employment share • Medium-skilled employment share • Ease of finding skilled employees

For Finland, there are some very interesting findings in the report. First, Finland's rankings in the various age groups:

- Under 15: 1st (2nd: Ireland)
- 15-24: 2 (1st: Canada, Japan 21st, USA 7th)
- 25-54: 1st (2nd: Switzerland, Japan 5th, USA 17th).

- 55-64: 6th (1st New Zealand, Japan 2nd, USA 15 nd).

Overall, Finland is the best ranked country, but importantly, we are at the very top when it comes to the future potential. The report presents the relationship between GDP per capita and human capital index and shows a clear correlation between the two. So – considering the recent economic problems – Finland as the top country in the index has good potential for doing well. This all is not just statistics, but gives some foundation to the idea, that as a country we still have great potential to do more demanding things and to do them better than other countries. So, the national visions are not base just on stereotypes, but there are important facts behind them.

One interesting characteristic is that in the EU, Finland had in 2014 the highest portion of ICT specialists of the workforce, 6,7%, according to the statistical office of the EU, Eurostat (Eurostat Press Office, 2016), followed closely by Sweden, 6,0%. The next were Luxembourg (5,1%), Estonia (5,0%), The EU average was 3,7%. That implies a country that more than others in EU focuses on, lives and breathes ICT. [11]

One critical question about Finland (especially in the context of this dissertation), is how will the industry work? What kind of products and systems we will be producing that would in turn bring us the necessary income? That will no doubt change more rapidly in the coming years than it did in the past. Yet, it seems that one stable characteristic that we should aim at are products that require our (real or still under-development…) special competences, which are assumed to be related to new technologies. We shall look into the ongoing changes in technologies later on.

Another set of indices is Digibarometri (Kaupan liitto et. al 2016), the digital barometer, produced by several Finnish associations and public actors. It aims at presenting a picture of digital Finland and to give guidance into what should be done for the future in that area. It compares a set of countries on a matrix of variables of three sectors (companies, citizens, and public sector) and three levels (preconditions, usage, and effects). Each matrix cell has four variables and thus the barometer is built from 36 variables, see Table 7.

[11] Another interesting information in the report is the portion of men. In Finland it was 77,1%, the EU average being 81.9%. Sweden was at 80,3%.

Table 7. Variables in Digibarometri 2016 (Kaupan liitto et. al 2016)

	Sector		
Level	Companies	Citizens	Public
Effects	ICT fills the needs of companies Effects of ICT in business models Growth contribution of ICT assets Electric acquisitions in companies	Effect of ICT in labour markets Portion of e-commerce in turnover ICT supports public services Mobile applications in healthcare	ICT and public sector productivity Public support for utilizing ICT Effects of public ICT actions Amount of competition in ICT services
Usage	ICT competence in jobs Electric management of delivery chains Use of cloud in companies Use of ERP systems	Reachability by electronic means Activity in social media Portion of e-commerce in acquisitions ICT competence in home use	Public e-transactions, citizens Openness of public data Public acquisitions of technology products Breath of public e-services
Preconditions	Use of broadband in companies Readiness for cloud services Ease of recruiting ICT personnel Prevalence of IPv6 support in web sites	Prevalence of fast broadband Use of mobile broadband Availability of ICT experts ICT in education	Cyber security, citizens Regulation in technology Good ICT legislation ICT in public information

Without going into the details of the barometer, the main finding in it are:

- Finland was the first in the overall index (75,6), followed closely by Norway (75,1), Denmark (74,1) and Sweden (72,6). Netherlands was next, then United states.
- Finland was again at first spot in the Companies sector.
- In the Citizens sector, Norway, Denmark and Sweden led Finland.
- In the Public sector, Finland was third, behind Estonia and Norway (as a side note, Estonia is often publicly praised in this regard).
- Finland was first in the preconditions level, followed by Sweden and Denmark.
- In the usage level, Finland was only fifth, after Denmark, Netherlands, Norway and USA.
- In the effects level, Finland was third, after Norway and Sweden.

This barometer supports the common conception that Finland has all the possibilities to do great things in the digital world, but the execution may be lacking, be it in public sector or in product development.

Let’s come back to discussing the testing and quality assurance competences. As well as product development competences, those can and need to be unique too. Testing naturally reflects the unique characteristics of the environment and activities around it. To be as beneficial as possible, we should be able to find practices that best suit us and bring added benefits. Ideally, the practices and competences feed from the values and “all good characteristics” of our society and in part support them (thus forming a positive loop of improvement).

On practical level, our testing and other quality practices and competences should help us:

- Make choices about what to produce.
- Help us understand what the customers value and prefer.
- Be innovative and able to create great products that offer value and are desirable.
- Help in making risk-managed entries to new markets and into new technological domains.
- Reduce product and business risks by producing good information to use in planning, design and decisions.
- Be more agile, flexible and adaptable to changing conditions.
- Be more efficient and effective in all of our doings.
- Be faster in product development.
- Develop the working life into more fruitful and pleasant for all of us.
- Etc.

What the important practices and their characteristics and related competences should be like is the core of this dissertation and the short discussion of Finland should prepare us for it. After these preliminaries, we will take a deep look into testing and then again return to the changes in our contexts.

3 Competence models and primary competences for testing and quality assurance in the coming year

3.1 Competences related to changes

In the following chapters, various changes in our environment are presented. They have a strong relation to the competence requirements. The mechanisms are like this.

First, a change requires some competences. As a trivial example, supporting business better by testing obviously requires understanding about business. The competences may be existing or something new. First of all, orientation competences are needed so that everyone can form a proper mind set for the change. Also, the change and the proposed new state of things needs to be understood in order to support the change and the activity after the change. To simplify things, the new activity can be seen as a proxy of the change to reduce the mental models of the system.

Any change will also require practical skills of doing things. The change needs to be implemented and the new actions need to be done properly.

A successful change results in a new activity system, with a new context, new rules and new collaboration that in turn enables creation of new competences, thus enabling yet another positive change!

For all this to happen, individuals and organisations need positive results from the changes and those are expected to come if and when the changes are positive.

The mechanisms are visualised in Figure 15.

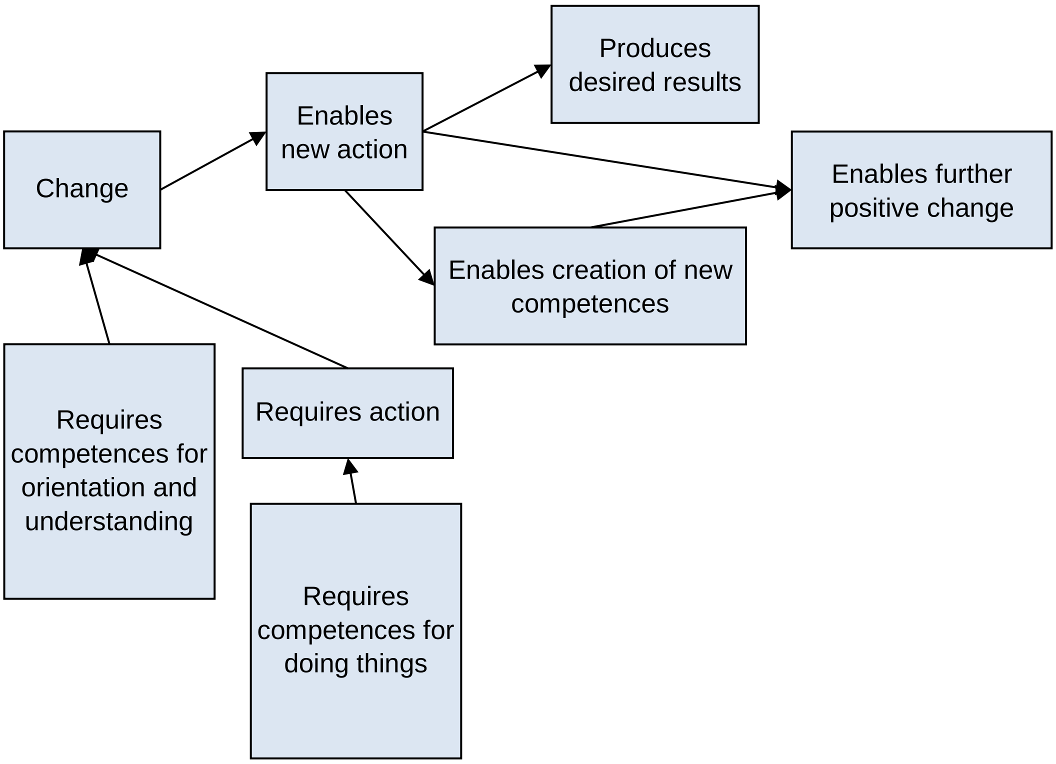

Produces desired results
Enables new action
Change
Enables further positive change
Enables creation of new competences
Requires competences for orientation and understanding
Requires action
Requires competences for doing things

Figure 15. How changes require competences and produce change.

Of course, the changes and competences are not completely new. There will be some new actions and new competences, but at the same time there will remain some old action and old competences. Those may be the most essential in many cases, as when things change, they will still retain some core elements. For example, even though testing will find new forms, the key principles of good tests and experiments will remain largely the same. But while learning, some unlearning is necessary. Thus, some actions and related competences become invalid – indeed, they need to be made invalid. Those three change types are visualised in Figure 16.

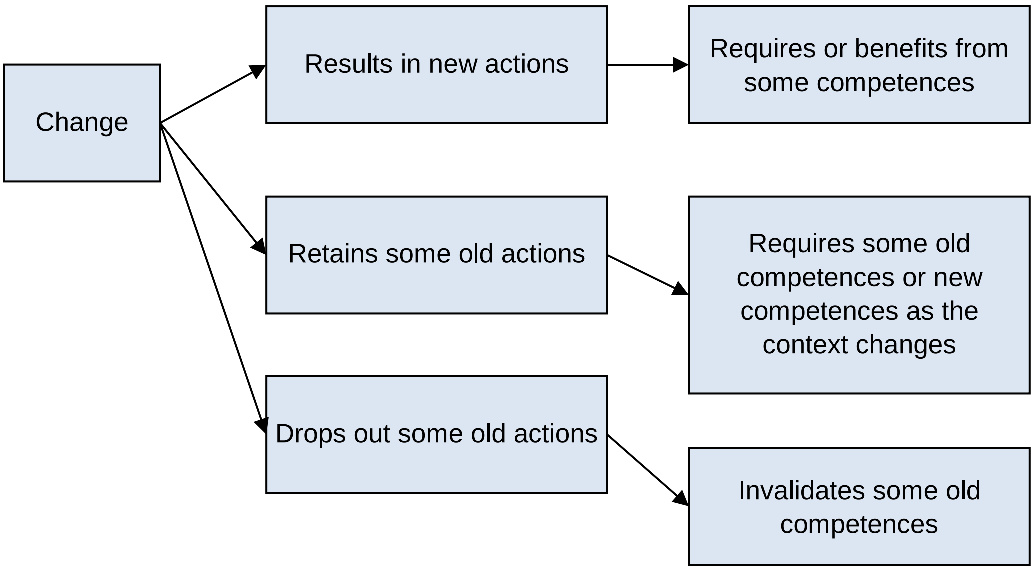

Change
Results in new actions
Retains some old actions
Drops out some old actions
Requires or benefits from some competences
Requires some old competences or new competences as the context changes
Invalidates some old competences

Figure 16. Various types of changes – not only additions are needed.

3.2 Pattern language for changes and competences – change-competence snippets

All software development activities are done in “practices”. A software development process may have phases or tasks for e.g. requirement specification. How those are done, is defined in explicit and or implicit way by the practices used.

How do the practices develop? There are many routes:

- The key persons of a company (founders and others who join in early and form the ways of action) bring with themselves habits that they have used previously. Those are then developed organically further.
- The practices in an organisation’s may be built around some framework, such as Scrum (Scrum Guides, 2015), which is tailored as needed, and perhaps renewed at some point.
- A redesign of practices or a set of them may be defined in a special process aided by a consultant, perhaps when the current practices are seen to be insufficient and it is not clear how to change things.

Mostly changes are made when there are discontinuities in the company: when it starts, when a startup moves into the growth phase, when new products are more critical than before, or when there is a productivity or quality crisis.

Identifying those situations and the new requirements for quality and testing is essential and it is a unique competence that should be present.

The practices can be specified at rough level as patters. Patterns are recurring structures or relationships between elements. The concept is used in trying to understand and share the understanding about complex phenomena both in humans' actions and in a technological system. They are developed by examining an existing or described activity and detecting the pattern by analysing. Patterns use a concise "pattern" language that describes the defining elements of the pattern in a generic form. The elements include attributes like name, context, solution, the resulting context and other information.

Patterns have been developed for many purposes since the 1990's:

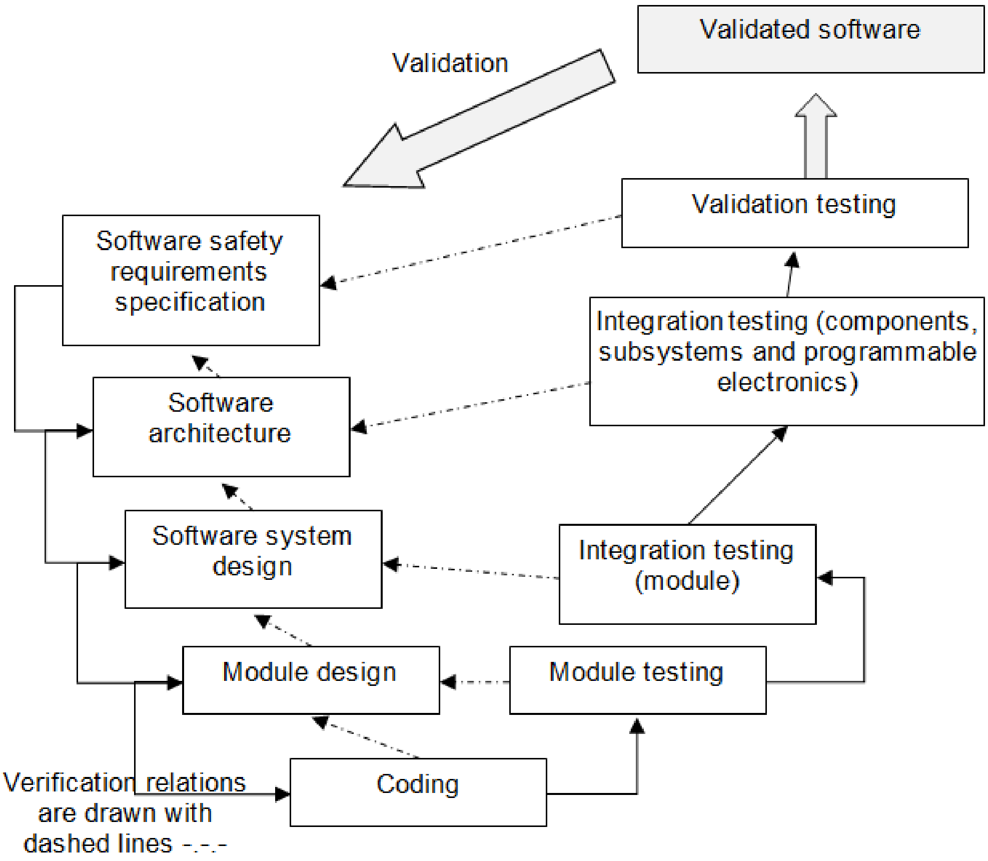
- Organisational patterns have been developed to make organisational structures and behaviour visible, also in software development organisations, including agile development.
- Software design patterns have advanced understanding about software design and architectures.
- At TUT, safety process patterns have been written to describe practices in safety-critical development (Koskinen, Vuori & Katara 2012 and Vuori et al. 2011).
- Use cases are a very important usage of the pattern philosophy (Use case. Wikipedia article). A use case is a very recurring element in every software development project – many of those are identified and presented in a standard way.
- Process patterns capture among other things, software development issues (Ambler, 2011 has built a nice web site around those).
- Project patterns research has included studies of global software development projects (Välimäki, Kääriäinen & Koskimies, 2009).
- Communication and knowledge sharing in the context of software engineering (see Vesiluoma 2009 whose dissertation also contains plenty of information about patterns).

Thus, patterns have evolved into a proven tool for understanding the activity in a domain, any issues in the activities, and to externalise and share knowledge.

Patterns use a concise presentation consisting of short description of key elements. This same principle has been utilised in many organisations as a template for process instructions, helping instructions have a generic, standardised form, short length (optimally one page) and thus better understandability than traditional, longer instructions. Thus, patterns have a lot of potential to be used in companies' processes.

The author has been involved in the creation of a large pattern collection for the development of safety-critical systems (Vuori et al. 2011). An example pattern from that collection, “Verification testing”, is show in Table 8.

Table 8. An example process pattern from Vuori et al. (2011).

Name	Verification Testing
Context	After a software module has been implemented or integrated, the resulting executable program is verified by testing.
Problem	How to verify programs and their components at all abstraction levels?
Forces	During the course of software system development, all work products and implementations are verified. For implemented software, testing is the most important general verification method. Due to the nature of safety-critical development, testing needs to be both high quality and highly efficient.
Solution	Verification testing of software in the context of IEC 61508 (2nd ed.) mostly follows the traditional V-model.  Validated software Validation Validation testing Software safety requirements specification Integration testing (components, subsystems and programmable electronics) Software architecture Software system design Integration testing (module) Module design Module testing Coding Verification relations are drawn with dashed lines -.-.-
	Key features of this scheme:

Name	Verification Testing
	There are several testing levels. Testing at each level is based on a design item or the same level. Software must pass the lower test level before testing of it at the next level can begin. There are strict rules for passing each test level. If the software doesn't pass the testing, it needs to be corrected and the testing repeated. If the correction requires design changes, the designs need to be updated with Software Modification process. If the requirements or specifications that a test or test case is based on changes, the verification obtained with the tests is invalidated and the tests need to be repeated. This includes any regression testing based on impact analysis. Patterns for the test levels will describe the process more.
Resulting Context	A software item or system that has been verified by testing. A software system configuration, behaviour or which is known and understood due to the testing.
Related Patterns	The test level patterns: Module Testing and Simulation, Module Integration Testing, PE Integration Testing Regression Testing Software Validation
Standard References	(See the patterns for test levels) IEC 61508-3 (2nd ed.), table C.12 describes the strictness of various ways of application of dynamic analysis and testing techniques IEC 61508-3 (2nd ed.), table B.5 presents recommended modelling techniques at different SIL levels IEC 61508-3 (2nd ed.), table C.15 describes the strictness of various ways of application of techniques for modelling IEC 61508-3 (2nd ed.), table C.13 describes the strictness of various ways of application of functional and black-box testing techniques
Authors	Matti Vuori
Status	Version 2011-04-29
Notes	See Wikipedia article Software Testing: http://en.wikipedia.org/wiki/Software_testing
Tags	verification, testing, V-model, process

The example clearly describes one situation in a process, within some regulations (standards, product and process requirements), in which one needs to carry out a set of actions due to some force, to reach an outcome – a resulting context.

The competence-related issues that this description presents include:

- Understanding the context and the actions is a necessary competence.

- One must know the mandatory requirements for activity, especially in regulated processes (such as the development of safety-critical systems).
- For every activity there is a body or helping information available, either already collected or for collecting with an Internet search engine.
- No pattern is isolated from the rest. When doing any activity, the tester must understand how activities relate to each other. Most importantly: what will happen to the system under test next?

It should be noted that each type of pattern description is a choice. The patterns could be described in many ways, but compromises need to be made to keep the descriptions as compact as possible.

One question that might arise to “pattern practitioners” is, whether we could try crafting a competence applying pattern language? It might not have a wide range of uses as such and should most likely be seen as an extension to existing pattern languages and patterns – just add a row or two to the pattern description tables. On the other hand, we could devise a shorter format that we could use for examples. One such could be like this.

Table 9. An example structure of a competence utilisation pattern language.

Attribute	Description
Competence case	Name of the pattern – problem, situation, etc.
Context	Context of the case
Actor(s)	Actors, subjects; roles, job titles
Goal	Goal, desired outcome
Competence required	What competences are required to reach the goals

There was an idea of using this pattern systematically in this thesis, but it was realised that in such a broad scope the work would produce a mass of information that would be too structured and instead of clarifying thinking, would tie it too tightly to small level structures.

Instead, more simple “change-competence snippets” are used. The form a summary of competences related to a change (or a stable condition) in a compact way and enable collecting all the snippets together.

Table 10. Structure of a “change-competence snippet” with an example.

Change-competence snippet N	Experimentation culture
Change caused by -> enables	Need for innovation -> validated ideas, concepts
Competence implications (re: quality and testing)	Experiment design skills #A Prototyping skills #A UX and usability testing #A Understanding permission, security, privacy #O #U Data analysis #U #A Creativity #A
Links with	<- Innovation in product development <- Fast product development -> Need for personal understanding of quality -> Flexibility over maturity -> Changing engineering education

A snippet has a name which it is referred by and by which it is linked to other snippets. Links list may be empty. Links will have arrows showing the direction of effect. Right arrow means that the change enables the linked change or has a positive effect on it, and vice versa. In some technology cultures, the change would be called a "driver" for the linked changes, and the other way around. In some pattern cultures the relation could be called a "force". Using the links, snippets can be combined into graphs that show the relationships between changes.

A change has competence implications, which are listed with their competence type – orientation #O, understanding #U or ability #A to do the required actions. These are marked as tags (thus the hash signs) for easy visual and programmatic identification in the text. The types are explained later.

The snippets are introduced during the dissertation in the chapters that analyse the changes. They form a linking between the level of change, the actual changes and the competences implied by the change. Graphs will be generated to visualise the linkings and presented in the appropriate dissertation section. Figure 17 shows an example of such graph, though the actual graphs may use varying visual styles according to space issues. The graphs are created from the snippets by scripts.[12]

[12] First, a Word Visual Basic for Applications (VBA) macro will collect the snippets into one table, then the graphs are extracted from that by another macro that outputs Python code that builds the graphs using Graphviz tool.

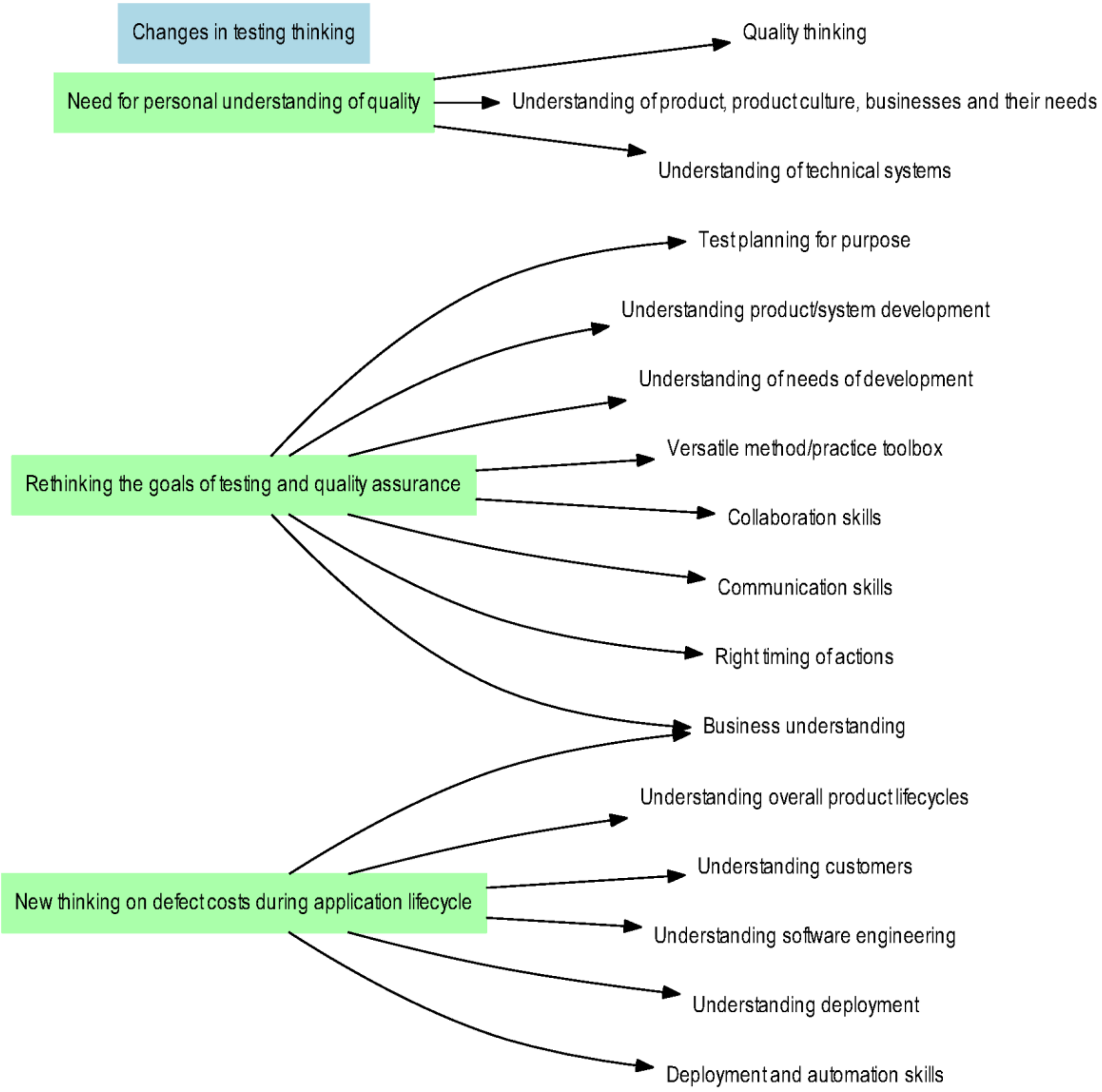

Changes in testing thinking
Need for personal understanding of quality
Quality thinking
Understanding of product, product culture, businesses and their needs
Understanding of technical systems
Rethinking the goals of testing and quality assurance
Test planning for purpose
Understanding product/system development
Understanding of needs of development
Versatile method/practice toolbox
Collaboration skills
Communication skills
Right timing of actions
Business understanding
New thinking on defect costs during application lifecycle
Understanding overall product lifecycles
Understanding customers
Understanding software engineering
Understanding deployment
Deployment and automation skills

Figure 17. Example graph created of the snippets at the level of global environment.

All snippets are also collected into Appendix 3 into one table.

3.3 Competence levels

3.3.1 Knowledge levels used in education, training and certification

There are levels in the competence, not just areas of it. When competence advances, one becomes more able to do her own evaluation of issues and not just apply learned things. Bloom’s taxonomy of educational objectives (Bloom et.al 1956, Buckley & Exton, 2003; Krathwohl, 2002) is often used to characterize the levels of knowledge. It has roots in the education, but has been used in assessing programmer’s knowledge on software systems (Buckley & Exton, 2003), and a version of it is used in the ISQTQ certification system for labelling how well various topics should be understood by a tester and thus, to formulate learning goals.

Basically, the taxonomy presents cognitive levels starting from the lowest:

1. Recall (originally called “knowledge” by Bloom, but many practitioners have usually used recall to reduce ambiguity).
2. Comprehension.
3. Application.
4. Analysis.
5. Synthesis.
6. Evaluation.

The levels are divided further, but we will not discuss the whole hierarchy here.

Recall is the level where a tester remembers and recalls information. The tester might, for example, just remember that there are some testing standards that she needs to use – and that is sufficient, she can look them up as needed.

Comprehension is the level means understanding about concepts. The key concepts of testing should be understood at that level. For example, a tester should understand model-based testing even though she cannot do that herself.

Application is the level where the tester can really use something, for example, testing techniques, such are equivalence partitioning and defect reporting.

Analysis level represents the level where the tester can analyse things, divide wholes into parts. Test analysis, the analysis of system under test and deriving tests for it, goes into this category.

Synthesis level is where any small parts of information can be combined into something new. The parts may include experiences from previous work, items collected from many sources, and of those the tester can for example synthesize an approach or a test strategy for testing a new system. This level of knowledge is essential when context – and not just their contents – change.

Evaluation level is about the ability to make judgements and decisions. For example, based on many things, the tester may need to form an opinion about should a system be ready for delivery to the customer.

It is clear that all the levels are needed in practical work, but they may have a different emphasis depending on the paradigms of testing. For example, if we think of testing as systematic low-competence activity where tests are executed according to test plans and designs made by somebody else, the lower levels seem most essential. For intelligent testing in a new complex domain, the higher levels are critically needed.

As mentioned, the ISTQB certification system uses a taxonomy based on Bloom's taxonomy (ISTQB, 2013):

- Level 1: Remember: "The candidate will recognize, remember and recall a term or concept."
- Level 2: Understand: "The candidate can select the reasons or explanations for statements related to the topic, and can summarize, differentiate, classify and give examples for facts (e.g., compare terms), testing concepts and test procedures (explaining the sequence of tasks)."
- Level 3: Apply: "The candidate can select the correct application of a concept or technique and apply it to a given context. K3 is normally applicable to procedural knowledge. There is no creative act involved such as evaluating a software application or creating a model for given software. When we have a given model and cover the procedural steps to create test cases from the model in the syllabus, then it is K3."
- Level 4: Analyse: "The candidate can separate information related to a procedure or technique into its constituent parts for better understanding, and can distinguish between facts and inferences. Typical application is to analyse a document, software or a project situation and propose appropriate actions to solve a problem or accomplish a task.
- Level 5: Evaluate: "The candidate may make judgments based on criteria and standards. He detects inconsistencies or fallacies within a process or product, determines whether a process or product has internal consistency and detects the effectiveness of a procedure as it is being implemented (e.g., determine if a scientist's conclusions follow from observed data)."
- Level 6: Create: "The candidate puts elements together to form a coherent or functional whole. Typical application is to reorganize elements into a new pattern or structure, devise a procedure for accomplishing some task, or invent a product (e.g., build habitats for a specific purpose)."

The author's view is that the terms for the levels concretize their essence especially on the higher levels (create vs. synthetise) and placing the creation to the top of the list emphasizes its role, because much of traditional evaluation is rule-based by nature.

Obviously, there are other revisions of the taxonomy, of which the revision by Krathwohl (2002) is interesting, as it introduces knowledge dimensions to the taxonomy, making it two dimensional – think of a matrix with the dimensions as rows and the knowledge levels as columns. The dimensions are:

Factual Knowledge: The basic elements that students must know to be acquainted with a discipline or solve problems in it. This consists of knowledge of terminology and knowledge of specific details and elements,

Conceptual Knowledge: The interrelationships among the basic elements within a larger structure that enable them to function together. This consists of knowledge of classifications and categories, knowledge of principles, and generalizations and knowledge of theories, models and structures.

Procedural Knowledge: How to do something; methods of inquiry, and criteria for using skills, algorithms, techniques, and methods. This consists of knowledge of subject-specific skills and algorithms, knowledge of subject-specific techniques and methods, and knowledge of criteria for determining when to use appropriate procedures

Metacognitive Knowledge: Knowledge of cognition in general as well as awareness and knowledge of one's own cognition. This consists of strategic knowledge, knowledge about cognitive tasks, including appropriate contextual and conditional knowledge, and self-knowledge.

This is very much similar with many of the classifications for competence areas.

Kaner (2006b) has also utilised a similar taxonomy and analysed how it could be adopted for software testing and presents the taxonomy in table form in Figure 18. In the taxonomy the cognitive levels are mapped to things that the person should have competence on.

The Knowledge Dimension	The Cognitive Process Dimension					
	Remember	Understand	Apply	Analyse	Evaluate	Create
Facts						
Concepts						
Procedures						
Cognitive strategies						
Models						
Skills						
Attitudes						
Metacognition						

Figure 18. Kaner’s taxonomy (2006b).

In Kaner’s model, the levels of cognitive processes are in the columns, but the rows define the objects of the cognitive processes. For a reader of this dissertation, the meaning of the objects should be clear without writing out Kaner’s descriptions, except for the last one. Metacognition is defined as: “Metacognition refers to the executive process that is involved in such tasks as: planning (such as choosing which procedure or cognitive strategy to adopt for a specific task); estimating how long it will take (or at least, deciding to estimate and figuring out what skill / procedure / slave-labour to apply to obtain that information); monitoring how well you are applying the procedure or strategy; remembering a definition or realizing that you don’t remember it and rooting through Google for an adequate substitute”.

These objects of the cognitive processes are very practical to the point in testing.

3.3.2 Competence levels used in this dissertation

This dissertation proposes three competence types – orientation #O, understanding #U or ability #A to do the required actions. How do they relate to competence levels? Obviously they as such are levels in the tradition, where the higher levels imply an actual ability to do things. That is as such a simple competence level scale. And it is important to call it that. When some systems call even ability to do things a “knowledge level”, they mispresent the ideas. Knowledge is not skill. For example, a music critic might know everything about music, but is not able to make a good composition.

There are levels in the three types. There has not yet been a sufficient motivation to define those exactly or for actual use, but for illustration, the types could have the following imprecise scales:

Orientation #O starts with acknowledging that a system has an element or characteristic and that matters. For example, a person might recognise that systems have user and that they should be considered in some way in the development and testing. But a higher motivation would cause a person to act in some way towards the consideration – talking about it, starting to build personal understanding about users, going to a course about usability testing and so on. Awareness is a term used lately for being culturally tuned ("cultural awareness"). Similarly, one should be aware of the various quality issues.

Similarly understanding #U starts from something vague. Weak understanding about product business might mean recognising the basic elements of business: that there are producers and customers, that money and schedule matters. That is sufficient for considering those matters in testing. Deeper understanding would mean more detailed understanding of B2B customers’ business, their process flows, and business risks and so on. But at that level a tester would still not be able to run a business.

Ability #A to actually do something also has levels, starting from simple things and from there to more complex things. For example, practical test automation competences could start from understanding where and when to do it, then moving to the ability phase with simple modification to scripts others have written; after that comes writing own scripts from scratch and after some more levels, being able to create complex automated test systems.

Combined into one example, the types would look something like the following for a competence "security testing".

Orientation #O:

- Understanding that information related risks exist and that everybody must work for controlling them.
- Understanding the severity of the potential issues for customers, users and the company's business.
- Willingness to communicate about security, and in managerial role, to give resources for handling it (training, experts, testing tools, time in projects).
- Readiness to take action if there are deficiencies related to security.

Understanding #U:

- Understanding of the mechanisms of security breaches, such as common attack vectors of OWASP.
- Understanding what actions should be carried out in development projects for assessing risks, analysing security vulnerabilities and in testing and the rigour and resources needed for those tasks.
- Understanding of the principles of handling any security violation situation.

Ability #A:

- Ability to perform information risk analysis for a system or product concept of business plan.
- Ability to perform security analysis for designs.
- Ability to select security testing tools and integrate their use into the development lifecycle used.
- Ability to design and perform security tests.

The practical items on the lists will always somewhat depend on the context and a person's role.

3.4 Competence model structured by the activity system

As competence is always in relation to something in the activity system, we use as a structure the elements of the “triangle” model of action research, presented earlier, with some additions such as the overall environment.

The table below shows how the terms have been concretised.

Table 11. Elements of the activity system used in competence analysis and their abstract terms in action research

Action research triangle	Element used here
Subject (person or team)	Self
Object	System under test
Mediating artefact (tools)	Tools and methods
Outcome	Development goals
Community	Organisation
Division of labour	Teamwork
Rules	Processes

In the analysis we list as the attributes of the competences the following:

1) Type of the competence: is it something that orients an actor, makes her understood what should be done, or an actual skill of doing something essential.

The competences are divided into three types, #O, #U and #A as described already.

The orientation competences and understanding competences can be quite similar for all roles that people do testing in, but the concrete abilities are such that they divide the real experts and those who can do tasks in supporting roles or in simple form.

2) Forces for that competence, based on the nature of systems development and the change vectors presented earlier.

3) The most important contexts where a given competence is the most useful – understanding that for example safety-critical industrial development has different needs than game development.

NOTE 3: A word about experience. People have asked why is experience not mentioned as a competence? That is because it is not a competence, but a way to gain competence. The working environment in testing is really quite complex with many implicit / tacit elements around that doing things is the only way to learn to perform. Still, it is just a way to learn and can be accelerated or replaced by for example training.

This architecture is used with varying formality to collect the competences related to the activity system and considering the changes in it.

4 Current understanding about a software tester's competence

4.1 Introduction

In this section, we look into what is currently known about tester's competence. Because competence is very much based on what a tester does and how she things, we need to analyse different views into what testing is even on a paradigmatic level. Simple concrete actions done in tasks and roles are not sufficient for us to create an understanding about the "essence" of being a tester and acting truly like one. We need to go deeper into areas that are not visible, but carved into the mindset of testers. Also, we shall take a look into how testers work in organisations, because cultural factors and organisational relations form our behaviour perhaps the most. Furthermore, we look into how testers are part of larger contexts: ecosystems and communities.

The reader should remember that contexts are contained inside larger contexts and thus we need to assess the whole "stack" (to use a technical term) from inside the thinking patterns up.

All this is necessary preparation for the later analysis of the changes around us and the contexts where we work in.

4.2 Different views to testing

4.2.1 The schools of software testing show differing thinking about testing

As in any area of life, there are paradigmatic schools in software testing too. The practitioners are gradually becoming familiar with that, but still the schools have not been analysed much in literature – perhaps because practitioners may often assume that their school is not a school, but the most perfected and correct approach. Here a look into the schools is in order. The schools should not be thought as something definitive and with clear boundaries, but just a tool of making visible differing approaches to testing. That is why they definitely will and look different depending on the culture and viewpoints from which they are defined. Because of that, they must not be taken too literally. Furthermore, not everyone agrees that the idea of schools is a good idea. Kaner (2015) provides a conference discussion between a proponent (Kaner) and an opponent (consultant Rex Black) about the idea of schools of testing.

Overall, testing that is based on one approach can be dangerously narrow-sighted. It is understood that multiple viewpoints and diversity of methodologies are beneficial. Furthermore, the characteristics of systems vary and evolve, and narrow-minded approaches are not as flexible and adaptable to new situations as one would wish. Companies may even move into new areas and system types and then the stickiness of approaches can be a problem.

The schools can sometimes be a part of the culture of a domain’s and is something the people living in cannot even acknowledge – “does the fish know it lives in water” is an old question illustrating this[13]. So, this is clearly something that needs thinking of.

Then again, in human activity, there is never only one truth, and even in the same context, two approaches can be considered the best ones. This is in contrast to engineering, where there might be just one truth and thus the best approach could be found “objectively”. The practitioners are often heard saying that there are not best practices (consultants may obviously present their solution as the universally best one). So different approaches need to be respected and we need to learn from them – from what is good in them and what deficiencies and even dangers they might have in some circumstances. That understanding can be valuable in the collaboration between people who have different ideas that we do and when times and needs change.

Let’s take a look into how the concept of schools has evolved during the years. Kaner (2006a) provides a view into how he and Bach discussed the various paradigms of testing and presents a classification of schools as:

- Factory school, which emphasises reduction of testing tasks to routines that can be automated or delegated to cheap labour.
- Control school, which emphasises standards and processes that enforce or rely heavily on standards.
- Testing driven school, which emphasises code-focused testing, which is done by programmers. Note that this does not imply the test-driven development paradigm.

[13] There’s even a book about national cultures affecting corporate strategy by almost that name, “Fish Can’t See Water” (Hammerich & Lewis, 2013).

- Analytical school, which emphasises analytical methods for assessing the quality of the software, including improvement of testability by improved precision of specifications and many types of modelling.
- Context-driven school: emphasis on adapting to the circumstances under which the product is developed and used.

Another analysis of the schools is given by Pettichord (2007). First he describes a “school” to be defined by intellectual affinity, social interaction and common goals and to be made of hierarchies of values, exemplar techniques, the standards of criticism, organising institutions and common vocabulary. Then he presents the schools as:

- Analytical school, which sees testing as rigorous and technical. It has many proponents in academia. It sees software as a logical artefact and that testing is technical and that the key question is “which techniques should we use”,
- Quality School, which emphasises process, policing developers and acting as the gatekeeper. In its viewpoint, quality requires discipline and testing determines whether development processes are followed.
- Standard school, which sees testing as a way to measure progress with emphasis on cost and repeatable standards. It sees that testing must be managed and cost-effective and that testing validates the product and measures development progress.
- Context-driven school, which emphasises people, seeking bugs that stakeholders care about. It understands that software is created by people and people set the context. Testers find bugs and a bug is anything that matters to a stakeholder. Testing is also multidisciplinary.
- Agile school, which uses testing to prove that development is complete; emphasizes automated testing. Testing is an on-going conversation and tells that a development story is complete and “done”.

A year later, Bach (2008) acknowledges the existence of context-driven school (which is his “religion” – by his own words), factory school, quality control school, analytical school and test-driven design school.

Of course, those are stereotypes only and there does not exist a single, objective classification for the schools – the classification depends on the context in which testing is looked at. The schools are also prone to mix and produce new cultures – for example, context-driven testing is as of this writing very common in agile development, not just test automation, as Pettichord saw it. So, the schools presented should not be relied on too much, but just be seen as an eye-opener.

Because the situations change, we are allowed to present the author’s personal view to the schools as they show in Finland in 2013. Again, similarly as the “classical” school descriptions, this classification is not validated in any way and also the schools overlap, meaning that they are not pure schools. Instead we could call them populations that have a dominant approach, but as that is a too complex term, we will just call them “schools”:

Standardization “school”. In it, testing should be based on standards, such as ISO/IEC 29119 or ISTQB and the proof of conformance is seen as proof of competence. Testing is similar in all contexts and should have similar practices. Planning and organisation are usually seen as essential. This kind of thinking seems to be often found in industrial manufacturing settings where there is a tradition to rely on standards as proof of correct action for cultural reasons and for juridical purposes – If for example a flaw is found in a product, it will look good in the court if the product was tested based on international standards.

Quality management and assurance “school”. In it, testing is a part of quality management and assurance. Testing is a means of verification and validation (V&V) and carried out with defined, repeatable practices. Testing is carefully defined in the software production processes and in separate sub processes. Test management and measurement have a big role. Often closely related to standardization “school”.

Automation “school”. All testing should be automated and no testing should be done manually. Defects are in nature such that a test automat is able to find them. Testing should have near-perfect coverage. Testing is really a form of a logistic process. This school is seen in the domains where speed of development is emphasised and where the setting is in engineering.

Developer-centric “school”. Unit and integration testing (and similar) done by developers is mostly sufficient. Testing should be integrated in software production processes. Test automation must be practical, fast, simple and easy. Agile development has been accused of being very much developer-centric and a large portion of practical agile development relies on developers and their skills on testing.

QA centric test automation “school”. Testing requires high-end automation. No amount of complexion is bad, as the challenges are complex. Testing requires good science behind it for optimized systems and results. Good planning and an analytical approach are essential. This school is found mostly in complex industrial settings, for example in verification and validation of advanced automation systems.

The context-driven “school”. How testing is done should depend on the context. Assessing the context is critical for the success of testing. There are no general best practices. It should be noted that mature context-driven testing is not opposed to test automation, while understands that new defects are not often found by test automation.

The intelligent testing “school”. Testing is intelligent activity that requires human capabilities to do it. Exploration is essential for finding defects. There is more to every test condition than meets the eye. Tools can help in testing but they do not “do” testing. There is a difference between simple checking and true testing that reveals new information. This is a sub-school of context-driven school.

The routine “school”. Testing is simple activity that requires no special skills. Most anyone in an organisation is capable of testing. Discipline and accuracy and following plans are most important things in tester’s work. This is a practical "school" that represents somewhat outdated ideas of testing, but would have been relevant during the era when large portion of testing was manual, repetitive regression testing, which now has been replaced by regression testing automation.

Holistic “school”. There is no single best way to do testing. Good testing uses many paradigms in a mix that depends on the goals and context. All approaches complement each other. Having contrasting paradigms is good for testing. This is actually close to mature context-driven testing, but also realises that there are situations where a one-size fits all may be a good choice.

Some schools have a lesser role as of writing this (2015). First, the analytical school. There are mostly no “general analytical persons” and even the role term "test analyst" is rarely met at least in Finland. Instead, all paradigms have analytical people doing perhaps research based method development, as well as pragmatic people in many tasks. Second, the factory school. The software factories and testing factories were a passing phenomenon at the turn of the century and are seldom mentioned anymore. Some of its ideas are included in other schools, though.

A repeated word of warning: these schools are not meant to take too literally and are not to be assumed to be stable. They are only presented to visualise the diversity of thinking and approaches in our rich world of testing and partly the existence clinging into certain ideas in testing. That is something to bear in mind.

Most of the differences in the schools can for practical purposes be condensed into two cultures: the classical plan-driven school, based on careful planning, managing and documenting of tests, and the context-driven testing. Therefore, we will look into those in more detail.

4.2.2 The classical plan-driven approach to software testing

The viewpoints of the plan-driven approach are widely described in academic literature, in many textbooks and in standards and in the ISTQB certification documents, which we will assess later in this dissertation. Let’s look into some of the important literature. We begin from history, because seeing where we come from is essential for understanding where we are going to. One of the first proper descriptions of software testing is found in Leeds and Weinberg (1961). Obviously, the software systems of that day were very different than today and globally testing was a much unknown activity, but the book already contains fine understanding about testing. It includes emphasis for a systematic approach, understanding about psychological factors, code reviews before testing, writing test cases, using tools to aid in testing. So, the book describes a quite holistic view into testing, while such activities as test management or integration testing were not relevant at that time – software development was mostly individuals' activity with simple programs.

Boris Beizer is one of the "gurus" of the field and his book Software Testing Techniques (Beitzer, 1990) is a classic from an era when testing was already getting systematised and many people had had experience of doing it in professional manner. As the title implies, the focus during that era was deeply in the testing techniques and book contains almost 400 pages of information for testing various aspects of software. Besides that, it contains a taxonomy of defects and around 30 pages of discussion about the implementation of testing, including advice on testing strategies and testing tools, also used for test automation. (Note that we look here at the second edition of the book, from 1990; the first edition was published in 1983.) It is interesting to note that the book contained 37 pages of references, so clearly the body of knowledge about testing was at that time quite large. That book was mostly about white-box testing and Beizer later continued with a book about black-box testing (Beizer, 1995), which follows the expansion of testing to levels where the testers do not have access to program code, that is, system testing in the traditional testing vocabulary.

Unit testing is the core of software testing. Runeson (2006) reports a survey for 50 companies from various domains that produced among others the following views to competence related to that testing type (list condensed by the author):

- While it is the developers' competence, external independent testers are also favoured.
- It is mostly functional testing so the understanding about the system is related mostly to the functionality of the program code.
- Testing tools have a big role in it.
- Testing is problematic with code that has large data structures or depends on those, highlighting the skill of creating compact code for testability.
- It is hard to motivate developers to do unit testing, which points to a management competence area.

The core skills of testing – such as creating test cases – were not mentioned as issues in the study, but it is generally understood that for unit testing, good test cases are a critical issue. The competency related to this is the ability to utilise effective test case design techniques, which are documented in just about every textbook about testing.

As software systems became more complicated and the project culture matured, emphasis turned into managing testing during a software development project. Of note in that literature is the book about the TMap approach (Pol et al. 2002). While TMap is a proprietary method, it shows the thinking of that time about how testing should be managed in especially large projects and provides a framework, instructions and tools for that. It presents important concepts such as master test planning, the lifecycle of testing, test strategy, testability review, checklists for quality characteristics, test organisation, test control and metrics, test process improvement, and testing infrastructure. In that way, it has viewpoints into testing projects in the context of an organisation. The model has been widely used as such and as a reference model for companies overall testing processes.

It is good to see how the literature builds layers upon previous knowledge, in a continuum that also mirrors the changing culture of software development.

After the view into the progression of the literature, the next texts to review are the tester certification schemes. Those are not reports of scientific study, but contain a collective view into how experts saw the situation at the time of writing the certification syllabi. A notable certification scheme is ISTQB (2012), which is the dominant tester certification system, available in most countries of the world. The system defines the testers' competence into levels of foundation level, advanced level and expert level, which are described in corresponding syllabi. The foundation level (the syllabus ISTQB, 2011a and in more expanded unofficial form the textbook Graham et al. 2008) describes the very beginning level of competence, consisting of understanding fundamental principles of testing, the psychology of testing, tester's code of ethics, testing through the software lifecycle, static techniques, test design techniques, test management, tool support for testing. It could be said that a beginning tester should know a little of most areas of testing. Indeed, the focus is on knowing, as mostly the certificate applicants need to know things and only be able to apply their knowledge in practice in some areas of test design and reporting of incidents (errors). The advanced level (ISTQB, 2007) supplements the competence with another layer of knowledge in those areas, but also widens the scope into quality assurance topics, such as reviews and more demanding areas like test process improvement, test automation and people skills. Those are understood to be things that a real professional can apply. The expert level (ISTQB, 2015) is meant for advanced experts who specialise in some area in testing that most people do not. In the fall of 2015, that level was divided into modules for test management, test improvement, test automation engineering and security testing (ISTQB, 2015). So, the ISTQB's model mirrors a general view of growing expertise in steps from beginner to an expert. It also mirrors the division of people currently working in testing – the foundation level is targeted to all "testers", the advanced level to managers and more skilled professionals, of which there are perhaps 10 % of all people working in testing (this is just a rough, illustrative figure) and the expert level certificate is meant for the rare experts of which there are just a couple in a company or a company unit.

To have a glimpse of what kind of topics the ISTQB certification contains, the contents of the foundation level certificate syllabus are listed in Appendix 1.

Who does the testing is an essential question – or, who should have the competence to do testing? In the early literature (Leeds and Weinberg 1961; Beitzer, 1990), it was implicitly clear that the profession and role in question was the programmer. Only later did the role of “test engineer” and other roles appear in texts due to the more process-like testing activity and view that software development is done in projects. The essential competencies for those roles are the abilities to carry out the task given to each role in the contexts where that role is relevant. Indeed, the TMap handbook (Pol et al. 2002) defines the following roles and short descriptions for their skills and competences (in the TMap context): testing, team leader, test management, methodology support, technical support, subject matter support, intermediary, control, test regulation, monitoring, coordination and advice, system management support, application integrator, test automation architect, and test automation engineer.

The ISTQB certification system (ISTQB, 2011a) also emphasises roles, and acknowledges roles such as testers, test analysts, test engineers, test consultants, test managers, user acceptance testers, and software developers. At Expert level, ISTQB defines in the context of test improvement (ISTQB, 2015) also the roles of test improver and assessor.

The ISTQB Expert level module on Test Management (ISTQB, 2011b) takes the team roles further by presenting possible roles as, black box tester (test analyst), white box tester (technical test analyst), performance test engineer (technical test analyst), test environment and data administrator, tools developer, test lead (test manager), test architect, test automation specialist (technical test analyst), and security test engineer (technical test analyst).

Many of the roles reflect what have been seen in practice, but are abstracted in such way that their validity is questionable. Indeed, the specification of certification syllabi is committee work and while there are validation efforts by committees (in this case, national boards), the end result may often be a political compromise.

New testing paradigms introduce new viewpoints to the roles. Model-based testing is of special interest here, as it is an active research topic and will – for many reasons not analysed here – have a wide application and great influence in the future. For that, we will start looking at Jääskeläinen (2011), which describes the roles in using the TEMA toolset (model-based testing is by nature very tool centric) as follows:

- Test modeller: responsible for the creation and maintenance of the models used in testing.
- Test designer: defines what kind of tests are to be created and oversees their execution.

- Chief tester: oversees the whole test process.
- Test debugger: responsible for tracing any anomalies found back to their cause.
- Test engineer.

Especially the modeller and that role’s modelling skills are important here, as those are skills that are rare in the field of testing and currently include software developer’s skills. It is to be noted that whereas debugging is traditionally left outside of the scope of testing, and included as a software developer’s, here it is clearly a task of a dedicated role in the testing personnel, leading into information to the personnel. It should also be noted that the roles defined in the context of new paradigms are prone to evolve and change. For example, in an earlier paper (Jääskeläinen et al. 2008) in place of the Test Engineer was Test Model Execution Specialist. When testing technologies evolve, general roles can utilise them!

Another view to the certification issue is what competence is required from “general software development professionals”? Astiagarra et al. (2010) analyses the testing related portion of ACM Certified Software Development Professional (CSDP) Certification and find that “software test constitutes 5-17% of the exam, while issues relating to software quality fall somewhere between 6-8%. This yields a combined range of 21–25%” and report that syllabus a module “Software Testing” covers software testing overview, test types, and test design, while a module for software quality” includes software verification and validation, software quality assurance and data collection. Authors note that execution is entirely omitted along with many other things that are “core to” software testing in practice.

This is essential, as programmers do plenty of testing and many technical testing tasks are very much based on programming skills.

(Note that testers’ tasks often include various types of scripting, but we will not look further into that here.)

Let’s return briefly back to a higher level of purposes of testing. We already noted that ISTQB (2011a) describes the purpose to be finding defects, gaining confidence about the level of quality, providing information for decision-making, and preventing defects. At this stage, without further analysis we can see that those purposes imply a need for many kinds of skills. Finding defects means capability for carrying out all necessary actions of planning, implementing and executing of tests. Gaining confidence about the level of quality requires understanding about software quality, its elements and the usage of any particular software system. Providing information for decision-making requires understanding about the “business” level issues in software, priorities and risks and ability to communicate information in such a way that others can make valid decisions based on that. Preventing defects requires ability to communicate about software defects in such way that their re-occurrence can be prevented, and ability to analyse software defects and to understand their contributing factors in practices, processes, culture, tools, etc.

Some respected testing books also list characteristics of a good tester, but in a rather anecdotic way. Hetzel (1988) lists “Essential Ingredients to Good Software Testing” like this:

- Creativity and insight are important.
- Business knowledge is important.
- Testing experience is important.
- Testing methodology is important.

As another anecdote, Craig & Jaskiel (2002) show how students in their test management course see a good tester. The list has been arranged here under sub-headings by the author.

- General characteristics and personality: Has a good personality, is open-minded, has a positive attitude, handles stress well, has a sense of humour.
- Thinking style: Is logical, is a quick thinker, is inquisitive, is detail-oriented, has good common sense.
- Knowledge: Has functional/business knowledge, knows specific tools, understands the software development lifecycle.
- Background: Has a technical background, but does not want to be a programmer, has testing experience.
- Working in a team: Is a team player, is self-starting, is flexible, is self-reliant, is politically astute.

That is a quite varied list of characteristics and shows a practical view to many positive factors that companies have learned to respect. However, those are really just anecdotes, but important as such as they document the culture of that dat.

Another, personal anecdote from 2004. At that point, companies still lived mostly in the systematic ideal, but signs of a change to more agile way of working were seen. The author listed positive tester characteristics in 2004 (Vuori, 2004). The list was based on the analysis and reflections on the environment and requirements of testers working in large teams in outsourced testing services for large product development companies as clients. The characteristics included:

1) General characteristics:

- Flexibility. The tester is flexible in changing amount of work. Sometimes there is nothing to test, sometimes too much!
- The test engineer has to be interested on the functionality of the whole program.
- Strength of character. Courage to tell opinions and to speak for quality.

- Systematic way of working. The testing must be carried out according to the plans.
- Accuracy and carefulness. Testing is accurate work where little details matter the most.
- Realism regarding to the functioning of new technologies, applications and products. The test engineers can be in the software development team as a counter balance to the overly optimistic software designers.
- Dependability. Testing is dealing with unreleased product that requires absolute security about their existence and features.

2) Knowledge and skills:

- It is important that the test engineers have general knowledge of how different kinds of programs work.
- Written presentation skills are important.
- Team work skills are important.
- Has competence on the theoretic principles of testing in appropriate level.
- Is able to analyse and interpret problems. What causes errors, why doesn't the test environment work as planned?
- Communicates clearly both in writing and orally. The errors of the software must be reported clearly so that the software developers understand how the software behaves erroneously.
- A plus: Is suitably creative. Testing is not always just typing in simple test cases, but often requires improvising skills. Still, too creative or artistic people are not suitable for testing work.
- A plus: Has some level of programming skills. Basic programming skills help in estimating where there might be errors. In some types of testing it is necessary to be able to read program code, but not all.

3) Social characteristics

- The test engineer has social skills. Has an active and positive customer service attitude when dealing with the clients.
- The test engineer is extrovert. Paradoxically, jobs like testing require people who immediately raise problems and keep the team spirit up.

That list clearly shows the traditional values, but emphasis the test engineer as a social profession in an organisation.

4.2.3 Context-driven testing

The thinking of context-driven school is mostly described in non-academic books and sometimes it has been claimed by the most respected experts themselves that only small part of the know-how of the acknowledged experts has been documented. However, the academic research of the context driven approach has been gaining volume, as exemplified by the dissertation of Itkonen (2011) that focused on exploratory and experience-based testing. The main finding is that in exploratory testing (Whittaker, 2008; Hendrickson, 2013), the testers can more fully utilise their knowledge, but what that most valuable knowledge is, is still somewhat open for interpretation in a scientific sense.

The basic principles of context-driven testing are published in a manifest-like list in Kaner & Bach (2012):

1. The value of any practice depends on its context.
2. There are good practices in context, but there are no best practices.
3. People, working together, are the most important part of any project's context.
4. Projects unfold over time in ways that are often not predictable.
5. The product is a solution. If the problem isn't solved, the product doesn't work.
6. Good software testing is a challenging intellectual process.
7. Only through judgment and skill, exercised cooperatively throughout the entire project, are we able to do the right things at the right times to effectively test our products.

One of the most respected books that document the context-driven school's thinking is Lessons Learned in Software testing (Kaner et al. 2002), so it is important to see what kind of view it provides to competence. The book consists of a large number of individual lessons, but its structure reveals the main themes:

- Understanding the role of tester.
- Thinking like a tester.
- Ability to use testing techniques.
- An attitude to advocate errors.
- Documenting testing.
- Interacting with developers.
- Managing the test project.
- Managing the testing group.
- Managing own career.
- Planning the test strategy.

When we look at those themes we need to remember that the book is from a year when the agile software development culture largely did not exist and thus, the group-related issues might today be structured and expressed differently. Still, the valuable main lesson here is the context of working with people, in a context, the importance of which is often assessed to increase. The very basis of testing should, based on the book, be selecting the practices according to the situations at hand.

In the absence of that many analytical works of the context school, we must not shut our eyes from other documents, such as presentations given by consultants who have a significant role in the testing culture, as long as we acknowledge that position. Bolton (2009) is one such consultant and outlines critical principles and tester skills that should lead into a brighter future for testing. Some highlights (slightly edited):

- Testing is a deeply human activity, which is strengthened by the unique contribution of the individual tester.
- Testing is about exploration, discovery, investigation and learning.
- Innovative ideas come from outside the craft.
- Testers do not ask merely that software passes or fails, they ask “Is there a problem here”.
- Testers embrace change, deal with uncertainty, and handle time pressure.
- Testers seek and provide information, actively question, reject distorted information; they are sceptics.
- Testers consider cost vs. value and eliminate wasteful activity.
- Testing is complex so testers are diverse. Testers seek simplification, but do not trust it.
- Testers emphasize stories over numbers.

The approach can be seen to be scientific in nature – a tester explores the system under test, makes observations and deductions based on those. While exploratory and other context-centred testing styles are often associated with agile software development, the practices are often very different. The analysis of those differences is however outside the scope of this review so we will get into that later during the research.

With the emergence of agile software development, many of the principles of the context school were applied in “agile testing”, which encompasses both agile ways of doing testing and doing testing in the context of agile software development. Crispin and Gregory (2009) is one notable textbook of that. When we look at it contents, it very much emphasises organisational and cultural challenges and understanding the purpose of testing and has an approach that is closely linked to the goals and principles of agile software development – creating value for stakeholders with each testing activity, and not just testing or neutral information.

In the context-driven school there are no formal certifications, but Kaner’s BBST courses, such as BBST Foundations course (BBST Testing Course, 2014) aim at the same goal. It is interesting to compare the contents of BBST Foundations course to the ISTQB Foundations syllabus (ISTQB, 2011a).

The table of contents of the BBST Foundations of software testing workbook (Kaner & Fiedler, 2014) reads like this:

- Lecture 1: Overview & Basic Definitions.
- Lecture 2: Strategy (including role of testing group).

- Lecture 3: Oracles.
- Lecture 4: Programming Fundamentals & Coverage.
- Lecture 5: The Impossibility of Complete Testing.
- Lecture 6: Introduction to Measurement.

From the beginning it is clear that the contents cover much less than ISTQB (2011a). That is a deliberate choice from Kaner, who thinks that is it better to have deeper know-how on the essentials than shallow knowledge about many things. The idea really is to learn to test and not just to understand testing. In fact, here is reflected the idea already expressed in “Testing Computer Software” book (Kaner et al., 1999) the learnings are for someone who will do the testing in the real world, when nobody else does. That happens in a realistic environment, where textbook descriptions of V-models and such just do not exist. Because of those presuppositions, it makes sense to concentrate on the very essential things that are needed for testing – and for good testing. Because of that, a very large portion of the things in the ISTQB syllabus are not included – different goals, different contexts require different content. The “lacking” topics include (ISTQB numbering) things such as 1.6 Code of Ethics, 2. Testing Throughout the Software Life Cycle, 5. Test Management and 6. Tool Support for Testing. For the comparison, the contents of the foundation level certificate syllabus are listed in Appendix 1.

One interesting topic in the BBST foundation course is the computer representation of numbers. This is definitely something that testers should understand as it is invaluable information in understanding how data is stored and in designing good test cases in domain[14] testing, but those who do not have an computing education background will not necessarily know anything about it.

4.2.4 Comparison of the tester stereotypes

As an illustration of how the views regarding testing changed during the early years of the first decade of the 21st century, the author wrote the analysis show in Table 12 to be used in tester training with the idea of showing how the role models should change from the traditional. This was an era when exploratory testing was brought into discussion and the traditional outlook of a professional tester was questioned. A stereotype regarding the “old way” was a mechanistic, almost a robot-like worker who just followed plans and test designs. But now a new type of tester was emerging, one that works like a detective. A comparison of stereotypes always brings to light phenomena in a very clear way.

[14] Domain here refers to the domain of values for some variable or a data field, not for example a business domain.

Table 12. Comparison of “mechanistic robot” and “creative detective” tester stereotypes from the author, 2004.

Characteristic	Mechanistic robot	Creative detective
Relation to testing work		
Purpose of work.	Execution of tests.	Searching for defects. Revealing of the biggest risks.
Underlying view of professionalism.	Systematic approach. Repeatability in detail level. Preplanning, planning is ready in one shot. Information comes in as requirements. Monitoring of processes.	Understanding about what is being done. Personal understanding about the target of activity, generic civilization. Searching information from many sources. Information builds a frame and basis. Learning and agile steering of work based on what has been learned.
Level of understanding testing.	Knowledge and execution dominate.	Ability to apply principles (not just techniques) in ways that each situation requires.
Supervision of work.	Based of test planner’s and project manager’s beforehand made decisions.	Testing is given goals, the approach is self-steering.
Competence profile.	The accuracy criteria of demanding factory work.	Intellectualism. Self-guidance. Understanding about product culture – products, their true requirements, users and way of using.
Relation to learning.	Learning is learning of routines, which shows in more efficient work.	Learning is about the products and typical and potential problems.
Challenges in testing.	It is challenging, because it is repetitive work where business varies.	It is challenging, because finding defects is intellectually challenging: what have the developers not considered?
Relation to insecurity.	Insecurity and insufficient information are problems.	Insecurity and insufficient information are natural.
Replaceability.	Based on a machine’s characteristics and is wanted to be replaced with automated testing.	Based on human strengths and thus understood to be irreplaceable.

Characteristic	Mechanistic robot	Creative detective
Relation to the target of testing		
Underlying view of the product.	Specification-positiveness: Specifications describe everything that is essential.	Understanding about the nature of specifications: they can never cover all issues, they only describe how a product differs from other products
Things to test	Program's specifications. The things that have been documented.	The reality of the product, whether it has been documented of not.
Source of product information.	Specifications.	All descriptions of the product. The generic product type.
Principles of the test specification.	Detailed, repeatable routines.	Targets defined, defects are searched by contextual interpretation, varying tests based on new information and intuition.
Testing practices		
Testing strategy.	Positive testing with well-behaving test cases. Verification of the functions defined for the product. A controlled process of finding defects.	Looking for bad-behaving situations. Verification of all product characteristics. The process of finding defect does not matter.
Relation to test coverage.	There is at least one test for each requirement. Correct emphasis on negative tests. Coverage is systematically monitored.	Coverage does not need to be equal. Things to focus on are learned during testing.
Relation to using the product in testing.	Product is "used".	Product in understood, really tried, broken.
Scope and limits of testing.	Isolation of product areas.	Emphasising interactions between product's areas as a catalyst for problems. Appreciation of randomness.
Importance of repeatability.	Defects are always repeatable. Tests should be repeatable.	Defects are often caused by complex interactions and full repeatability should not be expected. Repeated testing is waste.

4.2.5 Non-functional testing types are different

Traditional testing literature is mostly about functional testing. Other testing types only emerged during the 1990's and sometimes have their own special competence needs. The most relevant types include usability testing, performance testing and security testing. Security testing often utilises the same principles as functional testing (focus on technology, identifying test conditions, test design to show vulnerabilities), but usability testing, as it is based on assessing the relationship between a user and a system, is seen to require special competence. A large portion of that relates to human factors and user interface issues, thus, knowledge of the issues that are under test. Rosenbaum (2008) states the broadness of usability evaluation as follows: "In the United States, what we now consider usability practice has its roots in several disciplines: human factors, cognitive psychology, anthropology, computer science, and technical communication."

Those are the roots. Practical tasks are the leaves. Wania et al. (2006) present some methods used in usability assurance: usability inspection, formal or model-based evaluation, empirical or user-based evaluation (including surveys and questionnaires), walkthroughs, and usability testing. Especially user-based evaluation and usability testing require skills that functional testing does not utilise. Usability testing requires many facilitating skills and people skills that are rarely used in "technical testing".

But still there are many essential similarities with for example usability testing and functional testing, or we could say that modern functional testing has gotten nearer to usability testing as before as the understanding about testing has progressed.

Some examples:

- Prioritization of tests requires understanding about user's processes and goals, which used to be mainly in the domain of usability in the 1990's.
- When designing test cases or doing exploratory testing the tester must have a good understanding about how the users act and what kind of human errors they might make.

Usability testing has evolved into testing of user experience, which has an even broader scope of knowledge domains. Overall, those testing types have such a large body of knowledge that further analysis of it is left outside of this review.

The author made in 2010 (Vuori, 2010) a rough analysis of knowledge needed in various test types. The aim was not to provide a complete an accurate analysis, but to point out that there are differences in the competence requirements. For that, the analysis is interesting.

Table 13. Know-how needs for various test types (Vuori, 2010)

	Context and use						Design and technology					Processes	
Test type	Culture and context of users	Purpose of use of product and business	User tasks, work tasks, processes	Use cases	Fluent use	Different ways to use	Functions to be used	Data flows	Internal architecture	External architecture	Implementation techniques	Activities in the development process / project	Other testing and quality assurance
Functionality testing at the system level	**	***	***	***		***	***	***		***		***	***
Information security testing, analysis of data security risks and analysis of information security	***	***	***	***		***	***	***	***	***	***		
Usability testing and analysis	***	***	***	***	***	***	***			(**)			
Regression testing (system level)		**	***	***	***		***	***		***	*	*	*
Load testing		***	**	***				***	**	***	***		
System integration testing		***	***	***			**	***		***		***	***
Low level integration testing						***		***	***		***	***	***
Unit testing								**	***		***	***	**
Acceptance testing	***	***	***	***	***	***	***	***	(***)	***			

Notes:

*** Extremely important know-how/understanding area

** Moderately important know-how/understanding area

* Little important know-how/understanding area

For this dissertation this provides a view to the knowledge element of testing, but the classification used and the depth of the analysis need to be built upon in later chapters.

4.2.6 Working in a quality assurance framework

Quality assurance in a software development process usually includes activities shown in Figure 19:

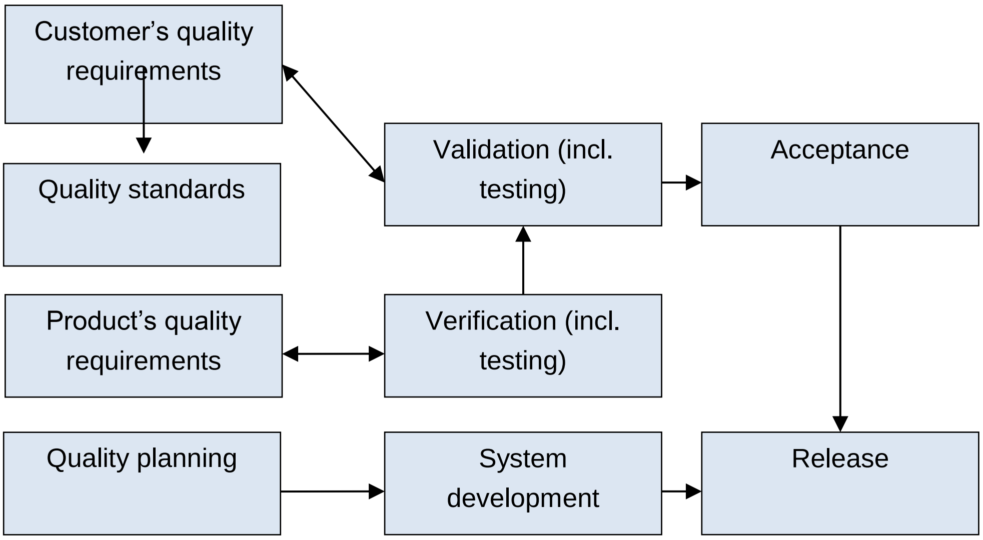

Customer's quality requirements
Quality standards
Validation (incl. testing)
Acceptance
Product's quality requirements
Verification (incl. testing)
Quality planning
System development
Release

Figure 19. A simplified model of quality assurance process.

Essential competence-related areas (for the purposes of this dissertation) include these:

- Quality requirements must be known and the system assessed against them. The requirements may come from the customer or from standards or other such sources.
- Quality standards may be product or process related. The latter give requirements for the assessment – for example the testing methods required.
- The tester must understand the requirements and be able to (in a team) plan the necessary verification and validation tasks.
- Those tasks must be as integrated into the development process as possible, so as not to cause any external delays or need to transfer products or information unnecessarily. Yet, in many domains, a part of the V&V process is outside the development process if independence is required for example due to certification requirements.

4.2.7 A short look into testing standards

Standards have not had a large impact on software testing, but still, new standards document the current understanding about how testing should be done. The main influencing standard has traditionally been IEEE 829 in its year 1998 (IEEE Std 829, 1998) revision – originally the standard was published in 1983. The standard was again revised in 2008 (IEEE, 2008). Comparison of those two gives some insight to the changed understanding about testing during the period between 1998 and 2008 – which was an active era in the development of testing practices.

Some key notices: The viewpoint of the standard has changed from document-centric to process orientation and takes into consideration the development lifecycle. Where the 1998 version did not separate testing levels and had a generic view to for example test plans, the new version documents ideas about the different phases and levels of the total testing process. The new version shows an understanding that testing is done in many processes, not just in development, but also in acquisitions, maintenance and other software lifecycle phases.

A new and important idea in the 2008 standard version is the integrity / risk level of a project and using that as a basis for what testing tasks need to be included in a project.

A central principle is also that the total documentation is something that an organisation can freely choose and many issues can be documented in other project documents (such as a project plan) instead of separate documents. Also, an organisation can work with almost no test documentation for example in agile software development.

The standard mirrors the scope of testing practices and presents some critical competencies: Understanding about risk level of a project and making justified choices in project practices. Those are important skills in any kind or activity as they enable the scaling of practices based on real needs – to assure quality, to handle risks and not to spend money without a real need. The principles are most important in the domains where the criticality of system is high, for example in safety-critical systems, so we need to take a look into that area now.

The normative requirements for the safety of software systems in safety-critical systems are growing. The requirements for the development of those are usually defined in safety standards, such as IEC 61508-3 (2010). What is notable is that in each version of such standards, the requirements grow in number, complexity and in what competencies they require. That means, that the more safety-critical a system is, the more skilled the testers need to be and they also need to be able to utilise such methods that are not widely used. Those include model-based testing and in some cases formal verification. Moreover, the testing units and teams need to be able to manage the vast quantity of normative process and product requirements and to follow the evolution of such requirements. Otherwise, the testing of safety-critical software is rarely dealt with in literature in such a way that would reflect it against what is understood to be good testing. One exception to this is Vuori (2011b), that discusses just that. Otherwise there is a cultural gap between the general testing community and the most safety-critical areas, most notably machinery applications, due to their history as products of mechanical engineering.

During the early 2010s an ISO/IEC testing standard series (also published as IEEE standards) was published in the hope that it might replace the aforementioned standard IEEE 829, consisting of three parts:

- Part 1: Definitions & Vocabulary (ISO/IEC 29119-1:2013).
- Part 2: Test Process (ISO/IEC 29119-2:2013).
- Part 3: Test Documentation (ISO/IEC 29119-3:2013).

Further parts are still under development.

Though used already in draft status in some studies (e.g. Kasurinen, 2011) its adaptation in industry is still unclear. Some characteristics of the standard include[15]:

- The standard series is extremely process and document centric and as such seems to implement ideals from the turn of century. An explanation for that may be that the series was years in the making and during that period the world changed, the needs for testing changed and the understanding about testing evolved. Thus, it may be that the standard was outdated at the time of its publication.
- It presents regulatory requirements as a central motive for testing (the basis for test context being "rules, regulations, standards and laws"), but that does not reflect the situations in most businesses, where business and customer needs always have a priority.
- It shows little understanding the nature and applicability of exploratory testing, which is a living reality in companies, even in safety-critical engineering domains.
- The standard aims at standardizing things that do not appear in reality, such as a role of "test strategist", and if they did, their benefits would be doubtful.
- It is very heavy and unlike the IEEE 829, does not give much help in tailoring its usage in different contexts although it permits tailoring, when sufficient rationale for it is shown.
- Yet, obviously, there are companies and domains where the thinking in this standard is more applicable than in others.

[15] The author spent some time assessing it with some colleagues from TestausOSY when the standard was at draft phase.

For these reasons this standard is not utilized in any role in this thesis.

4.2.8 IEEE’s Software Engineering Competency Model (SWECOM)

In 2014 IEEE published for public review “Software Engineering Competency Model (SWECOM)” (IEEE, 2014). The model is a “standard-like” description of various competences in software engineering. It “describes competencies for software engineers who participate in development of and modifications to software intensive systems. The model describes general skill areas, specific skills, and work activities for each skill, with activities specified at five levels of increasing competency. (…) SWECOM provides a framework that can be used by software developers, new hires, managers, curriculum designers, and staffing and training personnel to assess areas of strength and areas in need of improvement for thirteen software engineering skill areas."

IEEE’s work is always in the engineering domain, representing a systematic approach to software development and quality assurance and in some way. Here we present notes on the model (SWECOM, 2014).

The skill-related model elements that are common in all software engineering are:

- Cognitive skills: Reasoning, analytical skills, problem solving, innovation.
- Behavioural attributes and skills: aptitude, iniative, enthusiasm, work ethic, willingness, trustworthiness, cultural sensitivity, communication skills, team participation skills, technical leadership skills.
- Requisite knowledge, such as educational degrees required for some competence level.
- Related disciplines, such as computer engineering, computer science, general management, mathematics, project management, quality management, and systems engineering.
- Technical skills, divided by software engineering lifecycle, e.g. software requirements skills and software testing skills.

SWECOM has competence levels specified by the role of participation in the engineering process. In the engineering culture, it is typical that companies understand that different roles require different competence. Even in testing, there are defined “entry level” tasks and more demanding tasks require higher competence. In safety-critical systems development, safety standards may require the definition of competence requirements for some tasks. The levels in SWECOM are:

- Technician.
- Entry Level Practitioner.
- Practitioner.
- Technical Leader.

- Senior Software Engineer.

That defines only the status in organization. Another thing is how a person participates in activities, such as validation. The types of participation in SWECOM are following, assisting, participating, leading and creating. For example, planning a validation procedure is clearly a very different task than participating in validation testing.

The tasks that are done by each role in each software engineering lifecycle phase (area) are defined systematically in SWECOM, for example, what kind of tasks a technician should do and what a technical leader should do are defined.

This kind of guideline clearly works on a domain where there are some preconditions present. First, there should be a clear understanding about the tasks that need to be performed, and in the engineering domain, IEEE itself has with its many standards done just that – SWECOM naturally refers to the standards extensively. Second, there needs to be a clear status system in the organisation, where titles define “official competence” and there are enough people fill all tasks with suitable people. In large engineering companies, this is all very familiar and easy. Which makes the third precondition obvious: there must be an *engineering* culture and mind set present.

On the other hand, in other domains and small companies, specifications like SWECOM are exactly what are not needed, because of the needed agility in role selections, processes used and other factors – including the general attitude of keeping the organisational system lean and informal.

4.3 Thinking and testing

4.3.1 Tester's mental models

No matter what the paradigm for testing is, the tester will utilize various mental models in testing. Those are outlined in Figure 20

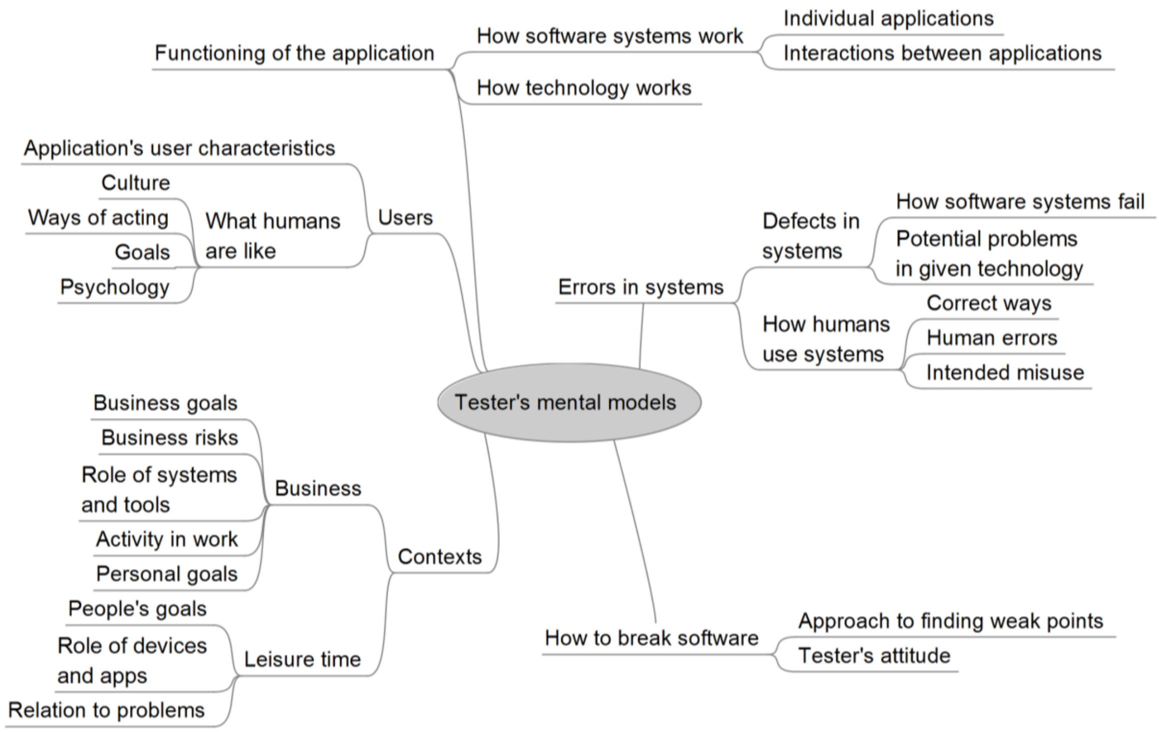


Figure 20. Tester's various mental models (not a complete model).

Some of the models form a "frame" for testing. First, the tester has an understanding about the users of the system. What they are like, how they do things, what goals they have, how a human being's (and the user population's) psychology works. The tester understands the context of the system's use. The nature of the business, including any risks – those are critical to take into consideration in testing. As the systems under test are technical systems, any tester needs to have a mental model of how technical and especially software systems work – how applications work and how they interact in systems.

The tester also has a model of how systems do not work, that is, what kind of defects there are in system, how and why they fail and what kind of potential problems there are in certain technologies. That includes understanding about human issues: how humans use system correctly and incorrectly. It is important to understand the nature of human errors.

Finally, in order to coax the defects to light in testing, a tester must have some “attack models” too. She must have an understanding about how to “break” software, finding its weak points. A part of that is the tester’s attitude of breaking things, but it also includes mental models of demanding test cases and other actions that might test whether the system can handle what it should handle. For an interesting practical collection of attack models for security testing, see Whittaker & Thompson (2004).

4.3.2 Tester’s understanding about how computers and software work

It is surprising how little in tester training it is talked about how computers and software work. The reasons for that may be various. At lower level testing it is assumed that the testers know how software work because otherwise working at that level would be impossible. At higher level the culture has been more about “user’s view” to the system and unfortunately towards positive testing based on ready-made software specifications and test cases. People who are experts in testing or in charge of it in companies may be blind to their own understanding about computing technology.

If a tester is supposed to really challenge the system with negative testing, how is that possible without knowledge about how the system under test works technically – in a generic level, we are not talking about source code here.

That is why Kaner & Fiedler (2014) spend considerable amount of time in the BBST Foundation training to discuss how computers store numbers. That should create a person with a non-technical background the ability to understand how numbers behave and how to find numbers for inputs in tests that may reveal a defect in the software.

Already a decade before that Whittaker (2001) presented areas that need to be understood in order to “break” the software in testing. First, it is important to understand the working of the operating system, including the file system and APIs, including practical issues and problems with files – such as invalid file names and insufficient disk space – and operating system modules (DLLs, libraries) and character sets. Another are is the model of how user input causes things to happen in an application; understanding memory requests, establishing interfaces to databases and libraries; opening, reading and writing of various initialization and working files. And last, the fundamental capabilities of all software: accepting input, producing output, storing data internally in variables and data structures, and performing computations.

Today, it would be also essential to understand the basic functioning of web browsers and browser client applications – what happens in the client and what in the server?

Understanding like that creates a necessary mental model for both good systematic design of test scripts and for good exploratory testing for all testers. Whittaker builds “attack models” based on the system understanding and shows how they can be used to bring up errors.

Obviously, for example a security testing expert is expected to fully understand the security framework of an operating system and the person testing communications protocols to understand the principles of those.

4.3.3 Do testers need to be creative?

Creativity is something that is very often mentioned when talking about the competences of any modern professional. Therefore, we need to discuss that just a little here.

Creativity is often associated to some professional activity, such as designing, advertising or visual arts, where a person creates something that has not existed before and does that in a manner that is not purely systematic – using analysis and synthesis by a “method”. Testing is usually assumed to just assess something that others have produced, or a person has produced herself. Yet, a tester produces something: an approach to testing, testing plans, test cases, strategy and flow for exploratory testing, mental models of where the defects might be. That process is not usually fully systematic. A good tester will use at least lateral thinking – thinking of the system under evaluation from various perspectives, looking in a creative mode for non-obvious things, in different thinking modes. See more about lateral thinking in de Bono (1990 and 1995).

There are certain areas that clearly benefit from creative thinking. Consider, for example, adding tests into a continuous integration system. Literature would propose a systematic analysis of testing tasks that should be added to the test runs, but that is not often done in practice. Mostly the testers ideate about how to design tests sets for various abstraction levels and may document the end result in a way that implies systematic design. Exploratory testing can be creative when it looks for ways to be violent towards the system. Experiments at concept level and design of usability test settings are very creative. Designing a usability or user experience testing environment that sufficiently mimics a real one requires often very creative solutions – similar creativity as an industrial designer uses.

Does the work use or could it benefit from creativity techniques? A team – a development team or a testing team – often uses techniques classified as creativity techniques, at least brainstorming. Usage of mind maps has increased recently and that is one traditional creativity tool, allowing the internal, mental model to be visualised “as it comes” (of course it may support purely systematic action too). Various kinds of analyses, such as reliability analyses and risk analyses, have traditionally been seen as creative, the idea being that the systematic thinking of the designed can only be truly challenged with other thinking patterns. Those analyses use brainstorming, keyword lists etc., which try to remain abstract and open to any new ideas. Especially in safety-critical contexts, performing or participating in assessments of that kind should be a common task for testers.

We need to make a distinction between applying tacit knowledge and creativity. For both, there is a “mystery of creation”. An outsider just sees things to appear as a result of action, but does not see how, nor does the actor necessarily know how. For example, finding potential actions to attack an implementation – how to challenge the designer / programmer – can mostly be considered an application of experience-based tacit knowledge, not creativity. This is somewhat semantics – where do we draw the line between creativity and other mental processes? Here the line would be whether the solutions are novel and not just replication of previous solutions.

It probably isn't beneficial to think of testing as “creative work”, but instead think of it as work where creative thinking and lateral thinking have a big role and should be supported.[16]

4.3.4 “Testerly ways of thinking”

Kaner et al. (2006) divide major categories of thinking in testing as: technical thinking, creative thinking, critical thinking and practical thinking. Here we open up the thinking in testing by using a concept of “testerly ways of thinking”. The term is borrowed from design researcher Nigel Cross, who wrote about designerly ways of knowing and thinking (Cross, 2006). The basis hypothesis here is that we don't exactly know what the testerly way of thinking is, and we should not even care about it too much. This is because the tester's occupation is a new one and the testers' mind sets are still evolving, but they might not be evolving to something that is optimal in the future! That is why a synthesis of what we know about the possible "good ways" of thinking is needed. Then, if the ideas seem fruitful, we can try to make them happen later.

This is important, because the thinking patterns – or hypotheses for such – present the very basic context and ground where any competence operates. So the following would be the most important elements of “testerly ways of thinking”. The text is written in first person as it expresses how a tester could think about herself.

- General: *I am critical. I know that there are problems in information, plans and technology. It is my role to find the problems. I try to understand the viewpoints of others. I am objective and base my opinions on facts or at least some rationale (understanding the nature of tacit knowledge and experience – experts "know"*

[16] As a side note: Rahkamo (2016) has a recent dissertation about creativity where she discusses the concept in the context of athletes, in particular, Finnish Olympic winners – another domain where creativity is not seen a main element of success. For methodology, it is also a well-documented case of Grounded Theory method.

things without knowing why). I am proactive towards problems and defects – I will try to prevent them before they get a chance to exist.

- Thinking toward one's own competence: *I am an expert. I can do things that others cannot, at least as well. I do not know everything, but what I don't know, I can find out. There are new things in every project and I must be fast in learning them. Other people have their special competences and I must understand and respect them.*
- Thinking towards technology: *There is always at least one defect in any system. It just must be found. People use technology in various ways – all of those are unlike my way or the programmer's way. I need to find out those ways.*
- Thinking towards testing as occupation: *Testing is a service occupation. My purpose is to produce information others can act on.*
- Thinking towards business: *I must help the business in any way I can. Business is what pays my salary. I must give information to managers that help them make the right decisions. Because they are busy people, that information must be timed right and be easy to digest.*
- Thinking towards team: *I am a team player. I am in the team to help others, and so are the others to help me. I must help the team in any way I can. I help the team to improve its action.*

4.3.5 Testing as sensemaking

In chapter two it was noted how much of our world is complex, complicated of even chaotic. It is hard to make sense of it. For example, it is difficult to understand the customers' preferences in order to start the development of a new product. People can think about things in so many ways that the developers can't even imagine. There can be so many, perhaps conflicting sources of information that it is hard to know what to trust. In general, it is often assumed that organisations need more approaching sensemaking to support their understanding and thinking. Sensemaking is even a paradigm today and there are many books about it, such as the one by Moore (2011). Its focus is on the actions of defence intelligence organisations, but the situations can be quite similar in many ordinary organisations. Testing can definitely clarify things and its goal is sometimes exactly that.

Examples of this include:

- Proof of concept tests for technology will be carried out to find out whether we can manage and trust a new technology.
- Open ended user experience tests should reveal the users' relationships towards a perhaps disruptive product.
- Testing of minimum viable products (MVP) to make sense of what they need and want.

- Big data analysis of usage logged user actions can reveal patters that we had no idea existed.

Thus testing can clarify us what a product development situation is like, and even turn a situation that seemed chaotic into something that we can think that we will be able to understand sufficiently in order to start product development. So, with testing we can start making better sense of things. After we make more sense of things, we can pick strategies and methods for actions.

4.3.6 Tester’s ethics

Ethics deserve a separate discussion. Ethics is mostly about doing what is “right” and having a sense of responsibility towards others. See more about the theories of it in Encyclopedia of Philosophy (2015). The most widely known and referred ethic principles of ICT professionals are the ACM Software Engineering Code of Ethics and Professional Practice (ACM, 2015). For software testers, ISTQB has included a code of ethics in the Foundation level syllabus (ISTQB. 2011a), influenced greatly by the ACM code, which reads like this:

- Public – Certified software testers shall act consistently with the public interest.
- Client and employer – Certified software testers shall act in a manner that is in the best interests of their client and employer, consistent with the public interest.
- Product – Certified software testers shall ensure that the deliverables they provide (on the products and systems they test) meet the highest professional standards possible.
- Judgment – Certified software testers shall maintain integrity and independence in their professional judgment.
- Management – Certified software test managers and leaders shall subscribe to and promote an ethical approach to the management of software testing.
- Profession – Certified software testers shall advance the integrity and reputation of the profession consistent with the public interest.
- Colleagues – Certified software testers shall be fair to and supportive of their colleagues, and promote cooperation with software developers.
- Self – Certified software testers shall participate in lifelong learning regarding the practice of their profession and shall promote an ethical approach to the practice of the profession.

Principles like this can sometimes be seen as empty wordplay, but they can have a significant meaning though there is no proper research on that. When the author provided basic training of testing to unemployed and those in danger of getting unemployed, and thus seeking a new career in testing (in 2006-2010), he thought that the ethics should be the first thing to train. The principles used in that training are listed in Vuori (2010d).

The reason for the importance of ethics was that testing was at that time seen to be either technical testing in engineering environments or low-skill that mostly just performed tasks according to orders. There was a need to make people understand that one major part of tester’s occupation is to serve others in their information needs, and that requires certain principles that remain mostly the same even if processes and practices vary.

It could be said that there are levels in ethics that also relate to the generations on testing thinking. The first one is related to the systematic testing era. In that the guiding principle is that the tester should do all expected tasks as accurately as possible and as have been agreed with the party that utilises the information. The main ethic is the industrial work ethic: doing work as agreed. The next level is doing all necessary testing so that the testing can be depended on – not the work, but its results. Above that is the ethics of helping people: helping the developers – perhaps the team a tester works in – in its tasks, helping the customers in their goals and helping the business reach its goal by doing things that give most value for that.

Overall, ethics for a basis of principles that produce quality of action be steering individuals and organisations to actions and results that have integrity from the viewpoint of others. At the same time, it offers motivating meanings for work, raising its quality and productivity. On the other hand, deficiencies in ethics have the opposite effect. So, ethics is quality and ethical competence is part of the overall competence of organizations and individuals. What makes this difficult is that ethics are also part of the organisational culture and thus hard to change at the deepest, subconscious level.

4.4 Experts in testing

4.4.1 General

Experts are people who have better knowledge about something. They are “knowledge stores” that can offer knowledge sharing services. We are using somewhat abstract language here on purpose – we need to expose the functions in order to be able to think of something better if needed and if possible.

There are many kinds of expert actors in testing in various types and roles, see Figure 21) and Table 14, where the elements are opened up.

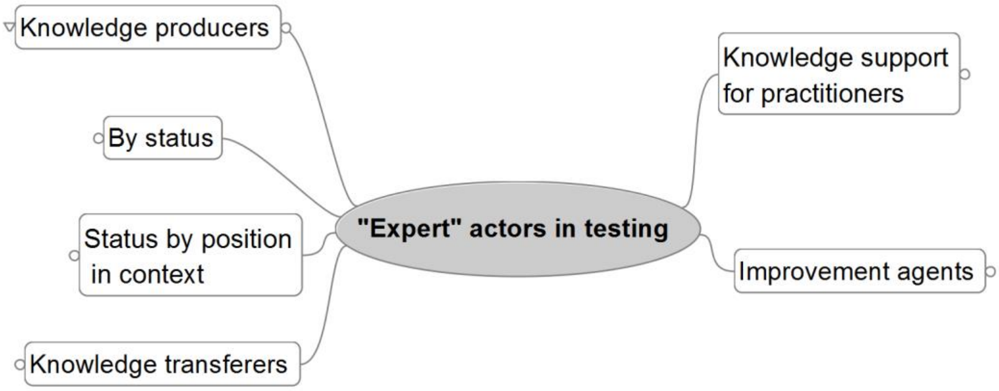

Knowledge producers
By status
Status by position in context
Knowledge transferers
"Expert" actors in testing
Knowledge support for practitioners
Improvement agents

Figure 21. Expert actors in testing.

Table 14. Expert actors in testing.

Main type	Sub-type	Variations
Knowledge producers	Practitioners in companies	
	Consultants	
	Producers of scientific knowledge (researchers)	
Knowledge support for practitioners	(Everyone knowledgeable)	Local context, contact network, social media, ad-hoc search, commonly known experts
	Named experts (in context)	Company's automation expert, official consultant, vendor contact person
	Opinion providers	discussion partners is social media, respected individuals, colleagues, testing tool marketers
Improvement agents	Coaches	For individuals: mentors, masters
		External, internal
		Consulting focus : organisational performance, process / activity, methods, techniques, tools, personal competence

Main type	Sub-type	Variations
By status	Cultural gurus	Of a testing school
	Domain experts	Business area
	Technology experts	Testing technology, product technology
	Generalists	Quality, software engineering, testing
	Methodology experts	Test type
		Trademarked methods, generic methods
	Local experts	
Status by position in context	Sales & marketing	Sales people acting as experts, service account managers
	Research	Research managers
	Company positions	Project manager, unit manager, test manager, chief quality officer (CQO)
Knowledge transferers	Educators	Occupational education, universities
	Trainers	Certification courses, tailored training in companies, public courses
	Conference speakers	Scientific conferences, Professional conferences
	Authors	Authors of books, articles, certification curricula, blog posts, testing magazines, professional journals, scientific papers

There are so many expert types that in this dissertation we only have a possibility to look into just a couple of them, selected by their "perceived cultural importance" as of year 2015.

4.4.2 Competences of educators

Gradually, testing has been brought to educational institutes and has a role especially in software engineering education. Special testing courses are found in many universities and testing is taught embedded in software methodology and programming courses. The author has personal experience of teaching testing in a university since 2011 (and has before that taught testing as a consultant's role since 2004).

The main problem for teachers is that the domain of testing is so dynamic and advances fast that it is hard to know the “hard core” that should be taught and not just something that is hype and will be replaced later. There is no time to wait. If we wait for years for something to be “proven”, it will at that time be old-fashioned and replaced with something else.

For example, the teaching of functional testing has traditionally focused on test cases, but later exploratory testing gained popularity, it was time to assess how that should be taught. Practical questions around that included (now we are thinking about year the 2005 or so):

- Is the exploratory testing a valid approach at all? There was at that time no research to prove that. The positive experiences may for example be caused by the famous Hawthorne effect, where workers are more productive and satisfied no matter what is changed in their workplace. The reason being the attention finally paid to them and their needs.
- As there are no standard or methodology that would suit the industrial culture, in what form should it be taught?
- For which situations should its use be recommended?
- How should it be mixed with more test design based testing?
- How much can the educators trust themselves when making recommendations?

The problem is emphasised by our understanding that testing is not just working with inputs and outputs on a single isolated component, but requires understanding about the context, how people work, what are the risks. Of course, besides functionality, many other aspects should be tested, including user experience and security. Also, the testing is human activity in organisations and projects. Yet again, it is often a technical activity closely integrated with the logistics of the software building and delivery. The body of knowledge has ballooned and it is hard to handle it, especially without falling into some “school” with a narrow viewpoint.

All this means that the knowledge requirements for teachers are quite vast, especially compared with the old approach of teaching testing techniques. Teachers need to continuously learn new things and update their materials. The updating requires time resources that can be sparse. Of course there are various scopes of teaching – teaching the whole “testing” is different than teaching unit testing, but even that has evolved a lot during the years.

One way to look at this is to position the various experts on the dimensions of level of knowledge and the broadness of focus of knowledge. An illustrative presentation of that is in Figure 22.

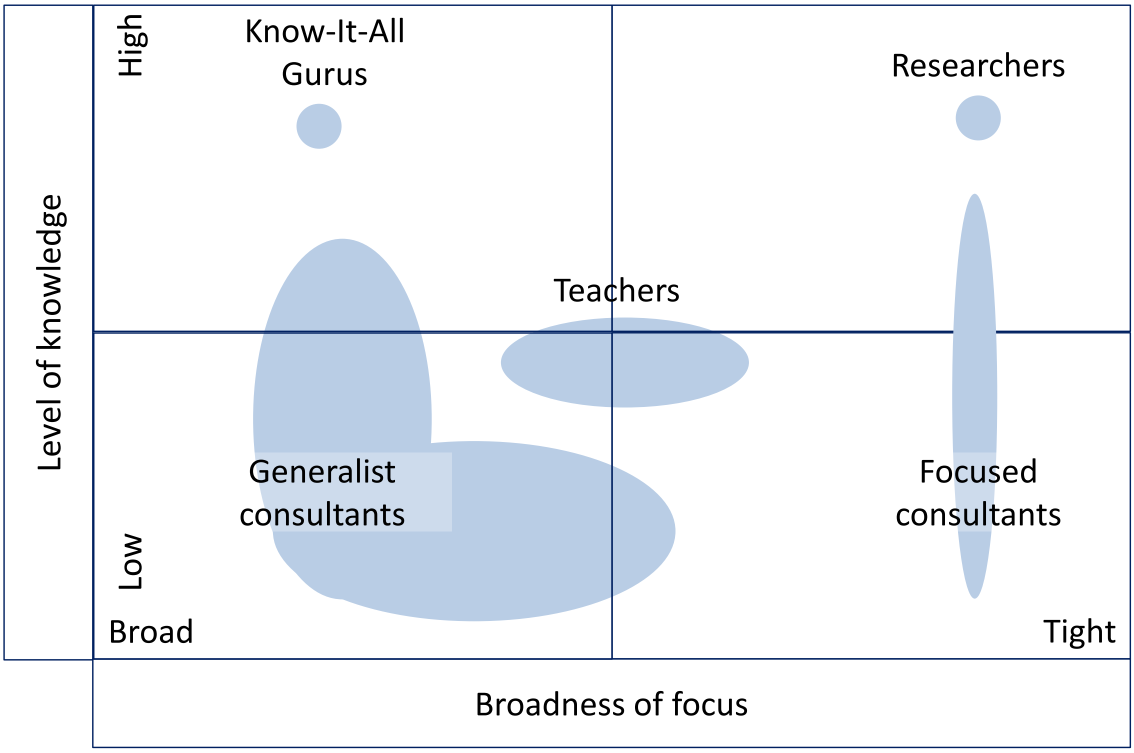


Figure 22. Positioning of some expert types by knowledge (by the author; just illustrative).

Teachers are by necessity somewhere in the middle on both axes, but naturally there are variations. For example, university teachers are often at the same time researchers and thus inclined differently than teachers in other types of schools.

Teachers could of course take the easy road, and just use a certification syllabus in their courses, but that is not something to expect from at least universities.

4.4.3 Competences of researchers

An old train of thought is that researchers learn to learn more and more about less and less. The world of science prefers specialization, because with a very tight focus one can produce results that are accepted to scientific journals and conferences. In many research institutes (definitely in Finnish universities) publishing venues and media are ranked so that only those that have a high-enough rank, are counted when departments are assessed. To get publications into those, really requires high, long-term specialization.

Here we turn into the problems of bringing testing forward by science. Testing can't be reduced to narrow, independent areas, as in the activity system of testing everything is connected and affects everything else. Thus, we cannot expect researchers to give much guidance that helps practitioners in all areas of testing, but just bounded problem areas. It is up to consultants and practitioners themselves to put the pieces together.

Examples:

- Researchers can find new algorithms for model-based testing, but have little to say about what the actual mix of model-based testing and other types of testing should be. They may have complete lack of knowledge about exploratory testing, for example.
- Researchers may study exploratory testing, but fail to separate its forms that are sufficient for industrial use and those that are not.
- Researchers may have the necessary knowledge, but for scientific reasons remain quiet about things outside their core research area.

The days are gone when one could have immediately thought of calling a researcher to give consulting about general testing issues and how to improve a company's practices.

4.4.4 Competences of a consulting experts

Note that industry the term "consultant" is often used to refer to a hired worker, but here we use it in the true consultative meaning: a person who can give advice on how to do things related to testing and quality. A key problem here is how to recognize a capable consultant – who can we trust? The age of social media amplifies the problems – and also possibilities – of finding the most suitable consultant or even a "guru" to listen to. The author analysed this in 2010, because then there really was a problem with this in the Finnish community, and because the author himself has long before that had to think about his competence as a consultant. The analysis resulted in a "checklist" that presents some issues to look hard in the consultant prospects (Vuori, 2010b). The main issues were these:

- Experience and time perspective – experience from before the latest hype.
- Conceptual models and multiple truths – having various explanation models for a phenomenon, understanding theories, the school of thought, awareness of different thinking patterns and expertises.
- Profiling to a way of working and ideas – for example:
  - Being a theoretician vs. a pragmatist.
  - Being a researcher vs. salesperson
  - Emphasising processes and techniques vs. people.
  - Substance consultation vs. process consultation
  - Comprehensiveness vs. developing of one issue.
  - Being a reformer vs. stabiliser of good practices.

  - Being a supporter of universal best practices vs. basing improvements on the needs and possibilities for the context
  - Being systematic vs. operating in agile and intuitive manner,
  - Seeking security vs. being a risk taker (the customer's risk!)
  - Riding the wave vs, being conservative
  - Preferring complexity vs. being a simplifier
  - Hiding information vs. being open.
  - Being a new expert vs. a veteran in the field
  - Being in a role vs. being herself
  - Being sure of most everything vs. allowing uncertainty.
- Knowing the context – what is the expert's context, from which viewpoint does the expert look at things, understanding about the goals of the domain and business, familiarity with the environment, speaking the domain's language.
- Reasoning for claims – is there a rationale to support the expert's claims, does she identify open questions and challenges in her points of view.
- Presenting the limitations of idea X – to what goals and needs is X suitable for, does the expert propose his idea to every situation, does she take into consideration the maturity of the existing activity, does she acknowledge the need to add things to a textbook solution, acknowledging risks and quality assurance activities of the proposed method, integration to higher level activities.
- Mystification and hiding of facts – opening the proposed methods for examination.
- Profiling to some level of competence or maturity level – what is the implicit maturity level where this expert operates?
- Integrity of the professionalism – in addition to skills and knowledge, essential issues are for example attitude, relation to the ones to be advised, dependability and ethics.
- Transfer of the know-how from another context – for example, the different planning professions can look analogous in different domains and sometimes they indeed are it. However, their approach to the substance can be sometimes very different even in a critical way.
- Has all been told? A general problem which is related to knowing is that one imagines that others also know the same basic things. Therefore, one does not recognise a need to tell them all relevant facts.
- Reputation. What do other people think of the expert, what do the people you respect think of her?
- Whose bread do you eat? Are there direct commercial linkages to third parties, such as companies that license methods promoted, certification schemes.
- Customer orientation. When all is said and done… does the expert have the customer's benefit in mind, or just her own?

4.4.5 The problem of guru-centric world view

Consultants and other “gurus” are indeed listed to a lot. One could even say that there are “guru-centric” world views among practitioners. However, there are of course others too. Figure 23 lists some of those.

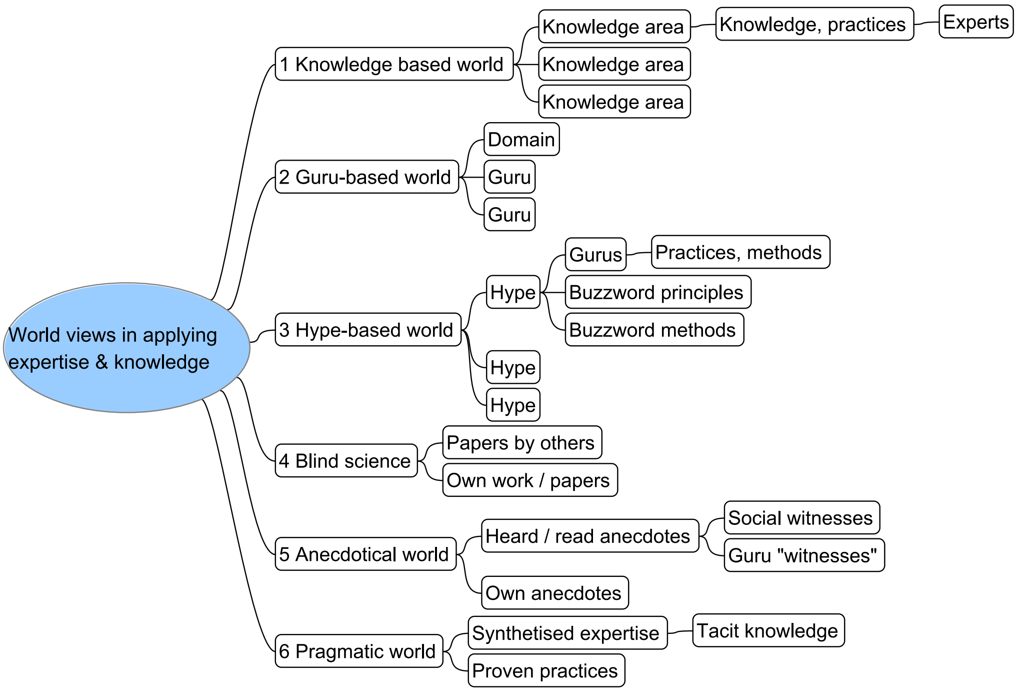


Figure 23. Word views in applying expertise & knowledge.

The “text book” world view is based on recognising knowledge areas in some domain (1). For example, in testing those might be test design, test automation and so on. Then one divides those further, identifying and learning about knowledge and practices related to them. Finally, one notices that there are recognised experts who can provide guidance.

In a guru-based world view (2), people structure the word of knowledge into know gurus and recognise as relevant knowledge what the gurus propose as such; what they present. In the world of social media this seems common, as the gurus advertise things through themselves. The end result is that we don’t think of what we know, but what the guru knows and has said. This is very much related to the hype-based world (3). After all, consulting gurus often ride on the hype and a hyped thing “is” always the best thing, because it is supposed to be the latest advancement.

Some people, most notably researchers, have a scientific view on knowledge. That, when taken too far, becomes blind science (4) where one knows only the subset of everything that has been printed on the scientific papers that the researcher reads. The real world and its diverse contexts have no importance here.

In companies, people still often live without good exposure to solid knowledge. Instead they build their knowledge base by anecdotes (5) that are then generalised – “In my latest project this and this happened, so a general rule is that and that”.

Finally, there are pragmatists (6) who collect the knowledge little by little from experiences that they reflect on. That way they produce solid understanding, but perhaps too focused in their own context.

Of course, the world views are stereotypical (perhaps even anecdotic!), but they point to some serious risks. It is easy to believe in gurus, whose main goal may be economic, aided by market-political tactics. Gurus in the areas of testing are, luckily, less risky than for example management gurus, as their advice can immediately be put into test in the next project. They have also not been analysed that much, but the elements of building “guruness” are similar than in the field of management. From that field, see for example Kantola (2014) as an analysis of the mechanisms of becoming and being a guru.

4.5 Test automation skills

Test automation is a very interesting phenomenon. First we need to look into the philosophy behind in order to really understand the various competences related to it.

4.5.1 Automation as utopia

One element of automation is that in many ways it is a shared utopia. Automation in the first place is the dream of the industrial society. Manual work is gotten rid of, the software is automatically created and automatically tested. This dream is always present in the testing (particularly in the thinking of the old factory school of testing). In the visions, all testing is automatic, nothing else is needed. The automatic tests cover all essential issues and find all the defects. Automation represents progress and competence and it is easy to fall in love with it. It is nice for the managers of a company to tell the guests about it. A machine is in many ways an ideal quality controller as it is an “objective” information machine, neutral and objective, exact and unambiguous. It does not change its opinion and never lies. It remembers everything and repeats everything in the same way every time. Thus a machine is easy to believe. The machine that knows and decides things brings security to humans’ lives. Ideally, a human being is only needed to dust off the machine sometimes.

Automation is the world view based on a mechanistic, structurally complex, but logical machine. Nowadays it has been begun to be noticed that the world is not quite such.

When the human element is removed from the equation, also the best characteristics of the human being are removed, and not just the bad ones, such as unsuitability to very fast or repetitive work and the need to pay the worker’s salary. Standardization and perfect homogeneity are connected to the dream. There is no variation in the way machines work. Also the errors of the automation are systematic and we can get rid of them one at a time, soon reaching perfectness. Automation is technical. We always associate with technical things the idea of best practices of the period, which are brought into use on a shared path of making the world perfect everywhere. It is not rare that there are many defects in the tools in the products created by people who advertise 100 percent test automation. Yet, from the existence of the dreams and utopian ideas we must not deduce that there was not also much sense in automation.

4.5.2 There are many kinds of test automation

Test automation is used as a general term but for example automated unit testing and load testing of the information system are very different things. It is important to analyse the different species of automation so that one can understand what the question is about and to react to each one in a right way. The thought of the core of automatic tests can vary. Some examples: the repetition of planned test cases, trying all variations of an interface, trusting to determinism and planning of details or trusting chance and oncoming possibilities. The abstraction level naturally varies, for example code, functions, abstract actions, keywords, different models of the program among others state machines, the user's actions, use cases and user stories, business processes and statistical use profiles.

Many kinds of things can be automated. The test automation associates with the execution of tests. This holds true for the unit testing and many types of scripted testing. The second basic area is reporting – as “reports” or “radiators". The model based testing advertises the automation of the test design but the test models are not automatically (usually) created at all. It is as important to automate for instance the creation and configuration of test environments or the creation of the test database or other data – in those tasks very much time can be consumed.

4.5.3 Competences related to test automation

The creation of the test automation is serious software engineering work and the maintenance of test scripts (and other assets) is serious and demanding maintenance work. So competence that exceeds the traditional tester's competence is needed. Most testers need the understanding about the basic ideas of automation, targets of its application and restrictions of test automation and the ability to execute tests. Specialized experts are still needed for demanding automating tasks. It is important to create tools which hide difficult details and allow the testers to use the abstraction level characteristic of them, and the mental models and concepts. That is an essential competence area for those who develop the test automation tools.

We could divide the most relevant actor roles like this:

- Generic tester. Is able to run automated tests and do small fixes to them. Note however that not every tester should be able to even fix automated tests. It is just essential that a team has such a composition that rapid dealing with problems in test automation is possible.
- Automation-skilled tester. Can do a variety of testing tasks, both manual and automated. Is able to design and implement automated tests.
- Test automation engineer. A specialist in automating tests and implementing test environments for the test execution.
- Programmer. She does implementation for the automation of own components and helps the testers of the same team do automation for the project.

The main competences for a tester could be divided into:

- Meta-competences: Understanding when test automation is a suitable choice and when not.
- Scripting and modelling skills: Ability to turn tests into traditional scripts or in the case of model-based testing, into models that tests can be generated from.
- Test asset management skills: Management of test scripts and other assets require similar skills as in software development.
- Collaboration skills: Getting the developers to develop software that is testable.

Of course, at the bottom of every testing task are the competences of devising good tests (and other “core” competences). Without good test designs, automated testing is worthless.

4.5.4 Example of automation competences: the difficulties in applying model-based testing

Competence is always a relation to some demands or difficulties and those are the clearest in the context of a new approach or methodology. Model-based testing is one such. The author has observed its development since the early 2000’s and in 2013 made a brief analysis of it which outlines how the challenges of new approaches are not just personal but span also cultural issues. There are many aspects of difficulty in applying model-base testing.

Attitudes and motivation: Modelling is seen as an unnecessary, indirect task – why not just go and do something that provides value. Why create models that test can be generated from when they can be written directly? There is no real perceived feeling that modelling would really work. There are no success stories to build that feeling. It is often said by model-based testing experts that it does not replace other types of testing. If the current types of testing cannot be replaced and they seem sufficient, it makes no sense to add to the palette. After all, testing should be kept as simple as possible.

Aesthetics and desirability of the type of testing and its associated things have a big influence in attitudes. Aesthetics is a larger issue than just visual beauty. The aesthetic aspects of modern testing include things such as: being at and getting to the point – sharp focusing, determination, forcefulness, attempting to break software and simplicity. Yet some model-based approaches have "soft focus" there are elements present but the test points are hard to see and the model seem to cover functionality but lack the look of being attack models that aim at breaking things. Finally, they are complex.

Cultural difficulty: People always cling to the culture and follow it. There is no modelling culture – instead there is perhaps a "hangover" from the times that modelling was more emphasised – the era of UML. As modelling is not used much, there are no modelling memes floating around that could stick to the tester and develop her thinking patterns. There is a lack on supportive interaction from others in teams, companies – as the others may have no idea what is being done and what is essential in it.

Cognitive difficulty: Modelling requires ability to abstract relations between things, to see the flows of events – all of that is not simple. Modelling may require the use of new concepts though that depends on the techniques and tools used. For example, some tools may not use the concept of "state" and instead just make the modeller think of "what is possible". Good modelling requires all the time decisions about what to include and what not, as test models should concentrate on essential issues and be kept concise to be understandable and maintainable.

Practical constructive difficulty – methods and tools: Models need to be developed using tools. This far, many of the tools are often complex and it is hard to get started using them. Some tools may require an architecture that draws much of the tester's attention (for example a two-layered architecture) from the view to the system and its potential problems. There may be requirements for naming conventions and constructs that are not natural. Models may need to be split into parts to manage when they grow larger.

Nature of modelling work: A "good work" should provide rapid feedback to the worker. For example, exploratory testing provides immediate feedback and execution of linear individual test cases can at least form a strong mental image of what could happen when the test is executed. Many model-based activities just form a network of interactions that provide no estimate of what would happen – the result is seen only after the model has been sufficiently developed so that it can be executed.

Lack of training: Lack of training influences many of the types of difficulty. Traditional testing is educated in some schools and is the focus of practically all tester training during their career. Some universities include a possibility to try model-based testing in practice on their testing courses (including TUT starting from fall 2013).

4.6 Relation to software engineer role and skills

4.6.1 Software engineering skills

Traditional software engineering skills are understood to be important for testers whether they use test automation or not. The need for those is increasing, as testing and testers are more integrated in software development teams and daily activities.

The skills include:

- Understanding the principles of software engineering: how software is created, the processes, tools, collaboration etc.
- Understanding about software lifecycle models and the goals of testing in any phases of activities during the lifecycle.
- Understanding the detailed process of software creation, in order to understand why and how the defects come to the system.
- Ability to understand specifications and product documentation. The purpose of requirements in their various forms; the syntax and semantics of the various presentation formats.
- Ability to understand architecture diagrams, in order to understand how the various functionalities are located in the system and to be able to plan their testing.
- Ability to use a version / configuration control system for two purposes: 1) loading the systems under test from the systems for testing and 2) to control the versions and configurations of all test artefacts, including plans, test designs, test cases, test scripts.
- Ability to professionally manage data in the file systems.
- Ability to communicate in a software engineering context – communication medias and tools, what to communicate and when.
- Ability to participate in reviews and lead them.
- Ability to create documents using software engineering conventions and style.
- Ability to read simple scripts, even if fixing them can be left to someone else.
- Understanding the professional programmer[17] and other occupations in the activity system.

Many or the things are cultural in nature and just general knowledge of the activity system and not practical skills of doing something concrete. That is the nature of working in a system – the workings of the system must be understood in order to be able to be a part of the system.

[17] Essential reading about this still are Weinberg (1988) and Weinberg (1998)

4.6.2 The phenomenon of software engineer in test

Starting from around 2010, “software engineer in test” (SET)[18] has been a phenomenon that requires attention. This is a role or even a paradigm that refers to people in development that create test-enabling infrastructure, tools, test designs and so on. The approach is most famous for being used at Google and Microsoft and being thoroughly documented in the book “How Google tests software” (Whittaker et al., 2012). It has raised plenty of discussion in the testing community and is sometimes seen as a preferable approach to many testing contexts.

The role can be seen to be a combination of the traditional roles of test architect, test automation engineer and similar. What is important is the close-knit collaboration in the team and the approach of creating very effective test automation for every feature. As the SETs need to be experts in test infrastructures they may form a pool in a company and get assigned to a project at some suitable milestone. That should depend on the system development lifecycle used.

It is clear that for automation intensive systems this approach gives benefits. The downside and the risk is that when automation is emphasised, there is less “mental” room for other testing approaches, such as manual and tool-assisted exploratory testing. So, roles like this need careful consideration and must not be dominating by default.

4.7 Meta-competences

All the competence models contain only descriptions about the actual competences, but it is important also to think of the meta-competences, the competences that are related to the understanding of the competences and of developing those. Those were already mentioned in as being part of the European competence and qualification principles and now it is time to discuss them a little further.

The author identifies at least the following meta-competences as essential. First the ones that relate to the tester herself:

- Competence reflection. Ability to assess her own competence in a context. Is it suitable for the demands of what should be achieved? Is it sufficient? A professional should always know the limits of her own knowledge and skills.

[18] Alternative terms include “software developer in test” and “developer in test”.

- Competence development. Ability to develop competencies further in many ways – learning in practical work (based on competence reflection), in courses, by reading books, by communicating with experts and peers.
- Communication about competences. One must be able to communicate to others about her competences. What added value could she provide for a project? What knowledge and experience are her opinions based on?
- Sharing competence. Shared knowledge and skill development. Mutual learning in a team.

We also work in collaboration with other people. To be able to work in a team effectively and to know when the others can be relied on in professional tasks and decisions, one must be able to reflect on the competence of others, to identify expertise and to know when that is not present. That understanding helps a tester in understanding when to step up and take an active role in something and when to trust in others to do something.

Some of the competences here could be called "competence awareness" – the term is borrowed from cultural awareness – that is about understanding that competences matter; that different competences are required for different tasks and that collaboration of experts may require personal adjustments to working patterns and communication.

4.8 Gaining competences

4.8.1 Knowledge creation – personally and in an organization

Previously we mentioned knowledge creation and that requires some discussion. The collective knowledge and understanding are elements and basic conditions of good organisational activity. One must remember to externalize information and to share it consciously. The tacit, quiet knowledge which stays hidden is in danger to get not utilized and to get lost.

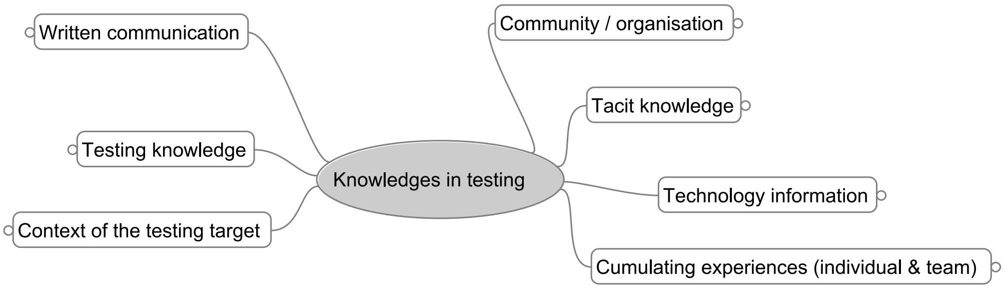


Figure 24. Areas of knowledge in testing.

The areas include knowledge about the following things:

Community / organisation:

- Habits of the domain.
- Role of testing.
- Needs and expectations of different parties.
- Characteristics of systems.
- Quality thinking.
- Shared experiences.
- Activity in projects.
- Work culture.
- Who know and can do something.
- How is information used.

Tacit knowledge:

- Hunch brought by experience.
- What is essential.
- What others like to hear, what is thanked.
- Where the bugs are.
- Interpreting of disturbances and problems.
- Making observations, seeing.
- Raising bugs – what, when.
- Way to do routines.
- Understanding about timing of actions.
- How things are communicated.
- Stereotypes, architypes.
- How team dynamics work.
- What should be said, when and how.
- When things should be planned and when we should be action driven.

Written communication:

- Plans.
- Instructions.
- Reports.
- Bug information.
- Review minutes.
- Literature.
- Internet discussions.
- Email communications.

Technology information:

- Structure and functioning of systems.
- Specifications, standards.
- Typical problems.
- Testability.
- Ways of testing.
- Experts, information sources.

Testing knowledge:

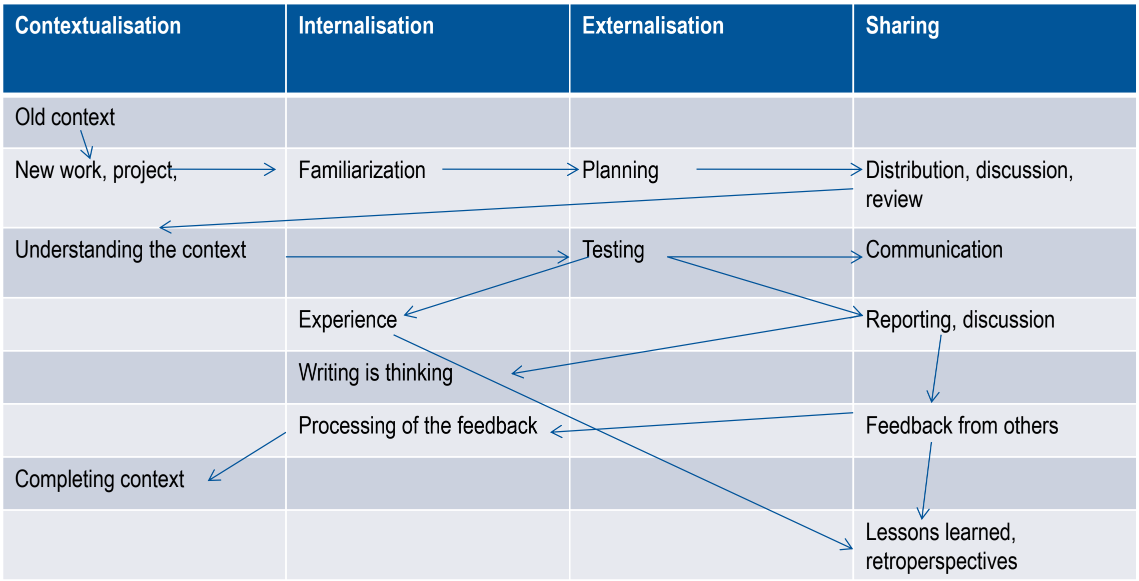
- Principles of testing.
- Testing in different situations.
- Testing in different processes.
- Testing as means of quality assurance.
- Testing near needs and near implementations.
- Testing techniques.
- Using tools.
- Communication.

Context of the testing target:

- Business.
- Activity processes.
- People and organisations in various roles.
- Pressures.
- Perceived problems.
- Usage of applications and systems.
- Elements of quality.
- Risks of activities and technologies.
- Unstated requirements.
- People's collaboration in work and in processes.
- Expectations on testing.

- Cumulating experiences (individual & team):
- What works and what doesn't.
- Collaboration between peoples.
- Actual values and real principles.
- Previous project experiences.
- Where are compromises made.
- Common problems in projects.
- Experiences from technology.
- New observations from new things.
- Realism of new solutions.
- Shared learning.
- Bug knowledge – where. what, why.

To understand the knowledge creation processes, the author devised the model in Figure 25 for practical testing work (originally published in Vuori, 2011c). It is based on the well-known SECI model by Nonaka and Takeuchi (1995), with some alterations – socialization has been converted to “sharing” – this in “contextual localization of terms.



Contextualisation	Internalisation	Externalisation	Sharing
Old context			
New work, project,	Familiarization	Planning	Distribution, discussion, review
Understanding the context		Testing	Communication
	Experience		Reporting, discussion
	Writing is thinking		
	Processing of the feedback		Feedback from others
Completing context			
			Lessons learned, retroperspectives

Figure 25. Knowledge creation process in testing.

The continuous process of changes between the states of the knowledge is the most important thing here. Knowledge needs continuous "movement" to evolve. Internalized knowledge must be externalized so that it can be processed. Tacit knowledge must be externalized so that we can learn from it, abstract it and properly use it in another situation. If we don't share our knowledge with others, they cannot use it and the whole team cannot learn. For example, we must share the knowledge to turn in into shared action, which promotes learning for ourselves and the team. The real fruits of the process are where the gained new understanding adds to our understanding about the context (including the activity system, the products under development), making us more fully able to understand the issues at hand. Some other knowledge transfer processes are presented in Figure 26 and Figure 27.

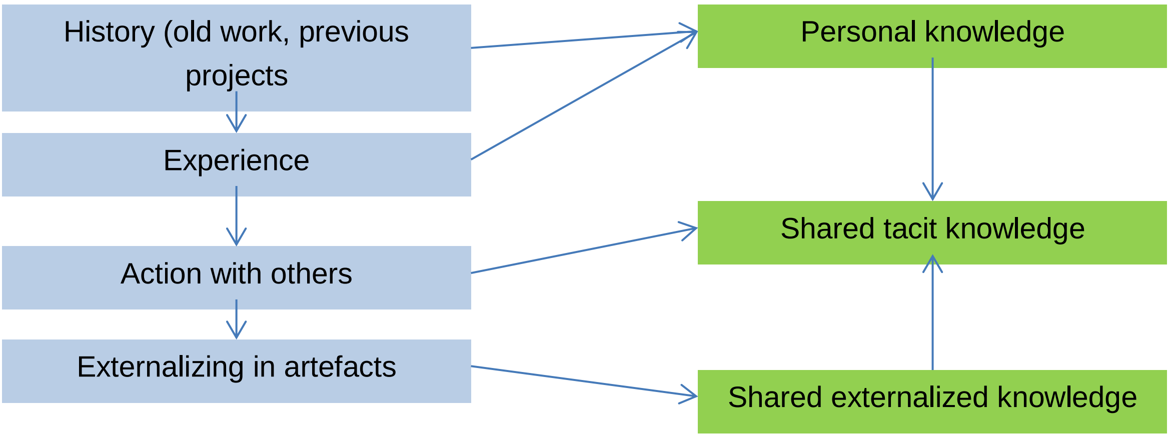

History (old work, previous projects
Experience
Action with others
Externalizing in artefacts
Personal knowledge
Shared tacit knowledge
Shared externalized knowledge

Figure 26. From personal to shared knowledge.

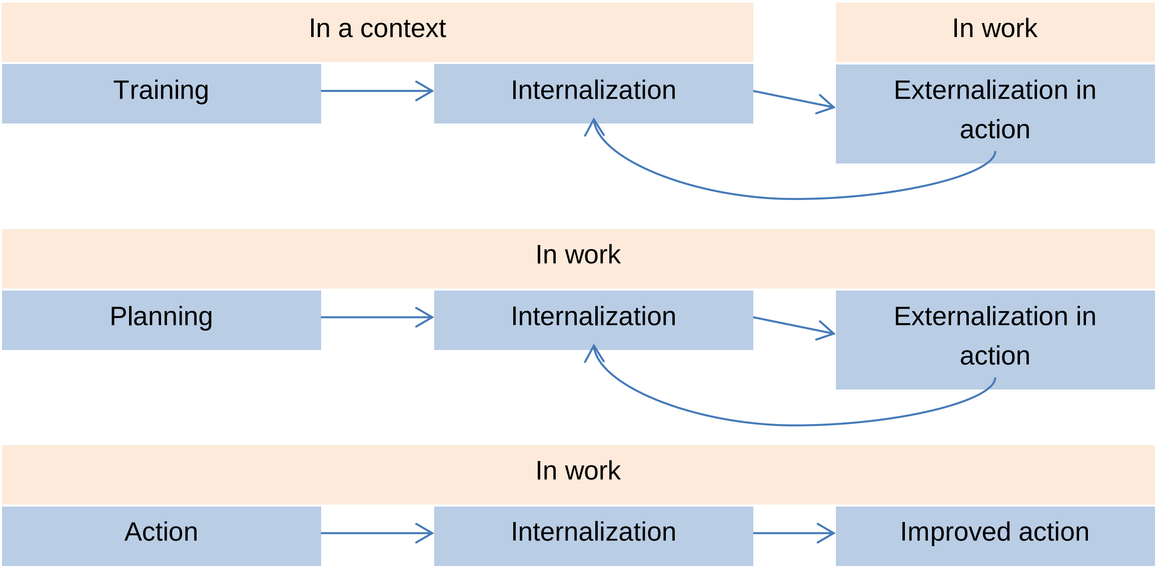


Figure 27. Action in context as an essential means of learning.

Some reflection from this subchapter model to the tester's competences:

- Context is the king and changing the context always requires extra work. It is a shared problem that requires shared knowledge creation.
- The amount of relevant information in testing is very large. We need to have conscious effort in turning that into relevant new knowledge.
- Much of the knowledge is tacit and we need to externalize much of that in order to make it visible, explicit, and an object of action.
- Communication is essential for knowledge creation and testers as team members must be able to communicate orally and in writing about more than just the core issues related to defects.
- The lessons learned and retrospectives used especially in agile developments support not only project and process control, but true shared knowledge creation.

4.8.2 Testing education

Software testing is increasingly taught in academia and obviously what and how it is taught should give an insight to what is considered essential. After all, at least in universities, there should be a solid basis for anything that is taught to the students. That principle is in contrast with companies that provide training to companies. That can be more based on marketing and wishes of the training providers than real value. Astiagarra et al. (2010) analysed the testing education in the USA by making a survey to educational programmes. They noted that generally, testing education is very weak and proposed a new candidate for test engineer education based on two things. First, expert crowd-sourced content, "courseware", (similar to what Kaner has used (Kaner, 2001)) reviewed by experts. This would help with the problem that professors are not usually that experienced with testing. Second, teaching the importance of good bug reports. The latter is often seen as one of the most important skills, as a tester needs to provide information about software errors and anomalies so that their importance is understood and they will be corrected. If that fails, testing has no value.

Students will meet testing in many roles after they graduate and will start their professional career. Thus, the testing education is not meant to people who will have a dedicated testing career. The courses need to provide “a common core” to all students or at least have a division into education that leads to test engineer skills and to testing skills for software developers. Harrison (2010) discusses how the two viewpoints are integrated in a course. It should still be noted that practically all testing courses assume that the students have knowledge on the basics of software production and programming. Yet those do not encompass the whole lifecycle of a system. A critical phase is the acceptance of the system to use by the customer, especially in the case of information systems. Acceptance testing is an important activity in that. Still, there seems to be no evidence of acceptance testing oriented courses that would see the whole process from the viewpoint of acquiring software, especially information systems.

In Finland, testing courses are often found in universities. For example, Tampere University of Technology (Katara et al., 2015) offers a course that has two main parts:

1) Lectures containing the following themes: test cases, testing as a part of a software engineering process (incl. test levels), dynamic testing techniques, defect reporting, measuring software, agile testing, automation and tools, object-oriented testing, testing of information security, static testing, test process improvement.

2) Practical testing assignments, including test planning and execution using more than one paradigm. Those are highly emphasised in the course’s grading too.

This example suggests that practical ability is understood to be most valuable for a tester – of course it is difficult to see any other occupation or activity being different in this regard. So, the main ingredients to competence here are the broad understanding about the core issues in testing and the ability to do some central testing tasks, such as the planning of testing, doing exploratory testing and doing test automation.

One good thing about educational institutes is that courses can be used as a test bed to assess new phenomena in testing. For example, Itkonen (2011) has done comparisons with students about alternative test strategies. Thus, educational institutes can by themselves assess approaches and thus give the needed validation to them before expanding their teaching.

4.8.3 Testing included in various curricula

In the USA, IEEE and ACM have published guidelines for software engineering curriculums for undergraduate students. They are relevant globally and in Finland too, as the guidelines are used in guiding the local curricula development. The first version was published in 2004 (ACM/IEEE Joint Task Force on Computing Curricula, 2004 & Lethbridge et.al. 2006) and the latest version in 2014 (SE2014). The first version of the guidelines was mainly a national effort, but after that the guidelines have been developed in international collaboration using surveys, workshops and feedback sessions in conferences, so now they can be seen as quite "global" guidelines. For example, a survey about the needed improvements on the 2004 version got 477 responses from software engineering educators and practitioners in 42 countries (ACM/IEEE Joint Task Force on Computing Curricula, 2014). It is interesting to take a look into how the latest guidelines see testing and quality.

The guidelines have as a basis a view about the disciplines of software engineering. The same disciplines apply to all areas of software engineering, thus testing and quality assurance too. The main disciplines are computing, engineering, but also mathematics and statistics, psychology and the social sciences and management science have a role.

As the guidelines are positioned in the engineering domain, it makes sense that they reflect the traditional view of testing belonging in the area of verification and validation (V&V). The suggested topics in that area, and suggested number of hours are as follows. Each sub-topic is marked with a target cognitive skill level (knowledge, comprehension or application). Overall hours for V&V are 37.

- V&V terminology and foundations, 5 h: V&V objectives and constraints (knowledge), planning the V&V effort (knowledge), documenting V&V strategy, including tests and other artefacts (application), metrics and measurement (e.g., reliability, usability, and performance) (knowledge), and V&V involvement at different points in the life cycle (knowledge).
- Reviews and static analysis, 9 h: Personal reviews (design, code, etc.) (application), peer reviews (inspections, walkthroughs, etc.) (application), static analysis (common defect detection, checking against, and formal specifications, etc.) (application).
- Testing, 18 h: Unit testing and test-driven development (application), exception handling (testing edge cases and boundary conditions) (application), coverage analysis and structure-based testing (application), black-box functional testing techniques (application), integration testing (comprehension), developing test cases based on use cases and/or user stories (application), testing based on operational profiles (e.g., most-used operations first) (knowledge), system and acceptance testing (application), testing across quality attributes (e.g., usability, security,

compatibility, and accessibility) (application), regression testing (comprehension), testing tools and automation (application), user interface testing (knowledge), usability testing (application, and performance testing (knowledge).

- Problem analysis and reporting, 5 h: Analysing failure reports (comprehension), debugging and fault isolation techniques (application), defect analysis (e.g., identifying product or process root cause for critical defect injection or late detection) (knowledge), and problem tracking (comprehension).

The numbers for suggested hours are very small considering the number of topics aiming at application level skills. This is always a sign of plenty of compromises. Therefore, we should not look into the hours, but the topics. The topics seem to cover the skill areas of V&V quite well. All the modern developments are inside the terse topic names. For example, it is sensible that specific testing techniques are not mentioned at this level, such as model-based testing and it is up to each university to include those as they see they are needed, based on the local conditions. That leaves the possibility of neglecting important new concepts, such as exploratory testing, which is not mentioned in the guidelines. In general, this gives flexibility for the curriculum planners, especially as there is no comprehensive publication defining the body of knowledge of testing. The guidelines use SWEBOK (Bourque & Fairley, 2014) as such, but claiming so does not make a book a representation of a real body of knowledge – in this case, at least for verification and validation SWEBOK is not that.

Yet, some topics seem to be missing, such as test environments or test management. Test environments have not been a big topic in developer centric thinking: just use the workstation and then test elsewhere. Note also that the basis for many topics is laid in other parts of the curriculum. For example, cloud computing and unit testing tools are presented under “computing essentials”, as they should be, and automated testing and continuous integration are presented under “software process”. That is a sensible strategy and followed at least in the author’s university.

The other relevant area is quality. Its suggested 10 hours are divided like this:

- Software quality concepts and culture, 2 h: Definitions of quality (knowledge) , society’s concern for quality (knowledge), the costs and impacts of bad quality (knowledge), a cost of quality model (comprehension), quality attributes for software (e.g., dependability, usability, and safety) (knowledge), and roles of people, processes, methods, tools, and technology (knowledge).
- Process assurance, 4 h: The nature of process assurance (knowledge), quality planning (knowledge), and process assurance techniques (knowledge).
- Product assurance, 4 h: The nature of product assurance (knowledge), distinctions between assurance and V&V (knowledge), quality product models (knowledge), root cause analysis and defect prevention (comprehension), quality product metrics

and measurement (comprehension), and assessment of product quality attributes (e.g., usability, reliability, and availability) (comprehension).

All in all, there are no surprises here. The contents are culturally solid and don't include any hype or silver bullets.

Traditionally, on top of the engineering is the domain of human-computer interaction (HCI), user interface and user experience design (and so on). In the world of curricula, it is positioned in computer science, and guidelines for that are included in the corresponding ACM/IEEE guideline set (ACM/IEEE Joint Task Force on Computing Curricula, 2013). The guideline proposes various HCI courses. HCI/Foundations (4 hours) includes, as the name suggests, the basics of HCI, including usability heuristics and the principles of usability testing. HCI/Designing Interaction (4 hours) does not include testing as a separate topic, but gives background for it as it includes task analysis, handling human/system failure and quantitative evaluations, such as keystroke levels. HCI/Programming Interactive Systems is an elective course and should continue from where the basic programming courses end, leading to the programming of user interfaces and their elements. HCI/User-Centred Design and Testing is also an elective course and very much focused on testing, including:

- Evaluation without users, using both qualitative and quantitative techniques, e.g., walkthroughs, GOMS, expert-based analysis, heuristics, guidelines, and standards.
- Evaluation with users, e.g., observation, think-aloud, interview, survey, experiment.
- Challenges to effective evaluation, e.g., sampling, generalization.
- Reporting the results of evaluations.

This course is not listed in the guidelines as a part of the core of HCI, but by today's knowledge, perhaps it should be. HCI/Human Factors and Security is a course that links security and HCI together and understanding that connection is absolutely critical for both secure design and security assessments. The world of security expertise is currently too focused on technical issues. The topics include: Applied psychology and security policies, security economics, regulatory environments – responsibility, liability and self-determination; organizational vulnerabilities and threats, usability design and security; pretext, impersonation and fraud, e.g., phishing and spear phishing (cross-reference IAS/Threats and Attacks), trust, privacy and deception, biometric authentication (camera, voice), and identity management. All in all, the students of curricula that follow the guidelines, have a great opportunity to develop their quality assurance and testing skills on the domain of user interaction.

Industrial designers are another occupation that has direct links into the design, quality and testing of systems, especially devices and machines. The device concepts, overall architectures and user interfaces are a critical part of any system and cannot be separated from the software system. There are no global guidelines for curricula in this area, so we'll just look at one snapshot from the most important university in Finland

and see what they teach to the students of collaborative and industrial design. That is described in their Study Guide 2015-16 (Aalto University, 2015). The competence area of advanced industrial design includes two courses about prototyping and that is one core element in testing. Prototypes exist solely for evaluation. Also, the design courses ("Experience Driven Design", "Designing for Services") should include testing in some form.

Indeed, the 10 credit point course Interactive Prototyping 1 has 44 hours allocated for generic testing and analysis of prototypes. The similarly sized course Interactive Prototyping 1 has 74hours allocated for user testing and analysis. Those courses give the students the basic skills, which are used in practice in the design courses. Testing clearly is emphasized in the studies, as it should be.

Now, let's turn to a domain that links the development of the systems and their usage in organization. That is the domain of information systems (IS). There is another ACM-produced set of guidelines for that (Joint IS 2010 Curriculum Task Force, 2010). This is the domain where information systems are acquired and maintained, which are built utilising the software engineering competences. There is a great need to perform good acceptance and maintenance testing and to do various kinds of quality related activities, but how does testing and quality show up in this set of guidelines?

First, the description of the scope for information systems includes this text:

> "The activity of developing or acquiring information technology applications for organizational and inter-organizational processes involves projects that define creative and productive use of information technology for transaction processing, data acquisition, communication, coordination, analysis, and decision support. Design, development or acquisition, and implementation techniques, technology, and methodologies are employed. Processes for creating and implementing information systems in organizations incorporate concepts of systems analysis and process design, innovation, quality, human-machine systems, human-machine interfaces, ebusiness [sic] design, socio-technical systems, and change management."

Note the lack of any mention of testing. In the guidelines there is no mention of acceptance testing. Noticing that, it is no wonder that good acceptance testing is still a quite rare activity and not understood well by information system professionals. Of course, again, testing can be included in a core course such as "IT Strategy, Management and Acquisition" or "IT Project Management", of in an elective course such as "Application Development", but the inclusion is up to the course designers. Therefore, participating in a proper testing course would be advisable for all information systems students, as a suitable course should be available in all universities.

Yet, it should be noted that the generic assumptions for information systems professional are such that they should promote good testing, as they include characteristics such as:

- [Information systems] professionals must have strong analytical and critical thinking skills to thrive in a competitive global environment. Students must therefore: Be problem solvers and critical thinkers, use systems concepts for understanding and framing problems, be capable of applying both traditional and new concepts and skills, understand that a system consists of people, procedures, hardware, software, and data within a global environment.
- [Information systems] professionals must exhibit strong ethical principles and have good interpersonal communication and team skills.
- [Information systems] professionals must design and implement information technology solutions that enhance organizational performance. Students must therefore: 1) Possess skills in understanding and modelling organizational processes and data, defining and implementing technical and process solutions, managing projects, and integrating systems within and across organizations. 2) Be fluent in techniques for acquiring, converting, transmitting, and storing data and information, including those related to data quality. 3) Focus on the application of information technology in helping individuals, groups, and organizations achieve their goals within a competitive global environment.

Those characteristics, combined with basic engineering competences, allow the IS professionals to recognize the need and opportunities for testing and to utilize the testing competences of software engineering professionals. After all, modern organisations benefit from diversity in viewpoints and competences.

4.8.4 Training in industry and tester certification

Currently, most of the people involved in testing gain their skills by experience and by training courses. The training courses are often modelled by matching generic testing knowhow with the perceived needs of a company.

Another type of training is based on the certification syllabi, such as the already mentioned ISTQB. The goals can include improving personnel’s competence, but also to give proof that the testing capability of the whole organisation has improved – this is important for example in subcontractors or testing service providers. When a portion of testers have a certificate, it can be perceived as a “proof” of the organisation’s capability.

Now, when we are talking about real competence, the critical question is: can we find any proof from literature for, or any analysis of, the certification trainings having a real impact on competence? Unfortunately, there seems to be no reliable research showing that.

Of course, such questions may have alternative answers based on the viewpoint of the person who is answering. There are also other arrangements than “official” certifications that aim to the same goal. We have already mentioned the BBST course by Kaner. He is one who would like to have a meaningful certification system and as his latest (as of March 2014) such activity, wrote a proposal for an advanced certification in software testing (Kaner, 2014).

Its main idea is to base the certification in various kinds of evidence of competence, not just in passing an exam. The evidences in the first proposal include:

- Education (Academic).
- Education (Practical) – such as testing courses.
- Examination ("The Certified Tester should have successfully completed a proctored, advanced, examination in software testing", "ISTQB Advanced or Expert exam might [qualify]. Similarly, BBST: Foundations would not qualify but BBST: Bug Advocacy might and BBST: Domain Testing definitely should).
- Professional Achievement – publications, honours.
- References – at least three letters of endorsement.
- Professional Experience.
- Continuing Education.
- Code of Ethics – "the candidates must agree to abide by a specific Code of Ethics, such as the ACM code".

This has some combined characteristics of a current certification system (at an advanced level), and a CV and written references (the referees are sometimes listed in CVs too). One person noticed in the discussion on Kaner's site that this resembles what the LinkedIn social media service provides currently. However, the proposal was just a starting point for discussion. The most important idea is that certification systems can advance and find ways to provide more validity and benefits than what they currently offer.

4.9 Testing in an organisation

4.9.1 Elements of an organisation

An organisation is obviously something that we should have a model for, as we need to in suitable abstraction level understand what this entity where people work consists of. For the purposes of this dissertation we use a practical mind-map of "things" in an organisation as the model. It is shown in Figure 28.

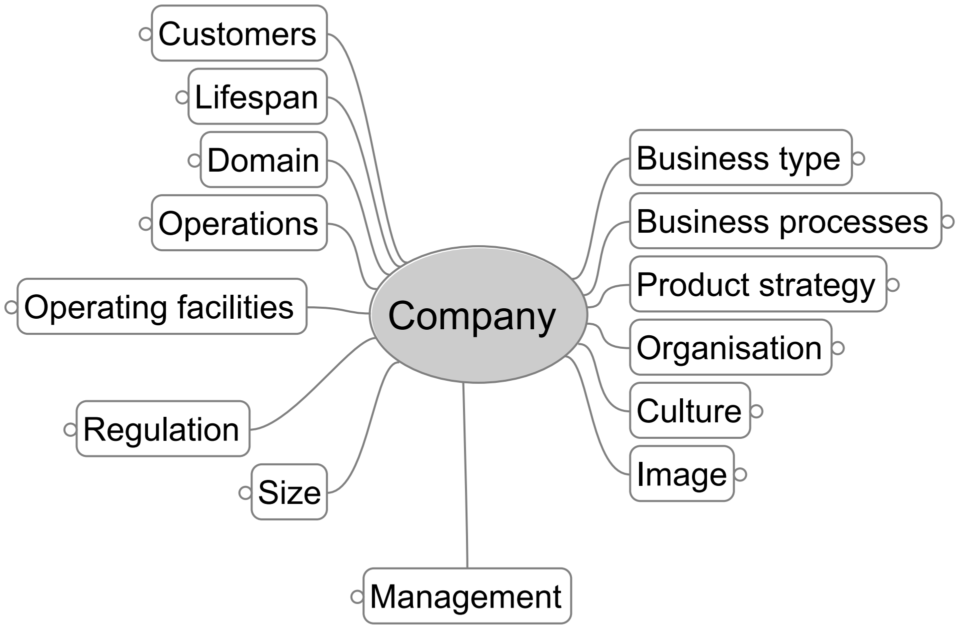


Figure 28. Main elements of a company.

Essential competence-related areas (for the purposes of this dissertation) include:

- Different product types require different knowledge about the user context and technology. Also, they may require different verification and validation practices (perhaps for certification purposes) and using different methods.
- In any regulated context, one must understand the regulations (such as safety standards) and be able to work in a way that satisfies the standards.
- Different organisation models require different collaboration and communication skills. Globally distributed activity is very different to working in the same room.
- In different phases of the lifecycle of a company, the critical needs for testing and tester may vary. In a small startup, a tester must be more holistic and have a sense

of responsibility and do more varying things than in a larger, experienced company, where typically roles and responsibilities are more strictly defined.

An organisation has various levels: The whole organisation, regionally distributed parts of it, units, which may have different purposes, groups and teams, and sub-teams.

The organisation provides a context for the activity of people. In a formal sense, we may refer to it as the activity system, which not only contains the people and processes, but various other things, such as purpose – a formal purpose and a perceived purpose, culture, management style, values, style of collaborating and co-working, rules and norms, communication and the technology and tools available.

If the purpose of the organisation is to develop products and systems, the development-related activities are the ones where testing and quality assurance related practices are mostly used. For other type organisations, those activities may more relate to the acquiring of systems and testing might be mostly acceptance testing. Those are very different things and testing may have very different goals.

In the context of product development, testing needs to provide information about the quality of the system under development for all stakeholders: what is the level of quality, where the problem areas are and how the development progresses – ultimately giving guidance to the decisions about releasing the product to the market or the customers. Part of that information is how the system fulfils any mandatory requirements, such as the ones given in mandatory standards.

In the context of acquiring systems, testing will need to provide information about how the system meets the practical needs of the users and the organisation in general. Is it “good enough” to be used? Does it have problems? Does it work like was expected from contracts, plans and specification etc.? Does it have defects that the supplier should correct before being paid?

Another thing to consider is the generic working style of an organisation. We have previously referred to the schools of testing and those by Kaner (2006a) clearly imply a dominant style of organisation, not only testing. Let’s reflect on how those imply an organisational style.

Table 15. Reflection of testing schools by Kaner (2006a) and corresponding organisational style.

Testing school	Corresponding organisation style
Factory school, which emphasises reduction of testing tasks to routines that can be automated or delegated to cheap labour.	A factory-styled organisation that emphasises efficiency above everything. “Software factories” are an example of those. Testing service providers are sometimes of this style.
Control school, which emphasises standards and processes that enforce or rely heavily on standards.	Many organisations are of this style in industry and in the public sector.
Test-driven school, which emphasises code-focused testing, which is done by programmers.	(No clear organisational correspondence)
Analytical school, which emphasises analytical methods for assessing the quality of the software, including improvement of testability by improved precision of specifications and many types of modelling.	A tradition in “quality” oriented organisations, such as those that rely on “lean” thinking. They may also be biased to “control school”.
Context-drive school: emphasis on adapting to the circumstances under which the product is developed and used.	New agile organisations and lean startups are like that. Companies that produce software products for some distinct domain may also go into this category.

The descriptions above are just illustrative examples. What this really means in practice, that the organisational style very much reflects of the testing and QA practices by: providing a general style of doing things, reflecting on how the organisation sees the “good way” of actions, and providing a general framework for identifying competences – how we do things results in how we see the essential skills and knowledge.

4.9.2 High reliability and high innovation organisations

Organisations can (with some rough stereotyping be divided into "high reliability organisations" (see Weick & Sutcliffe, 2007, for a book length discussion about such organisations) and "high innovation organisations". The latter is discussed today in almost all business books, but the high reliability style needs discussion. High reliability organisations work in domains where the process simply must not fail. Characteristic for such organisations are preoccupation with failure, reluctance to simplify ideas about systems and operations, sensitivity to operations and readiness for the unexpected, and commitment to resilience. Organisations of that kind constantly search for the possibility of failure, are open about process failures (as finding them prevents the failure of the mission) and respect competence over status. For example, any data centres should operate in the high reliability mode, as well as any deployment teams.

Engineering units, especially ones that develop safety-critical systems, are expected to be high reliability organisations, as the essence of engineering is non-failure. However, engineers need to also provide innovation, so the teams usually work in a dualistic mindset, partly innovating, partly ensuring reliability and safety. Innovative product development can be geared towards high innovation mode, but even then there needs to be aspects of high reliability presents.

This is where the testers come in. In high reliability organisations, the testers relentlessly search for possibility of failure, which perfectly matches the mindset or the whole organisation. In high innovation organisations, the testers produce a counterforce for innovation that keeps things sufficiently reliable, letting others safely carry on innovating. But the testers also produce information about successes, not failures.

Organisations are not indivisible units, but they have various units and sub units (teams are such). That makes it possible to have the two mindsets existing in the same company. For example, there can be teams more focused on innovation that produce work for a more reliability oriented implementation teams. The implementation teams can also be innovative, but a thoroughly reliability-oriented phase in the process looks after what they produce (that is the role of traditional QA).

The testing of innovations needs to be two-fold. The role of testing is to reflect on ideas, plans and innovative designs and implementations, that reflection must point out any flaws in them and any good characteristics too. That is what for example usability and user experience testing do today. The thing to keep in mind is that testing needs to consider the overall mode of the organisation and either support it by joining the mode, or support by giving it a "safety net", but also providing direct boost for innovation by pointing out good characteristics, elements that could turn inventions into real innovations. Note that high reliability organisations need innovation too, but the activity domains for making those need to be more separated from the domains that the core activity is done in.

4.9.3 Project skills

Testing is most often done in a project-like context – a development project, acquisition project, a testing project or something else. It is clear that testers, just like everyone else, need the competences related to that. In traditional competence classifications they have usually been named as “general” or “supporting” skills in relation to the key occupational skills.

They include:

- General understanding about projects, their principles and ways of action.
- Practical skills, such as project communications, participating in meetings.
- Project planning, project management, reporting skills.
- Adhering to agreed processes and practices.
- Collaboration skills in teamwork.
- Team leading skills.
- Cultural skills – how to work with different nationalities and representatives of different cultures.
- Problem solving skills.
- Openness about problems.
- Data management.
- Attitude towards work, such as discipline and values towards quality.
- Negotiating skills.
- Reliability, trustworthiness.

These are such generic and sometimes large issues that they are mostly left outside of the scope of this dissertation – and discussion of them would take far too much space. For example, cultural skills have been a big issue and are on area of development in many organisations. The whole personnel have often received training on it from the top management to the tester and many books have been published about it, for example Moll (2012). We will later discuss some of the most relevant competences listed in a more focused setting.

4.9.4 Tester as a team member

Testing has traditionally been divided into two types: testing in development and quality assurance testing. The testing in development obviously supports development and has often been agile in nature – but also been barely sufficient. The quality assurance testing has often been seen as unnecessary, bureaucratic and a general hindrance to business. Even testers do not like that kind of mode, but still it is essential for many kinds of products – the more critical they are, the more essential is a clearly QA oriented activity. Sometimes the separate QA activities may be required by a safety standard, so they are mandatory. Today, it is clear that testing should be integrated into development, but still it often should have a different emphasis than other activities. In the activity, various dynamics are needed that keep the overall system in shape. That is because we live in the real world where people (in their roles) have pressures and personal goals. Project and production managers want a simple flow and an efficient execution of tasks and deliveries, done when expected. Developers mostly just want a task out of their hands and move to the next one. Sales just want to sell. All these are – by necessity – not that interested in quality and therefore a balancing force is needed. That is the testing and testers. Testing as such is just a tool, but the people who do the testing make the difference.

- They have the role and the means to warn about product risks and potential problems in customer satisfaction.
- They have the experience that helps them warn about problems in new technical solutions.
- Generally, the testers can greatly help a team or an organization to “not shoot themselves in the foot”.
- They have the quality strongly present in their mental models and can serve as the team’s leader in that.

So, there should be dynamism of supporting the others in the “extended team” fully and helping others reflect on the results and providing critique as needed. Perhaps we could visualise this with the six thinking hats analogy by de Bono (1985). Perhaps due to de Bono’s influence, it is common to think of people in different roles and situations wearing different “hats”. In this case, the hats might be:

- The white hat when delivering quality information as such. Just the facts.
- The yellow hat, for optimism, when starting projects and planning things in a team and thinking about providing value to the customer.
- The black hat when finding potential problems or risks or assessing the product for release.
- The red hat, for intuition, when in general collaboration, in any issues.
- The green, for creativity, when planning testing in general or in detail, in the pressures of everyday things and in new situations.

- The blue hat is for controlling the whole process, for reflecting about oneself and the team.

Ability to this kind of dynamism is an essential skill and something that should be consciously developed.

4.10 Working in ecosystems and professional networks

4.10.1 Testing ecosystem

The testing world is also an ecosystem, presented in Figure 29.

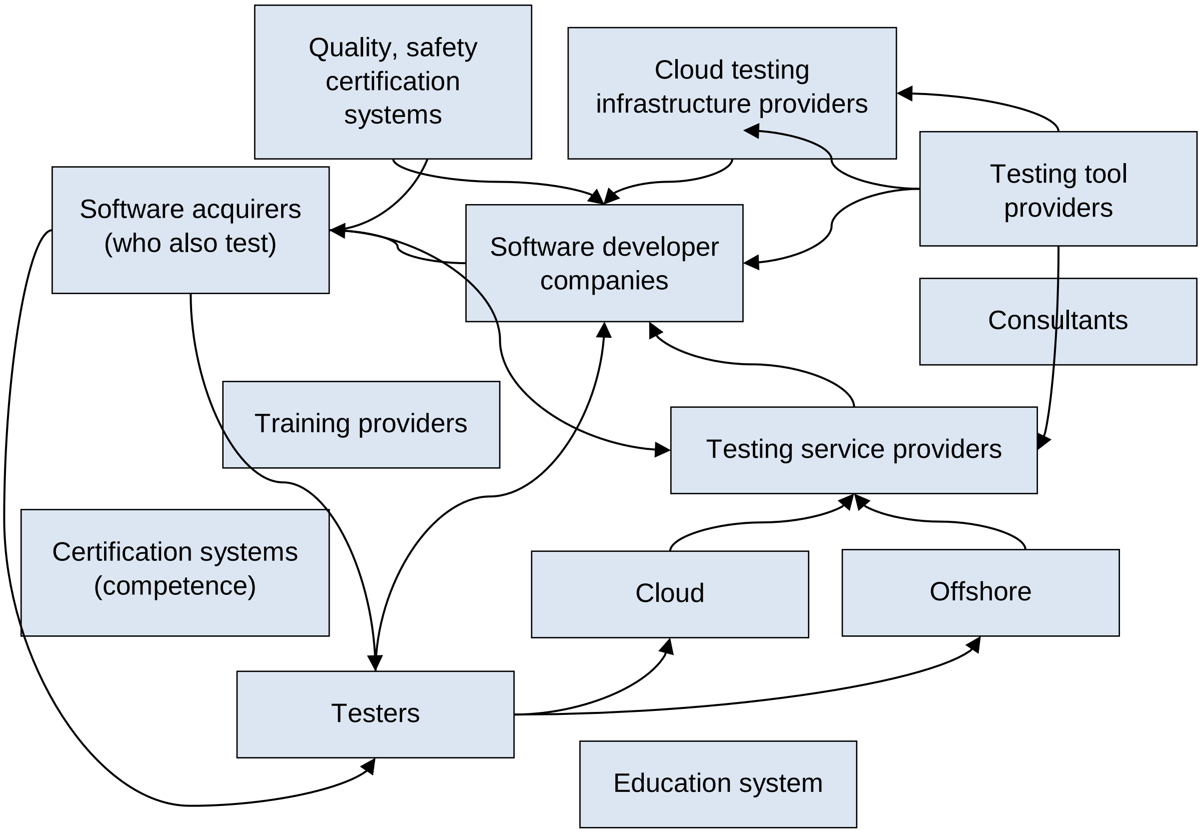

Quality, safety certification systems
Cloud testing infrastructure providers
Testing tool providers
Software acquirers (who also test)
Software developer companies
Consultants
Training providers
Testing service providers
Certification systems (competence)
Cloud
Offshore
Testers
Education system

Figure 29. A model of a testing ecosystem.

Essential competence-related areas (for the purposes of this dissertation) include:

- In the centre of it are the software development companies that execute a software development process. They need to have in their disposal all the necessary competences – and the ability to find the competences.

- Good testers can be employed by any means – in-house or by a testing service provider, either close-by in the same country, "off-shored" in another country or perhaps in a global "human cloud".
- Testers gain their competence by education and training (usually though their employer), and of course by experience.
- Certification systems provide "proof" of the competence and are closely linked with training providers who give courses that lead to readiness for the certificate exam.

4.10.2 Platform ecosystems

Recently, ecosystems have been under much discussion due to the competition in the mobile phone industry. An ecosystem consists of many players, each of which has a certain role – someone owns the brand, some players develop applications, someone arranges a store to sell them, someone provides training and so on. Of course, similar ecosystems exist everywhere, also in the field of information systems. Because ecosystems seem to be such an important concept and companies explicitly define themselves by their role in an ecosystem, we need to provide some kind of model for them, in order to be able to assess whether ecosystems might have some influence on testing and QA. A model of an ecosystem is presented in Figure 30.

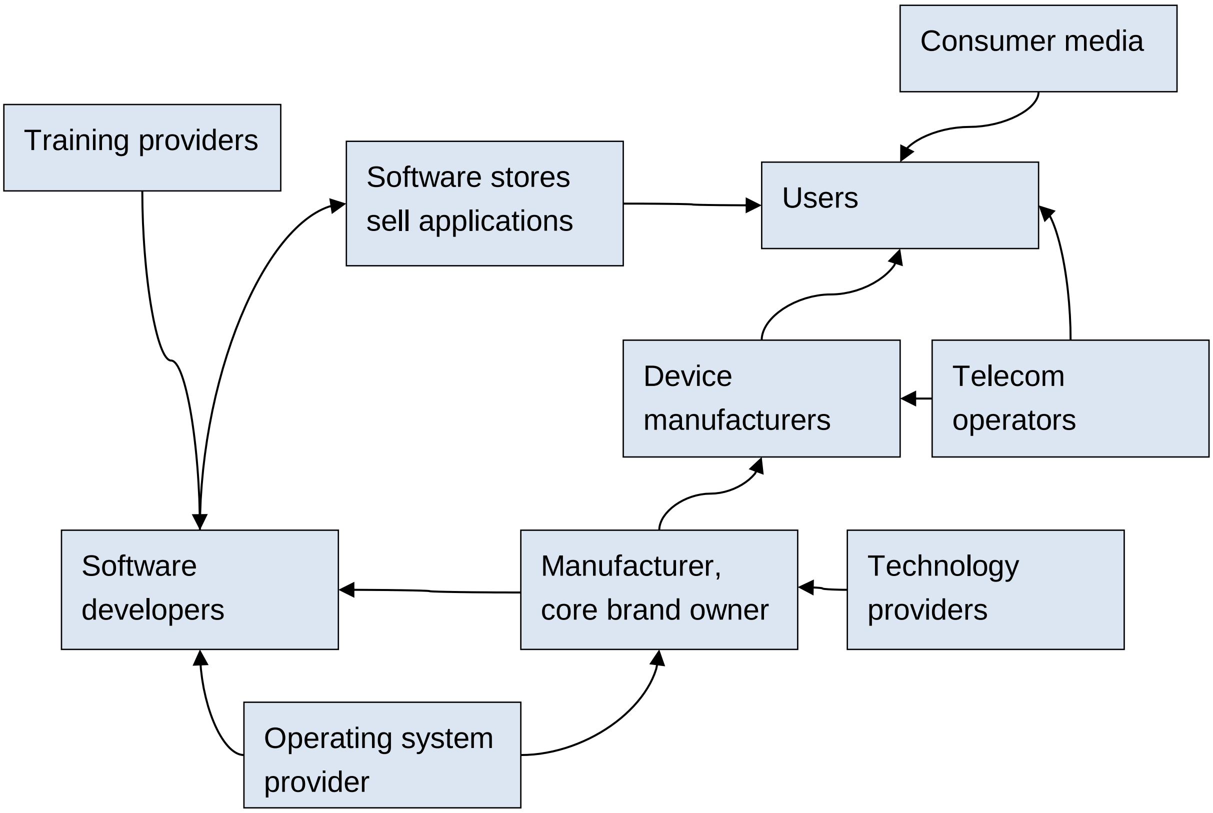


Figure 30. A model of an ICT ecosystem around a manufacturer in a mobile device cluster.

Essential competence-related areas (for the purposes of this dissertation) include:

- Application stores have some acceptance criteria for products. Testers must be able to test against those in an effective manner.
- Each ecosystem has its own technology in products and in development, and because of that, also for testing. Testers must understand the technology and be skilled with the testing tools.
- Ecosystems change. When a manufacturer changes the operating system of the devices, it has serious effects on all parties. There is a lot to learn for testers. That implies that lots of the testers' know-how should be platform independent and that the testers need to be fast learners for the case when big changes happen.
- For any serious ecosystem / platform, there are lots of information and lots of experts available, which makes working and learning easier.

4.10.3 Tester communities

In Finland, communities are important. Most notably, testing has a competence community TestausOSY, Finnish Association of Software Testing (FAST), which is neutral towards any testing paradigms. The author has been a member of its steering group since 2004. In Figure 31 we present the elements of such a community.

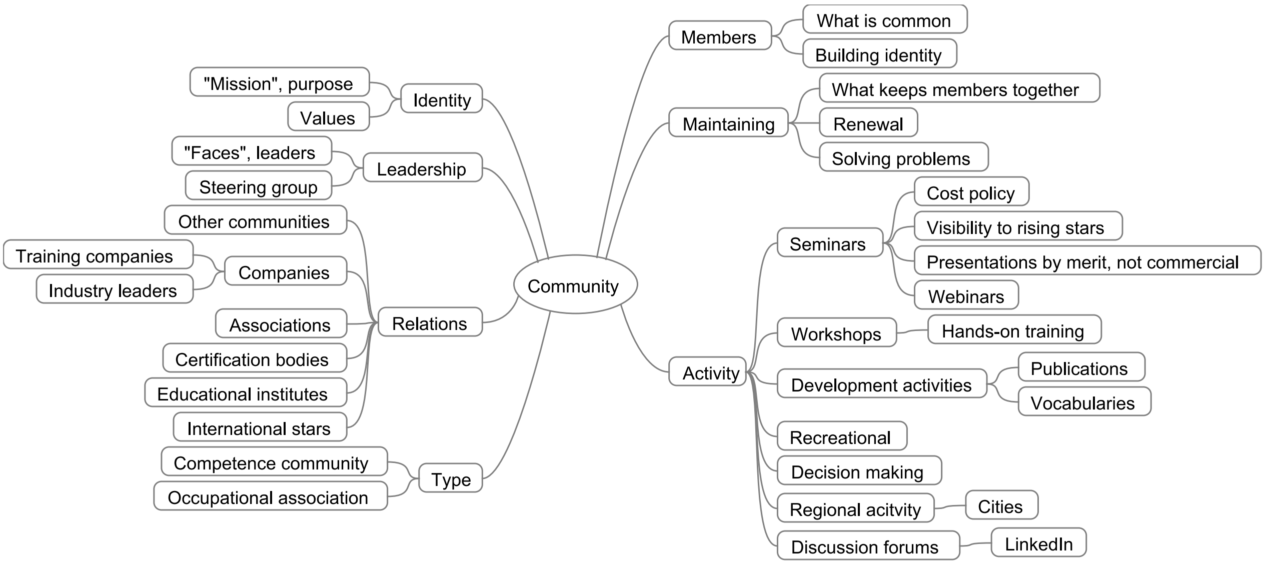


Figure 31. Elements of a community – such as TestausOSY-FAST.

Essential competence-related areas (for the purposes of this dissertation) include:

- Maintaining the community. Every community has a lifespan and is in constant danger or dying.
- The challenges depend on the age of the community and the craft. For example, for TestausOSY the challenges in the past were building the identity of the testers and

to share the basic understanding about testing. Today (2013) the challenges are in shared adjustment to new types of testing (such as exploratory testing and advanced test automation) and to the changing company cultures – the working environments are not centred around large corporations and their culture anymore.

- To build and share competence, the community needs activities. There needs to be seminars and webinars that are open to the members and where the lectures are chosen by merit and importance and not by commercial factors.
- A side effect of that is that it gives “boost” to local experts, thus giving faces to competence and examples to others about how good they can become too, when they learn their craft further.

4.11 Passion for testing

People only do good work if that like what they do. Some even claim that a passion for work is essential. Hamel (2007) proposes that in the future a competence profile of workers should consist of (sum 100 %): passion 35 %, creativity 25 %, initiative 20 %, intelligence 15 %, carefulness 5 % and obedience 0 %.

Definitely, people will do good work if they like it, but innovation and development of the work greatly benefits from passion. In the long run, Finnish expert work only succeeds, it we support this side of the equation and not the rational contents of work. The author saw this as an essential issue in 2010 (Vuori, 2010c) and analysed some reasons why a person could develop passion towards testing. Note that this is done purely for illustrative and motivational purposes.

1 Understanding technology, devices and systems

- Curiosity, how things work and what all they can tolerate.
- Understanding of quality experimentally, with one’s own hands and brain.
- Getting to know all kinds of products.
- Intellectual hobby, play.
- Challenging designers – and winning.

2 Making a better world

- Better products for humans.
- Occupational safety and product safety (for some particular products).
- (Special issues depending on the systems under test).

3 Will for progress

- Safety of new technologies (nuclear power).
- The success of missions (the conquest of space, air superiority).
- Management off new complex technology.
- Technology promotion.

4 Professional identity

- Sense of responsibility.
- Doing work that has a positive impact.
- Helping software developers.
- Helping customers.
- Being an end-user advocate.
- Success as a team, together.
- Perfectionism (of technology).

5 Quality aesthetics

- Aesthetics of error-free systems.
- Technological perfection.
- Will of making the world work.
- Will to control one's own world.

6 A will to be successful

- To be good at something, to be the best, to stand out from others.
- Making money.
- Business improvement.
- Success of the products.
- Joining a winning product development.
- Leaving a positive imprint (even though most would not notice it).

Things like that always something that we need to look after when forming new work profiles for people.

4.12 Paths to competence

4.12.1 model for changing competence needs

As we are talking about changing competence needs, we need to have a model that presents some of the mechanisms. For that we can again turn to the activity theory. When we consider that competence is used in any kind of activity system, we can see a transformation from one state of the activity system to another state will necessarily require changes in competence for the system to be in balance, if there are changes in other elements of the system. This is usually the premise of the activity theory: when there are changes in system elements, we need to change other elements to regain balance.

So, we can call any changes in activity system elements “drivers” for changes in competence.

We really need to stay at an abstract level at this point because of the diversity of the various activity systems and contexts. Even the figure below, Figure 32 , shows just one transformation.

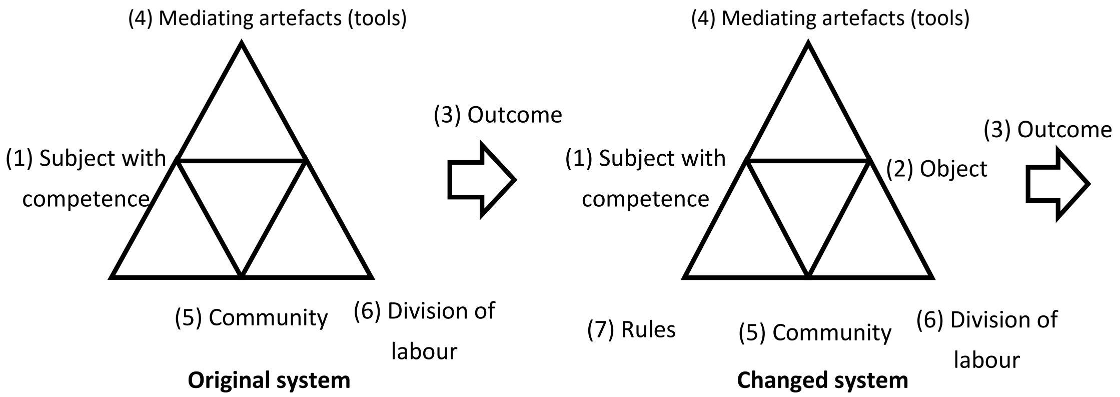


Figure 32. Changes in activity system cause a need for changes in subjects.

This is based on the supposition that the original system was in balanced. That is, however, something that cannot be accepted, as there is a global understanding about lacking quality and thus, efforts in producing quality and improving quality need to be made. So, the activity system is usually in unbalance to start with and changes may turn it into more unbalanced.

For an example, some usual changes in the activity system include the following:

- Object (2) – the system under development and test changes. Technology may change, as well the users of the system. That is reflects into the technical competencies and domain competencies of the testers (1).
- New technology may require changes in tools (1) and division of labour (6) – there may be special tasks that require expertise of even an external consultant.
- The changing of processes changes how the community (5) works and even the rules of the organisation (7) and obviously division of labour.
- Moving to another domain may change the rules (7) of activity. For example, on a safety-critical domain there may be requirements of strictness of action that would be unheard of in another domain.

- More skilled testers (1) can use more advanced methods and tools (4), whereas the lack of skills may work the other way around and even cause limitations to the product technology (2).

4.12.2 Learning paths related to career paths

First, learning by doing has been the most common path. This is very much related to how people enter into testing. In many organisations, testing has been seen as an entry job. Both the employers and the employees have seen testing as something that just about anyone can do. Therefore, it has been possible to enter a test team and to learn the ways others act and thus learn testing. Obviously, this requires an environment which has other testers to learn from, meaning an organization which has testing practices in place and people act in professional manner. Learning by doing – and observing and mimicking – is perhaps aided by short courses, including certification training. The problem with this is that there may not be test teams to provide possibility to learn from peers, nor are there necessarily mature processes in place. Even when they are, companies may only have a limited range of styles of testing, so the skills developed may be quite narrow and can be difficult to transfer to another environment. In practice, there are many testers with 15 years of experience but with a very narrow skill set and limited ability to work independently. So, we need to be very careful with this. Figure 33 shows the process based on learning by doing for testers who are entering the occupation with no existing competence.

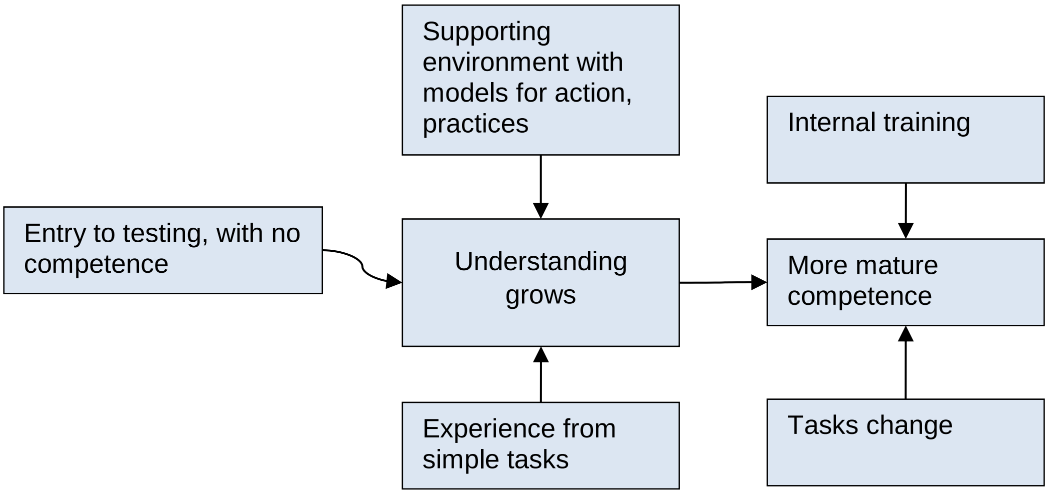

Supporting environment with models for action, practices
Internal training
Entry to testing, with no competence
Understanding grows
More mature competence
Experience from simple tasks
Tasks change

Figure 33. Learning by doing for incompetent enterers.

Similarly, developers can get to a role in a development team where they do more testing related tasks than others. Gradually, they will essentially become technical testers, taking care of for example development of test automation for the team, performance testing and similar tool-oriented testing tasks. These developers may even realize their role having been transformed to that of a tester and they may choose their future career based on that. Figure 34 shows the process based on learning by doing for testers who enter the world of testing as pure software developers.

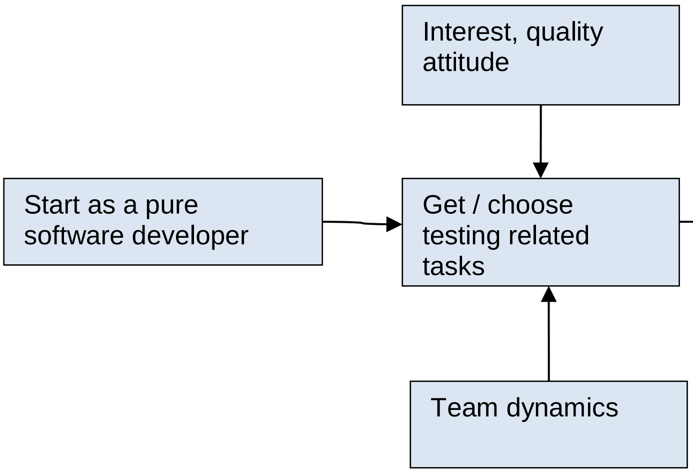

Interest, quality attitude
New self-perception
Start as a pure software developer
Get / choose testing related tasks
Become expert
Role changes
Team dynamics

Figure 34. Learning by doing for software developers.

It is more and more possible that testers get their first exposure to testing in a course during their education, which is more and more common in software engineering education (as was noted in the Introduction, first testing courses in Finnish universities were introduced at the turn of the millennium and by now they have become standard courses). For example, a five ECTS credit point (StudyInEurope.eu, 2015) testing course that includes good practice will give a good starting point for entering a company as a tester. What is good with this is that a good course will provide a wide range of viewpoints into testing and make it possible to adapt to the practical challenges – although with, naturally, some delays in becoming really productive. Also, such people may be more prone to find a book or a net site for learning new things about testing. Of course, they have the education-provided engineering skills which will help a lot. Figure 35 shows the learning process in a career started by education.

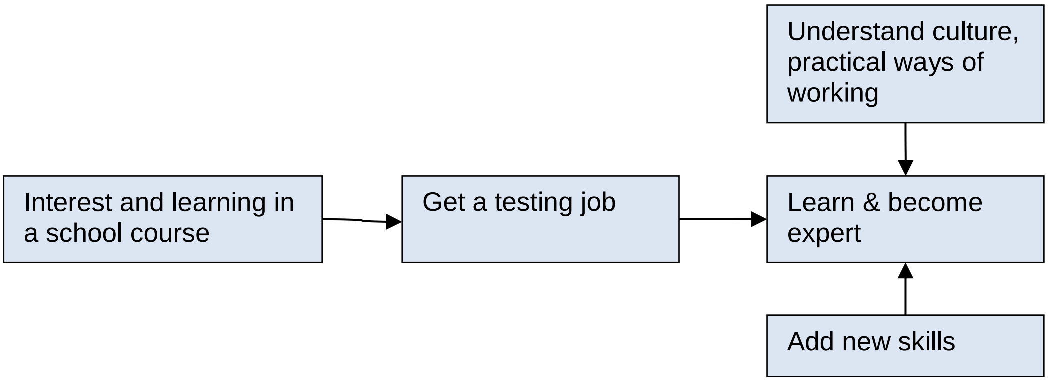

Understand culture, practical ways of working
Interest and learning in a school course
Get a testing job
Learn & become expert
Add new skills

Figure 35. Career start by education.

Those people will use two main mechanisms in their learning. First. analysis: What are the essential elements and challenges in this environment? Secondly, mapping and reflection: How do the things perceived map to the things already known in theory.

Of course, the competences can have many levels that provide a “competence path”. For example, the ISTQB certification is based on the idea that one first learns the “foundations” of the craft, and then more “advanced” skills and knowledge, finally becoming an expert. Because one cannot be a true expert of many things, at that point one needs to choose which expert she wishes to be.

In organizations, the choices have often been about whether one wishes to become a team leader or manager or to become an expert after gaining generic skills and experience. The orientation for testers seems also to be divided into two. Testers are inclined into manual testing or test automation – or at least we suppose so. So at some point they may make choices that reflect that. The manual testing paths nowadays usually means becoming an expert in exploratory testing. Figure 36 presents the traditional “organizational” career path for testers.

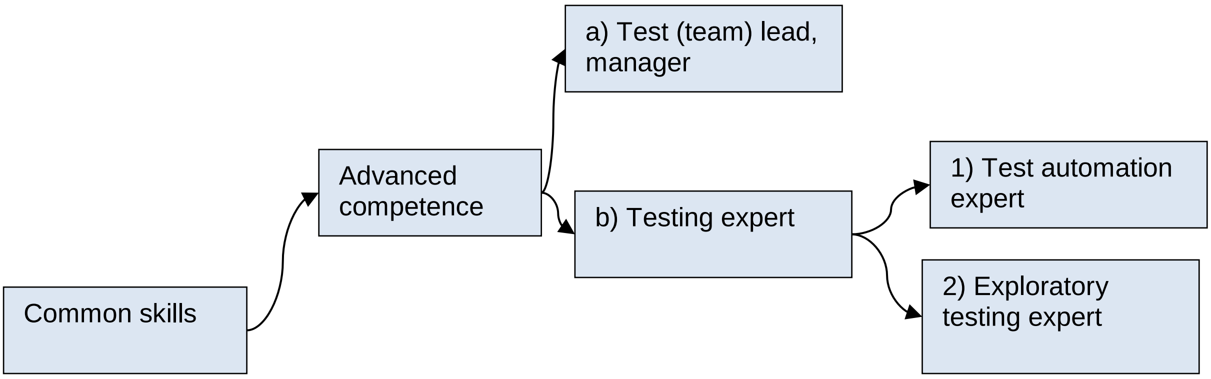

a) Test (team) lead, manager
Advanced competence
b) Testing expert
1) Test automation expert
2) Exploratory testing expert
Common skills

Figure 36. Organizational career path.

The idea of lifelong learning is essential. The technologies, processes and tools change rapidly so any tester needs to keep her competence in good shape. That requires personal investments and activity besides what is learned in the company context. A modern-day professional has plenty of ways to get new information to learn from. After the turn of the century, the number of generic and very specialized books available has been exploded. On web bookstores, such as Amazon, a great number of excellent books are available and often inexpensive prices. Similarly, the Web has plenty of social media material available, including these:

- Experts' opinions in Twitter (2013). For example, Canadian testing consultant Bolton has until 2015-11-26 written 52 600 tweets and has 12 900 followers. A respected testing expert from Finland, Maaret Pyhäjärvi, correspondingly had 6900 tweets and 1700 followers.

- Expert’s blogs.
- Discussion forums.

There are now good testing magazines available for free in PDF or a similar format. There are a lot of conferences, small local peer events and large international scientific conferences and everything between.

The above discussion is mostly about the professionals who work deeply in the substance of testing. Managers and directors in testing must not be forgotten as they may have a very critical influence on how testing is done. For those, similar detailed paths cannot be recognized. Managers (who do not grow from the testers) have a more generic, often an engineering background, and just are given or select managerial tasks that happen to do with testing.

4.12.3 Contexts for training and learning

We need to differentiate different types of training by their contexts and goals to really see what kind of possibilities they might have for improving the competences of individuals.

The following are the typical ones:

- Special courses about a topic aimed at individuals. They can be very effective if the substance matches the needs.
- Special courses about a topic targeted to a company. They can be very effective as the contents can be tailored to the specific problems and technologies the company has. Those are arranged and joined only if there is a real need for those.
- Tester certification courses. Those are said to be for testers to improve their competence, but can sometimes be of value only as a proof of professional knowledge. Their value varies based on the domain – on some domains the culture more matches what is being taught than on the others. Often the courses are arranged for a company and in those cases the real goal may not be improved competence, but to gain proof that a portion of testers has the certificate – for a client or for general marketing.
- Generic testing courses for a company. Courses that cover a wide range of topics may have a great value in building a shared understanding about testing in a company, which is essential for creating a solid quality culture.
- Mentoring. Mentoring is used very little, but it could have value when an experienced master aids a newcomer in how she builds her professional career and skills.
- Coaching in the context of work.
- Learning circles. For example, within the Finnish tester community there have been book reading circles where a group of people really try to understand what the author of a book is trying to say. The books chosen are ones that seem to be

important ones, based on how respected people in the local or international community have seen and recommended them.

- “Dojos”. A dojo is a peer training session where testers exercise their skills together. The idea started from “coding dojos” in the agile development culture. The name “dojo” is Japanese and means the exercise hall for martial arts, which reflects the skills base approach here. Things like this are seen important for some testers who see that they cannot fully utilise their skills in their work, lack tester colleagues and would like to see how others approach a testing task.
- Of course, conferences are also forums for learning. Of particular interest are local short peer conferences, as they are approachable to testers and are based on sharing participants’ practical experiences and views. It is easy to participate in a local three-hour session in the evening, particularly if it is free.

4.12.4 Personal learning from experienced people

Learning from more experienced people has always been seen as valuable. Most often it happens informally at the workplace where good practices are transferred by more experienced people to newcomers within the team, pair work of by a specified trainer who brings new people up to speed. People need guidance on more than the primary contents of the work in order to grow as professionals. They need some reflection on what is happening in their working and private life and that may not be possible in the work context. Mentoring offers that. In mentoring, a – usually – young person, “actor”, meets regularly a more experienced person, “mentor”, to discuss any issues related to the actor’s career as a whole. Mentoring programmes are available in Finland, for example The Finnish Information Processing Association, TIVIA, arranges them regularly. Still, a good master and apprentice approach is also invaluable in learning the practical issues of work. In that, a young person learns the craft from an experienced expert while working together. That may be a setting defined at the workplace or part of an education system and supported by more theoretical lessons in an institute.

Those two approaches are compared in Table 16.

Table 16. Comparison between mentor-actor and master-apprentice approaches

Element of interaction	Mentor-actor	Master-apprentice
Goal	To help the young person to understand herself, her professionalism and her path of development.	To raise a young person into a professional in some restricted work profile.
Formal relationship	Mentor is the actor's adviser, who does not have direct relation to her work and no impact to it.	Master is in practice the apprentice's supervisor.
Context of collaboration	Actor's (working) life, brought into shared discussion.	Apprentice's work. Work performance in shared context, with common goals.
Expertise of the ”older	Mentor is competent, experienced or has understanding about things in actor's domain, but not necessarily in the work methods.	Master is an expert in apprentices work.
Content	The whole of occupation. Opportunities. Actor's strengths and orientation.	Substance of work. Methods and tools. Quality work. Errors and avoiding them.
Means of expertise transfer	Tackling of acting and thinking by discussion, reflecting on experience, looking at things from top to bottom.	Transfer of tacit and explicit knowledge during the work. Advising, correction of error, looking at things from bottom up.
Results	An independently thinking, reflecting person who is open to alternatives.	A new skilled person, who can do the same things as the master and who has the same setting in her activities as the master (attitude, approach etc.)

5 Analysis of the changing environment

5.1 General and rationale for the selected changes

In this chapter, we analyse the various changes in the activity systems and context of various levels, first looking at global level phenomena, then going into the organisational level and the level of software development practices.

We do not, however, use the “formal” terms of context elements here, but more common expressions. This is done for making the text more approachable and to enable the readers to connect the information here to their contexts.

At any level, there are various changes and selecting the right ones is critical for the quality of the research. The rationale for selections was in this cases the following:

- The selected changes were subjectively felt as being representative of “what is happening” in the contexts. They are based on the mental model of the researcher and modelling is always subjective and even more so when it is a matter of qualitative research.
- The changes were seen as factually important and combine various phenomena in a sensible way.
- They offer a sensible “interface of thought and analysis”, a rich basis for thinking and writing.
- Some of them may be “imaginary” in a sense that they do not show yet strongly, but have a potential for expressing themselves powerfully in the future.
- They do not contain current, shallow hype. For example, the devops phenomenon is something that some readers would expect to see, but it really does not represent a credible change in the Finnish culture except in marketing.

5.2 Global environment

5.2.1 Digitalisation

The digitalisation unites many kinds of phenomena. Technologically it means that all devices and systems will be digitally controlled and thus are potentially networked together. Digitalisation is even a world view where the information technology takes over all areas of life and activities. As a social phenomenon it refers to the transition of the people's work to be done by robots and software automation, which changes the structures of the society. It influences our thoughts about the expected citizens' skills that now are perceived to include various digital skills from programming to using social media. From an information point of view, it refers to information being always and everywhere available to most everybody. The information available covers information about the people also and their behaviour. Digitalisation is at the same time a neutral phenomenon of the change of the world, provider of opportunities and also a threat. Which it is mostly, depends on the viewpoint chosen.

Here, we are the most interested about product and service development and will not address the social aspects. First we need to notice that while digitalisation is a very real phenomenon, it is at the same time a serious hype. To verify that, one needs only to check the Twitter hashtag #digitalist for Finnish discussion about it during the last couple of years. Another sign of it is the production of digitalisation guides by consulting companies. One example is a thick book by TeliaSonera Finland (2015) that addresses the "promise of digitalisation" and offers paths for new growth of business. For that reason, it allows us to discuss how such hypes can cause companies to make suboptimal products and system and thus harm the success of their organisation. Leading from that, we shall see how testing and various quality related practices may help here. Note that due to the broad scope of digitalisation, many of its sub-aspects will be addressed separately or among other themes.

A hype produces suboptimal solutions

Hyping tends to produce compulsive behaviour. A brain that functions compulsively does not make good planning where the importance of matters would be weighed and different solutions would be thought about. Instead, a digital solution is chosen compulsively and not the one which would serve objectives best. The quality of the concept of a new product or service suffers and a good concept is the starting point for all product quality. A careful realisation or tight functional testing cannot improve a bad concept. But concepts can be evaluated and that is a critical form of testing in this age.

Creativity is important nowadays (we'll see that addressed many times in later chapters) and will be even more so in the future. Compulsiveness reduces especially the creativity. The good scoping of objectives is a different matter and positive for quality.

In the technology hype, one proceeds technology first, instead of needs first. Often this is a mistake but of course in the product development it can also be a strategy for looking into opportunities and even chances – "let's try this and see what comes out of it".

In the phase when a new phenomenon enters the environment, people have no personal relation to the phenomenon besides through media and consultants. It is easy to believe in incompetent consultants. When a subject matter is new, only the media visibility will be up for discussion, instead of the competence. This danger is especially large, when the companies are lacking in ICT culture and do not have the competences for assessing the situation themselves. Obviously, digitalisation is offered to those companies first, that have not yet done much in that area.

The organisations naturally want to get along to the hype and to create ritual signs of the fact that they are in the frontline of the new culture. The employment of a Chief Digital Officer (or similar) is one such sign. According to social media, that has been a very common new position in companies during 2015-2016. Is it not a good thing? Not necessarily. A separate director keeps the new matter a separate issue and does not integrate it when necessary into the business. A similar case was previously in quality management. Unit directors did not have time to advance quality, but the quality director as an outsider could not do much inside the units. It would be a sorry s situation, if it were noticed that there are no profitable digitalisation projects in the range of vision! So one must invent ones. However, it is known that "culture swallows everything", anything glued-on does not live. Water carried into a well does not stay in the well. The hypes are often like carried water. Perhaps experts understanding the challenges of the business would be worth placing in the management tasks. Then the changes can focus on the improvements in the right parts of the system – and can utilise the actual experts of different subject matters in the planning, designing and implementations.

When there are always too few resources, hyping will take them from other, more relevant issues, matters. Thinking and operation of the organisation gets narrow-sighted. When the hype changes to something else, the organisational thinking will lack basis and direction.

Hypes are always associated with positive thinking. In other words, one wants to look for the strengths and the people who see weaknesses are stamped as reluctant to change. That is a destruction of the critical thinking. When under the power of hype, one always wants to be quick with things. Excessive speed in the planning is never positive. A healthy action-oriented way of doing things is a different matter than the pursuit of the speed records of making changes. Testing is a traditional means of bringing critical thinking into processes.

A hype is different from a strong vision

In the world of the digitalisation there is not only a hollow hype but also considered visions, which give the actors an ideal and the direction of the development.

The Germany-based "Industri 4.0" is one such. At its bottom the is a continuum of the operations of the current factories and operation which could be upgraded with a wider automatic machinery and logistics. This could almost form a new paradigm of autonomous manufacturing. Industri 4.0 gives a framework and the objectives for the development, inside which there is freedom for action to realise the unique objectives and needs of each business challenge. Many companies have similar, not so loud visions – for example a change from the seller of products into the seller of services, which is helped by the modern information technology whereby data collected for the excellent, proactive customer-oriented service. Those visions acknowledge the end of one period of operational paradigm and a possibility to find a new way of doing business. Note that Industri 4.0 is addressed separately.

Dangers of the unconsidered digitalisation

An old wisdom is that one should not digitise things that do not work. Sometimes it seems that companies want make the non-working things work with the magic in digitalisation (cf. test automation). The organisations often suffer from dozens of bad information systems. The unconsidered digitalisation will again produce more of such. Every one of them requires changes to people's tasks and produces more problems. Certain portion of them will never live to their expectations.

When organisations want to develop net-based services or mobile applications compulsively, they may resort to gimmickry that produces complexity, additional systems which do not work well for the customers and short-lived solutions which will be soon given up on. The ending of services, however, produces strong customer dissatisfaction. Big companies' services that have been killed after a couple of years are too familiar. Again, this emphasises evaluation of concepts.

The produced services often have bad usability and user experience and many kinds of technical shortcomings. Sometimes the customers are bound through them to social media services, which they otherwise would have no need or a desire to use. Handling this calls for assessment of usability and user experience through both analysis and testing.

We are extremely dependent on the information technology already. Increasing the reliance increases our vulnerability. Already the bases of services and telecommunications are vulnerable and every new service will increase our reliance and our vulnerability if the system serve a relevant need. If on the other hand it does not do that, the service is unnecessary. Note that the entertainment is also a relevant need. Cyber and information security and privacy protection are always connected with the digitalisation. Every digital system adds the threat surface. Risk analyses and security testing during the development are essential competences.

Ethics are an important issue for two reasons. First, there is a possibility for misleading customers (and also own business). Second, businesses and the society benefit from sustainable development and that is a serious ethical issue. Business and product planners should especially have solid understanding of ethical issues, but those doing product assessments should also have the skills to assess products from ethical viewpoint.

Different digitalisation situations

There is room for many different cases for the digitalisation in the world. The digitalisation of service processes is a traditional information systems development where the situation is usually relatively well understood (if the people in the team involved are professionals). The role of testing and of quality assurance is the same as always before. The same holds true for the normal automation of the manual work perhaps with ordinary robots.

In the disruptive projects, which shake our thoughts radically, the evaluation of the new concept is essential to carry out analytically and experimentally. It is not sensible to operate with traditional practices. Sometimes the customer's context is familiar but sometimes the planned development step takes us to a complex or chaotic state and first one must find out the logic by with the system works by analysis and experimenting, in other words by testing. Such cases include bringing intelligent human-like robots to social contexts. In the testing of such systems, the main focus is in the evaluation of the concept and the testing of the user experience, which is by nature "community experience" (the community here being for example a workplace community). That means being in an area of real challenges.

Sometimes the digitalisation is only about bringing customers into a community, letting them participate and empower them. Those require different competence than operation or product development projects. More essential than digital technology are the control of a community psychology, understanding communication and brand semantics in designing, as well as in the testing of the systems.

The main goal does not need to be more than getting of the data from the processes so that big or smaller data can be better used to analyse the customers' and users' actions and how the technology is used and how it works. That data can be used in learning about the product and the customers and in doing proactive planning and even designing new service concepts. The collecting of the data requires its own systems and emphasises the control of information security and the privacy protection.

Real competences are needed

All the above mentioned types of digitalisation are challenging and a high-quality execution of them requires real competence and not only the exchange of people's titles. At the organisation level a common orientation and will are needed. The implementers of the reform and participants in the post-change operations need to understand the change thoroughly and need to be able to participate in the change process and after it in the altered practices. Otherwise the situation is like in a bookshop whose customer was surprised about how a digital book was sold out and getting new stock would take a couple of days.

With the digitalisation, the account managers in a consulting company can magically become service designers by title who have no competence in the service design. Product development competence, in turn, may be replaced the use of communication consultants or with the narrow competences of focused technologists that aim only at selling their offerings, not at the success of the customer.

When the new services are produced, the company architecture of the organisation is emphasised. The organisations suffer from too many systems already and their integration is always deficient. The new services are easily separate from the others, they fragment the whole, they increase work and technical problems and reduce the interoperability of the systems. One issue here is that new developers and companies may not respect traditional ICT practices. They may for example use NoSQL databases, which in their "structureless structure" might help the development of rich systems, but which do not work well together with the rest of the SQL-based databases and may turn out to be very difficult to repair when there are problems (see for example Mei, 2013). In the architecture evaluation methods, one main principle is to identify future scenarios for the system that it must be able to handle. Corruption of a database is one example of such. Every new digitalisation case produces new information security risks and their competence in that area is needed in planning and testing.

Finding of the right direction

The customer orientation is a counter force for the supplier-driven digital hype. With that orientation, product or service reforms are assessed from the point of view of their customer value. It helps to make the whole set of services, visible to the customer, simpler and not a fragment digital mess. It empowers customers and does not bind them to new practices. With its help, shortcomings are improved, which are not just painted-on promises. Turning the attitude into the practice requires genuine development competence in addition to the attitudes.

It is central to understand the company's own identity. What kind of organisation is it? Is there a digital core or something else? If it is noticed that it is possible to carry out the transformation of the company for instance from a machine manufacturer to a production logistics service supplier house, genuine management of change is essential. In that, the essence of the whole organisation is addressed. The purpose of the organisation is connected to this. The people need meanings in their work. The purpose of few organisations is to digitise things. In the background there is, perhaps still hidden, a thought of making the customers' world is better or something else, like a vision of transforming some human activity. Such mental basis helps workers to free their motivation and turn it into action. All in all, digitalisation is about people and decision making more than technology. Sommarberg (2016) studied digitalisation in machine building industry from the viewpoint of it being a paradigm changer. His domain of study was the global container-handling industry. He notes: "(...) empirical results have indicated that the strongest inertia is related directly to people and decision making. Three of the strongest people-related inhibitive sub-drivers are lack of systemic understanding, management beliefs, and lack of capabilities". The first two are related to the self-understanding of the company and the third one about building into the needed competences.

An experimenting culture is good when suitably managed. Experimenting does not mean implementing everything that is tried, but experimenting for learning purposes and after that being able to decide whether to start development work or not. Saying "no" is often thought to separate successful companies from the rest, even if some consultants think it is the other way round. The emphasis for saying "no" is no doubt strengthened by the story of Steve Jobs who at his second career at Apple radically reduced the product line and returned the product designs to simplicity and returned Apple to business success. The purpose of testing in the experiments is not to verify prototypes but to create understanding about the new situation (often called sensemaking). Such is extremely important when entering new areas. But at the same time, testing must provide information that enables identification of bad concepts.

Sometimes the core organisation is unable to change in which case internal startups, which carry out that transformation in the small, can be used. The rest of the organisation will hopefully later join in, after the path has been presented. The startup phenomenon is addressed in a separate chapter.

Analogy with test automation

To the testing people, the idea and hyping of test automation is common from already some decades. In the ideals and dreams all testing will be automated and with just a push of a button, testing is quickly done. The reality is not quite like that. The test automation has seen great advances but still its significance is the biggest on the functional regression testing and its blind spots are in the quick testing of new features, in the concept level testing and in the testing of a use experience and usability. The new businesses and startups have a critical need for the latter ones. In those test types, the automation does not function, because the interpretation of observations requires a human. The testing is geared for increasing the people's understanding about its own product, not for finding technical defects. The good news is the fact that many companies understand that test automation is not intended to cover everything, but its biggest potential is the freeing people to a good creative testing. Also this will only succeed if the digitalisation of tests is done well and all the resources are not used for the maintenance of the test sets and test tools.

In a good digitalisation of testing, we can see this kind of principles:

- Such things are digitalised that are suitable for being digitalised.
- The technology is people's help and removes even unpleasant and harmful repetitive work from the people.
- The technology does not replace people in the overall process but frees them off routines to do sensible tasks, for which there has not been time earlier.
- Robust practices, robust infrastructure and compatible tools are used. The choices in tool chains are not based on trust and hope, but experience and knowledge about how the system works.
- The whole is built by professionals who are familiar with work processes and technologies, for themselves and for their colleagues.
- The development has clear goals and understood performance and effectiveness metrics.
- The digitalisation is made step by step, tailoring techniques and adjusting the whole.
- If the digitalisation has not been well made, the final result is bad for the people and the operations are ineffective and low quality.
- If it is made short-sightedly, it will not adapt itself to the changes in the operations (for example changes in product and development technologies).

Summary

The new digital technology is always a possibility but the ideas turn into innovations only by making the right choices and by designing the solutions customer-orientedly and making them sensible from the point of view of business.

Good testing can give great help. The evaluation and testing of the service concepts and product concepts improve decisions. The internal piloting which is done internally and with partners helps to build the solutions to the production condition. Various risk analyses are in the key role when reforms are thought.

This testing competence is not the world of functional testing and of test cases from the traditions but comes from experiment design, solid prototyping and the testing cultures of user experience and usability.

Change-competence snippet 1	Digitalisation
Change caused by -> enables	Advancement of technology -> opportunities, changes in products and systems, changes in cultures and societies
Competence implications (re: quality and testing)	Business understanding #O #U Customer-centredness #O #U #A Understanding new products and systems #O #U Working under insecurity and change #O #A Critical thinking and presenting critique #A Experiment design skills #A Evaluation of product concepts #A Doing proof of concept tests for technology #A Doing critical technology assessments #A Prototyping skills #A UX and usability testing #A Understanding permission, security, privacy #O #U Data analysis #U #A Understanding modern word and its new practices and thinking #O Understanding complex systems #U Managing change with information #A Understanding overall contexts #U System and system of systems thinking and testing #U #A Information systems and integration competences #O #U #A Risk thinking #U Risk analysis skills #A Understanding information security risks #O #U Business and product concept level testing #A Architecture evaluation #A Understanding about ethics #O #U Doing ethical assessment #A

Links with	-> Pervasive communication -> Changing Finland -> Information security and privacy -> Experimentation culture -> From products to services -> The startup phenomenon -> Machine industry turning into software industry -> Networked communication -> Experimentation culture -> Business understanding for all -> Industrial Internet -> Big Data -> Innovation in product development -> Rethinking the goals of testing and quality assurance

5.2.2 Industrie 4.0

It is claimed that the industry is in a process of a real revolution for the fourth time. The first revolution was the mechanization of industry with water and steam power, the second was the entry of electricity and mass production, the third one was when computers and automation were presented and now it is time for the fourth one.

The name "Industrie 4.0" was coined by a German project that promoted computerization of manufacturing and used for the first time in 2011 (Von Henning & Wolf-Dieter, 2011). So the concept was partly a signal of national strategy and a goalpost for industry. It is still a prediction, not something that is present. In non-engineering domains, the phenomena is even called the 4th industrial revolution.

Originally the term referred to stronger automatisation of factories, products that steer their manufacturing, advanced logistics and so on (Von Henning & Wolf-Dieter, 2011). For more background and drivers behind Industrie 4.0, see Draht & Horch (2014).

The ideas have obviously evolved. Hermann, Pentek & Otto (2016) made text analysis of descriptions of Industrie 4.0 and based on that describe the "design principles" of it as:

- Interoperability: The ability of machines, devices, sensors, and people to connect and communicate with each other via the Internet of Things (IoT) or the Internet of People (IoP).
- Information transparency: The ability of information systems to create a virtual copy of the physical world by enriching digital plant models with sensor data. This requires the aggregation of raw sensor data to higher-value context information.

- Technical assistance: First, the ability of assistance systems to support humans by aggregating and visualizing information comprehensibly for making informed decisions and solving urgent problems on short notice. Second, the ability of cyber physical systems to physically support humans by conducting a range of tasks that are unpleasant, too exhausting, or unsafe for their human co-workers.
- Decentralized decisions: The ability of cyber physical systems to make decisions on their own and to perform their tasks as autonomous as possible. Only in case of exceptions, interferences, or conflicting goals, tasks are delegated to a higher level.

All in all, this is a very wide umbrella term, covering large issues, which is why we shall later address it part by part as appropriate and reflecting the issues to the Finnish environment.

5.2.3 Responding to change

The scope of this dissertation is the near future. The aim is to understand what the near future might be like and to respond to that understanding with new ideas that would make that near future better. The idea is to be proactive towards it, and not reactive, not to mention it being necessary to act in a constant firefighting mode, if there is no preparation.

A general problem of thinking about the near future is that we don’t usually understand even the present fully. When the world around us is changing, we gradually understand the past better. The farther things are in the past, the better we understand them. Because of that, we see the present as a reflection of the past – the things that we understand. Because of that we often solve the problems of the past even though we think that we are solving things that are most essential now.

Regular thinking is like that. Futures research takes another approach[19]. In that we try to remove us from the present and see what the future might be like. Then we interpolate from that into the present and can more freely see what the next phases might be. Of course we do not know anything about the future, but we can make scenarios about it and base our thinking on those. Gradually, we can see what scenarios are turning into reality – or if we can affect things, what scenarios we should work towards, so they could turn into reality.

In that work we can use knowledge of trends. Some things have been progressing linearly and we can assume that they continue to do so. For example, we see no end to

[19] The author has experience of this from the domain of developing future user interfaces, where methods and practical toolboxes were developed for companies, see Vuori, Kivistö-Rahnasto & Toivonen (2001) and Vuori & Kivistö-Rahnasto (2000).

lack of product development resources and time. We can even think that it will get even worse. We can also see “weak signals” about something that is emerging and will be a part of the future. For example, we can see the startup companies getting more emphasis and may think that in some futures, the nature of companies may be different than it is now.

Usually, the scenarios are developed at various levels. The whole world around is changing and it affects us in many ways. We can see big changes in the world, often called megatrends that really affect everything. They can be technological – like the Internet being present everywhere, in everything – or perhaps political, environmental or economic. Indeed, the first thing to do is to divide the world into such areas that can be assessed individually. PEST analysis (PEST analysis, 2014) – including its variation, such as PESTLE – is one example of that. The letters in PESTLE refer to Political, Economic, Social, Technical, Legal and Environmental factors.

Below the global level is the national level. The phenomena in the world affect us, but there are also national phenomena that we can ourselves influence. Those include education, the local ecosystems, the professional communities and similar.

In the context of this dissertation, the elements of change can be divided into three activity types: general, product development and testing (including quality assurance). Those can be assessed at five levels: global, national, company, practice and personal.

To really find the issues that are relevant in any cell of the framework, we need some classification of the things that are relevant. We already mentioned the PEST analysis that looks into political, economic, social, technological and environmental issues. Something like that could be valuable. PEST is a generic, high level method. For this work we need something different, but we still don’t know what it is. Yet, we “know” that it most likely will be something different at every level and in every activity system type. In order to reach the essential issues, we need to do some free-form analysis of the phenomena around us and to see what kind of types of issues emerge from that. So, we will return to this approach a little later.

As for the changes, we are interested in various kinds of those. Some most important types of those, in relation to time, are illustrated in Figure 37.

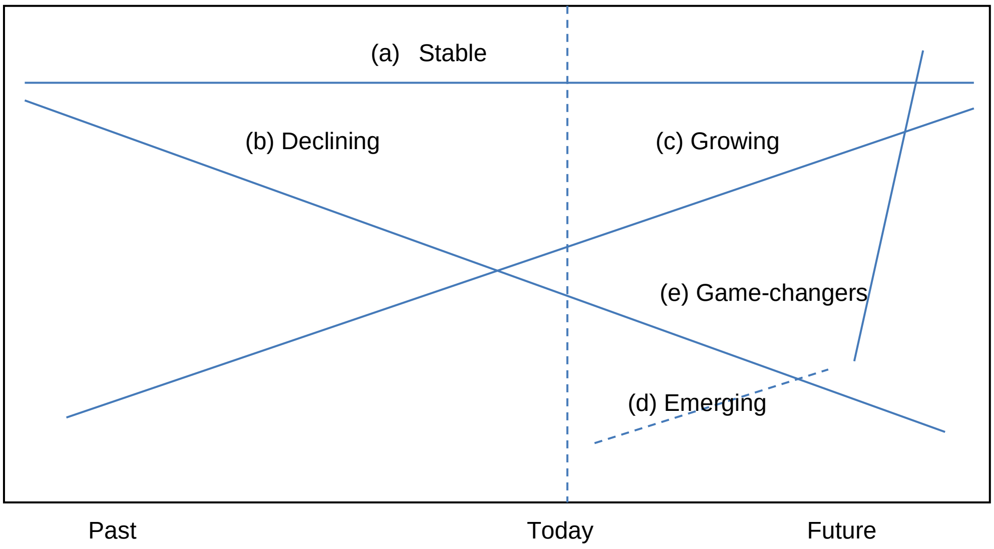


Figure 37. Various kinds of changes.

Some things remain stable #A. Still, even though they may be the same, they really are not, because when other things change, the relation to the stable things change and they need to be reassessed. Some things are declining (b), such as the role of waterfall processes. They may be replaced by some other approach to things, that gradually grows (c) such as general agility. The most important things are the emerging issues (d), which we really do not know what kind of shape and importance they will get later. Sometimes they may emerge very rapidly and be seen as instant “game-changers” (e), but most changes give some hints of them. Of the others we may have plenty of information, both practical experiences and research-produced information. Changes there will be. Let’s look into those on a personal level and think of a tester who has worked in the telecom industry for 15 years. Some of the changes she may have seen during the career include:

- Spatial changes: Going from local organization to a truly global mode of working.
- Cultural changes: Transformation from single culture to multicultural organization even locally.
- Management changes: Many organizational changes, many styles of management and leadership – from line management to working in projects or service teams.
- Test management: Growing expectations to produce more packaged testing services internally.
- Style of testing: Starting with systematic testing and gradually moving to an exploratory style. Expectations for testing skills grow.
- Organisation of testing: Moving from testing teams to testers located in development teams.
- Test automation. Expectations for being able to use test automation is growing gradually.

- Product technology. First simple products. Sudden change to complex computer-like products with tremendous amount of functionality. Product platforms change every few years.
- Project lifecycles: Transformation from waterfall projects to a distributed agile process.
- Pace of working. Shortening product cycles, more emphasis on delivery than development. Due to the new processes, the rhythm of working is more even, giving less slack time – and time for self and team improvement.
- Quality requirements. Quality requirements become wider in scope. Change from technical focus to overall quality, to user experience.
- Working infrastructure: New kinds of tools are introduced each year.
- Pressures in general continuously growing.

There are a lot of changes. Some of them may be easy to adjust, but for all it would be better if there were competences with which we were proactively ready for the changes – for any change, as the true nature of the changes is seldom known in advance.

Change-competence snippet 2	Responding to change
Change caused by -> enables	Changing world -> new ideas, practices
Competence implications (re: quality and testing)	Understanding modern word and its new practices and thinking #O Understanding domains, contexts and situations #O #U Understanding changing nature of quality #O #U Understanding new products and systems #O #U Working under insecurity and change #O #A Understanding complex systems #U Managing change with information #A Collaboration skills #U #A Right timing of actions #A
Links with	-> Responding to change -> Living with contradictions -> Agility and flexibility -> Need for new types of workers -> The changing requirements of technical software systems -> Fast product development

5.2.4 Living with contradictions

Our world view is often inclined to think that things should be in harmony. Yet, there is some evidence that contradictions are beneficial to good performance. For example, Osono et al. (2008) have researched auto manufacturer Toyota – in this context the company is most relevant for being the origin of Lean – and have identified many contradictions, which may be essential to the company's success. They are not something to get rid of, but something to cultivate. They list the main contradictions on page 9 as:

- Moving gradually and also taking big leaps.
- Cultivating frugality while spending huge sums.
- Operating efficiently as well as redundantly.
- Cultivating stability and paranoid mind-set.
- Respecting bureaucratic hierarchy and allowing to dissent.
- Maintaining simplified and complex communication.

In the worlds of process quality, similar items have been identified, for example the simultaneous goal of standardisation and improvement by making changes to the elements of activity. The agile software development culture emphasises freedom, but at the same time usually works in a tightly controlled process. The author has previously drafted a list of contradictions in testing (this was published in TestausOSY's LinkedIn group in 2012).

Some contradictions are related to attitudes. In testing there needs a simultaneous pride of own skills and know-how and humility in front of the limitations of those. A tester is an expert, but expertise is never complete and often insufficient.

In the practices and ways or working there are many contradictions. There is a need for systematic action and at the same time a need for simultaneous improvisation, when the new observations are learned. A tester must focus on one thing and at the same time have her senses open to observations about anything else. There should be minimizing randomness and chance, but at the same time the utilization of them is important. Testers should have broad utilization of tools and at the same better utilisation of human capabilities.

Some contradictions are related to learning and the ability to apply learning elsewhere. Testers need contextual thinking and at the same time learning of general issues to be able to apply it elsewhere. Learning practices and techniques turns them into effective routines, but we need to be sensitive to new things. There should be identification of good practices and continuous renewal of ways to work. Especially in the agile development, specialised expertise used in joint activity. There is a need for incremental operating and improvement, and a need for quantum leaps.

There are many contradictions in the working environment. There should be a climate of success trust towards others, including the developers, but at the same time errors are expected to exist always and everywhere. Participation should be sometimes extrovert participation, but good testing requires thinking about things in an introvert way. Testers should be bringing in new information, but letting people think about things themselves. This is leadership without leading. They should have close communication within team and assuring visibility to the outside.

Even the relation to quality is contradictory. One needs to understand that there are multiple “truths” about quality – quality to one can be non-quality to another. And yet one needs to develop the quality and efficiency without compromises. Quality of systems and processes is holistic. One cannot form an optimal whole when all parts are optimised. Perfectionistic attitude is used in order to achieve compromises. The invisibility of best quality is difficult. We know that the driveability of a car is the best when it is not even noticed.

Time produces contradictions. At the very minute when something has been learned to do well, it turns into old-fashioned.

Clearly, when there are so many contradictions, the ability to understand them and to utilise them, should be valuable.

Change-competence snippet 3	Living with contradictions
Change caused by -> enables	Modern complex world view -> making good decisions
Competence implications (re: quality and testing)	Handling contradictions #U Understanding overall contexts #U Broad flexible competence #O #U #A Multi-skilledness #O #U #A Working under insecurity and change #O Creativity #A Understanding changing nature of quality #O #U
Links with	<- Relation to change <- Changing Finland -> Agility and flexibility -> Flexibility over maturity

5.2.5 Pervasive communication

When the modern world of testing started developing a couple of decades ago, the information environment was really different. There were physical books about ICT, but not many about testing – just a couple of handbooks by Beizer and others, which a tester might accidentally get her hands on. Today there are tens of books about testing. There were no testing magazines. Some ICT magazines were circulating in companies based on the internal circulation list. Today there are many testing magazines. Of course there was no Internet, not to mention WWW and its blogs and discussion groups. Discussion forums, however, came to wide use during the 1990's in the form of Usenet News that carried tens of software engineering groups – not many about testing though. Communication with external parties was by phone, telefax and letters. Phones were wired and due to cost, long-distance calls often needed to be especially ordered. Project information was by large plans and reports, not by real-time dashboards.

In general, the amount of information around us was very small in comparison with the volume today. This progress makes it much easier to get information about anything, but also causes requirements for new competences. One must manage all the information, be able to judge which sources can be trusted and so on. One needs media literacy.

People need to be able to use the new information channels properly for their own information delivery. Using social media systems is very much different than using traditional information systems.

Change-competence snippet 4	Pervasive communication
Change caused by -> enables	ICT technology -> social media, embedded communication
Competence implications (re: quality and testing)	Using social media and web in getting information and sharing information #O #U #A Reputation management #A Communication skills #U #A Cultural skills (national, occupation, domains) #O #U #A Understanding permission, security, privacy #O #U
Links with	<- Information security and privacy

5.2.6 Information security and privacy

The author recalls that a decade ago it was often said that information security will be critical in the future, because every device will be connected to the Internet. Now that has happened, doorbells, coffee makers, refrigerator etc. are often connected to the home network and can be accessed from the Internet either by design or by flaws in the design or configuration of the system.

At the same time, the Internet services people used have been multiplied. People do their shopping, banking and government communications on-line, often using mobile devices and applications that may not be optimally secured. Not to mention social media, use of which is an everyday activity for many people. The shopping sites and social media are sometimes not that secure and it is not uncommon to see media headlines in the form of “Over 100 000 passwords leaked in [some service]”.

Views about the parties that are interested in our data have also changed. Traditionally the main conceived thread was an individual attacker who might for example get access to our computers and get our credit card information, but now there is an element of spying by foreign governments or platforms that might collect personal information and use it in some way they are not given permission to.

There used to be a trust to system components from reliable parties, but now there is a default suspicion that any component may contain some spying functionality.

The threats are no longer about data, but the whole of our lives. Systematic attacks can stop vehicles, heating, factories and communication channels. Governments and citizens are living a cyber-war every second.

We have entered an age of fear and even paranoia.

For product and systems development competences this situation implies the following:

- Everyone involved in the development needs to understand the security issues.
- Security risk analysis, security analysis and testing needs to be done for every application, system and system element.
- When doing usability assessments and functional testing, problems need to be identified that might cause security issues.
- There is a need for more people who can audit the security of platforms.
- Security must be a critical factor in the selection of platforms and components.
- An open attitude is beneficial. Preference for open, thus analysable and auditable components is very valuable.

Currently, operational security skills are rare. If a company would like to have security testing done, they need to hire a professional security consultant to do that. Companies need to have local people with sufficient competence. To reach that education systems must be developed, but before that can have an effect, companies need to take action.

There are developments that help in this. Understanding of how security testing is done in practice is spreading with the help of community projects such as OWASP (2016a) and its list of Top 10 threats. OWASP now provides practical guides for testing the threats and has guidance for both traditional web systems and mobile applications.

Another development is the entry of free open source tools for security testing. OWASP Testing Guide has an appendix (OWASP, 2016b) that lists various tools for these purposes:

- General black box testing.
- Testing for specific vulnerabilities: DOM XSS, AJAX, SQL injection, Oracle, SSL, brute force password, buffer overflow, fuzzer.
- Source code analysers.
- Acceptance testing tools.
- Other tools (runtime analysis, binary analysis, requirements management, site mirroring)

The OWASP Mobile Security Project has also a tools listing for security testing of mobile systems (OWASP, 2016c). And as can be expected, simple googling will reveal several other tool listings.

This area is related to the analysis skills required in any critical development. Safety and reliability analyses follow a similar strategy and use similar methodology as security risk analysis. Also, fuzzers are important for making systems tolerant to bad data and understanding their use is valuable anywhere systems are made functionally robust.

Any decisions about exposing people and businesses to risks are also ethical decisions and thus the handling of security and privacy issues requires ethical competences.

Change-competence snippet 5	Information security and privacy
Change caused by -> enables	All information online, connected systems and devices
Competence implications (re: quality and testing)	Risk thinking #U Understanding information security risks #O #U Risk analysis skills #A Product risk analysis #A Customer’s risk analysis #A Security assessment and testing #A System and system of systems thinking and testing #U #A Understanding about ethics #O #U Doing ethical assessment #A
Links with	-> Pervasive communication -> Industrial Internet -> Cloud testing

5.2.7 Attitudes towards competence

As we have noted already, competence seems to be important, as there are so many important and complex things to do. Furthermore, studies on startups (see Crowne (2002) and Thompsen (2003)) point out that competence can be critical to the success of new companies. This is very important in Finland now and will probably be so later too. One might even estimate that competence is absolutely critical and we should try to find the very best people to all the positions that are important to a company. Should not all positions be such? That might very well be the goal, but traditional personnel management is based on different principles. Good enough is good enough. Good enough is less expensive than the best. This may in some cases be good business sense, but its roots are in the factory era, where even skilled professionals just observed and used machines. Today's knowledge workers are working in different types of tasks and can have various kinds of influence on a company's success. Luckily, the testing tasks are improving too. At the turn of the century, many tasks were so repetitive and boring that a competent person would soon seek something else to do. Exploratory testing and creative test automation let people utilize a much wider set of their skills. Because of this we should see a real emphasis on competence.

Change-competence snippet 6	Emphasis on real competence
Change caused by -> enables	Companies rely on competences -> competences into use -> better business
Competence implications (re: quality and testing)	Understanding about competence #U Broad flexible competence #O #U #A Multi-skilledness #O #U #A Collaboration skills #U #A Team skills #A Work and process design #A (to tempt competent people) Competence development focused on business needs #U #A
Links with	-> Responding to change -> Agility and flexibility -> Need for new types of workers -> Quest for multi-skilledness -> Finnish style challenged -> Changing engineering education

5.2.8 Changes and implied competences for this section visualised

A visualised summary of the change-competence snippets in this section are visualised in Figure 38.

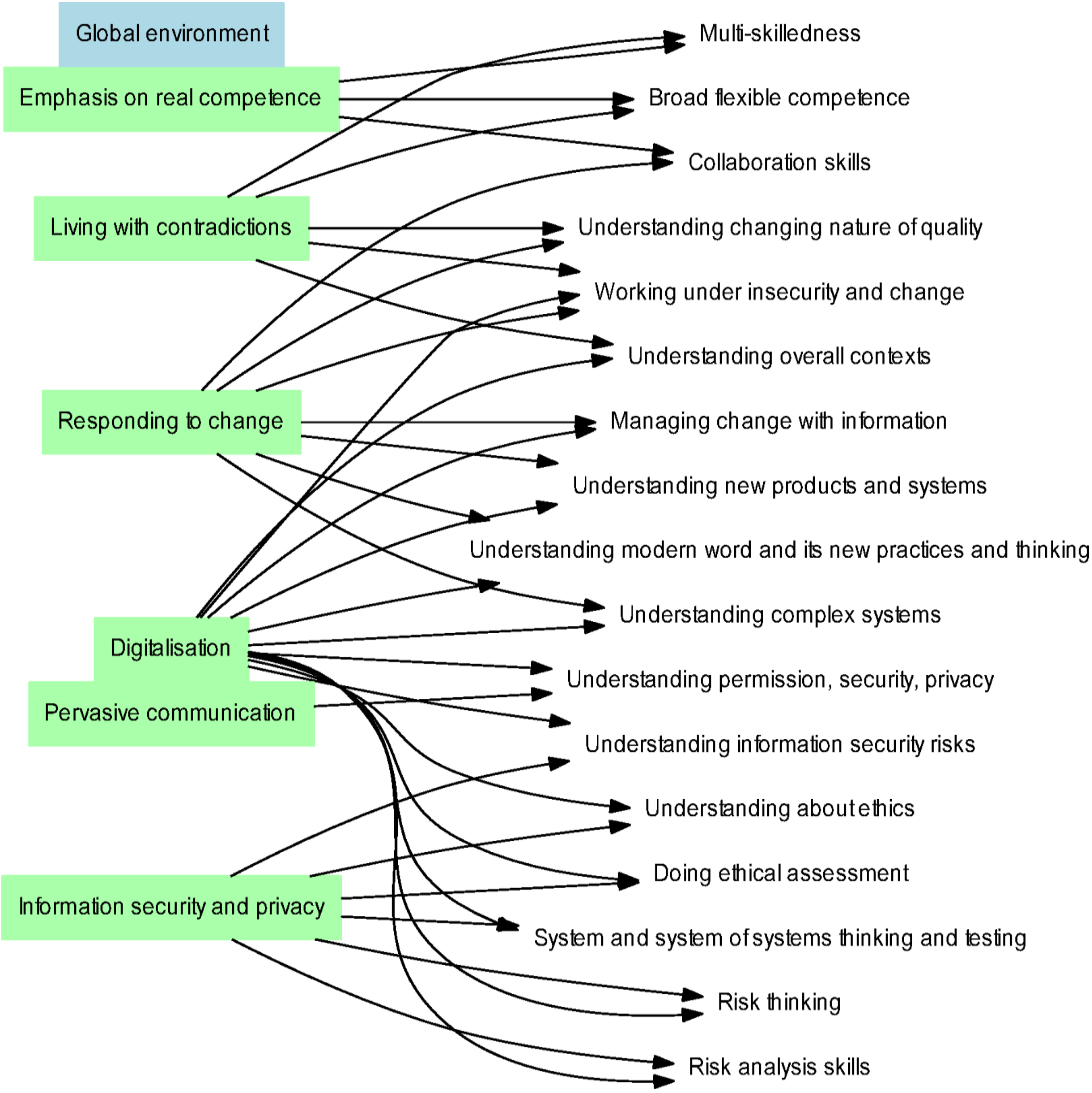

Global environment
Emphasis on real competence
Living with contradictions
Responding to change
Digitalisation
Pervasive communication
Information security and privacy
Multi-skilledness
Broad flexible competence
Collaboration skills
Understanding changing nature of quality
Working under insecurity and change
Understanding overall contexts
Managing change with information
Understanding new products and systems
Understanding modern word and its new practices and thinking
Understanding complex systems
Understanding permission, security, privacy
Understanding information security risks
Understanding about ethics
Doing ethical assessment
System and system of systems thinking and testing
Risk thinking
Risk analysis skills

Figure 38. Visualised summary of the change-competence snippets in this section.

5.3 Changing national working life

5.3.1 Model of a nation

As the scope of this dissertation is testing and quality assurance in Finland, we need to include a model of a nation too. It is presented in Figure 39. Models at that level are often used in practice in futures research, as the future scenarios of a nation form a context for lower level scenarios. As we have a hypothesis that there is – or should be

– something special in Finland, the model should help us in analysing what that is exactly.

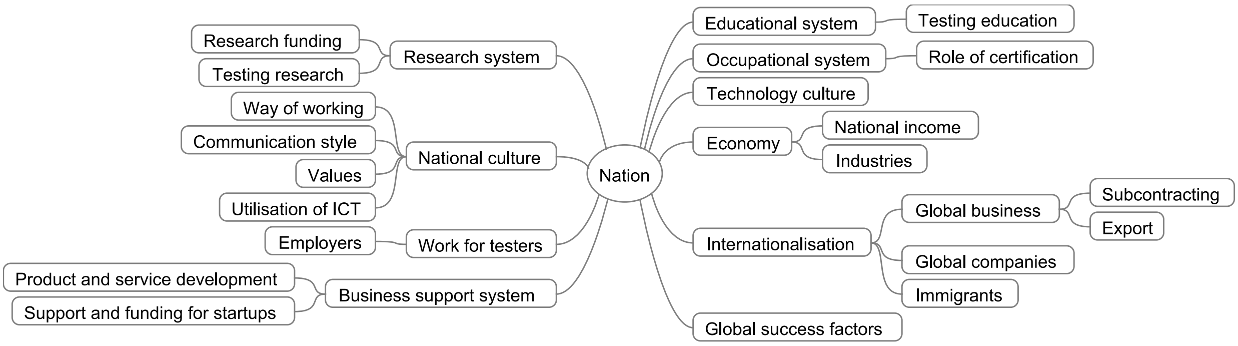


Figure 39. Elements of a nation – such as Finland.

Essential competence-related areas (for the purposes of this dissertation) include these:

- There should be an "activity stack" from advanced research to daily working, in order to keep the culture living and to push continuous advances.
- The educational system is critical. The people who enter companies need to know about testing and quality.
- Naturally the structure of the economy is important. If we see that large companies are important, our skill set should support the needs of those, but if small companies and startups are seen as critical, we need to promote competencies that those need.

Change-competence snippet 7	Changing Finland
Change caused by -> enables	Political changes, economy changes -> New opportunities
Competence implications (re: quality and testing)	National competence infrastructure development #A Improving education for quality and testing #U #A
Links with	<- Responding to change -> New external operating environment -> Smaller companies -> Need for new types of workers -> Changing engineering education -> Effective work in small, smart companies -> The startup phenomenon -> Finnish style challenged

5.3.2 Changing working life

In 2012, the Ministry of Employment and Economy published a general report about Finnish working life 2030 (Alasoini, Järvensivu & Mäkitalo, 2012). The report sees the working life in Finland to be a turning point, due to the forces of the developments of technologies, the globalisation of the economy, the actualisation of environmental questions, the ageing of the population, and changes in communal relations and values. Because of that, proactive development of the working life is needed at the level of societal policies, including experimenting and developing new solutions that, in the connection with of the change, promote productivity, employment, quality of the working life and work welfare.

A turning point has challenges according to the report. First, the work is always unfinished due to renewals and developments. Note that in the context of this dissertation this concerns mostly the acquirers of ICT systems; the development projects get their tasks done just as well as before. The fluency of work suffers because all changes, including new technology and new customers, continuously cause exceptional situations and more work and stress. Besides that, the work has lots of distractions and fragmentation – it used to be that a worker (including a tester) could before do similar tasks for days and years, but that will be rarer in the future. The meaningfulness of work is at danger, as the changes also affect the purposes and goals of work, and changing those is difficult. The working in the new operational models is also divided for networks formed of various actors. That causes a need to continually conceptualise and better understand one's own work. Workers need to find good answers to questions like "what am I doing?" and "how does the work done by others relate to my work?"

The report looks into the working life 2030 through various themes. The object of management is no more a clearly defined and stable organisation, a network formed of various kinds of actors and value communities, which is also dynamic, changing all the time. A challenge here is to create shared values and goals and views between the actors. That requires more dialogue.

Development of products, services and innovation is the work or more people than now. Organisation need to react rapidly to the changing needs and exceptions of users, customers and markets. Success in that requires open and distributed development and managing a constant unfinished state of things. A lean organising of development and innovation will be a core competence of successful Finnish companies.

Working hours, ways of working and the terms of employment are significantly more individual than now. In the organising of work, cultural characteristics and the diversity of people is utilised more than today. There is a change away from the traditional views of organisation as a machine and humans as rational to more cultural views. Networked ways of working will be more common. Organising of work will be done by more people (for example by the teams and not just by their managers). It will be like constant self-organising. More and more of that will be done by the workers themselves and the communities formed by them, done in agile ways based on the needs of each situation (self-organising development teams should be like that). Organising of work will be more and more a personal matter. There will be more diversity in the working life. There will be very different people working in very different ways. The age distribution will be broader, but that will be used as a positive thing instead of a problem. The human agency will in general be increased and the workers use it in more various ways. They will have various "game strategies" where they navigate in the field of working life and build a working life story for themselves. Organisations will be more democratic than today. The substance of work and the practices of organisations need to match the values of workers and customers better than now. This also shows in increased societal responsibility in the companies.

The competence requirements will be based on the previously listed changes and include readiness for distributed leadership, innovation skills, reflection skills, values-related skills, readiness to negotiate contracts and deals, the ability to build working communities (networks) and to act in those, readiness to manage of competence, working time and places, competences in ICT and social media, ability to utilise diversity and to build whole and the ability to take care of the health and coping with workload of oneself and the others.

The changes in working life in general are directly coupled with the activities in companies and we shall look at those more in later chapters.

Change-competence snippet 8	Changing working life
Change caused by -> enables	Changing society, new generation of population -> new ways of working, using people’s competences fully, rich work
Competence implications (re: quality and testing)	Broad flexible competence #O #U #A Multi-skilledness #O #U #A Collaboration skills #U #A Social skills #A Team skills #A Role finding #A Creativity #A Business understanding #O #U Active, self-steered working for quality #A Understanding innovation #U Cultural skills (national, occupation, domains) #O #U #A Communication skills #U #A

Links with	<- Smaller companies -> Relation to change -> Finnish style challenged -> Need for new types of workers -> Quest for multi-skilledness -> Changing engineering education -> Emphasis on real competence -> Better workplaces

5.3.3 Need for new types of workers

One basic premise of the research is that most of the views of testing that are deeply rooted in culture are based on ideas from some decades ago, and that things have changed since so much that every principle needs to be reassessed. How things really have changed is something that also needs to be reviewed at this stage of the research. There are two main elements in this: A model of the elements that our environment consists of, including a high level PEST/PESTLE modelling (see an example of that in Williams & Figueiredo, 2011), changing organisational and working life cultures and practices, views to lower level systems and structures and their changes – like project work, information technology, company structures and business models etc.

Another thing is the analysis of the actual changes. As the environment is mostly local, national, and we seek to find its unique characteristics, we need to find literature that assesses changes in Finland.

The final report of a three year Oivallus project (Oivallus, 2011) by Confederation of Finnish Industries assesses the forms of working in the future. The project has researched what kind of competences the business and industry will need and how that competence can be developed. Some notices from the report:

- The nature of society. We are moving from information society towards experience and experimental society.
- The business logic of companies is more and more based on innovations.
- The challenge of activities is whether we can work in a new way. Doing things “by the book” will no longer be sufficient.
- Nature of the work will be like jazz improvisation.
- The central competencies in the new environment are by the report: willingness and ability to work in a new way, ability to network, internationalism, business skills, technological skills, environmental skills, service skills, design skills. The report notes that even in the future “super individuals” are not needed – it is essential that the necessary competencies are found in teams and networks.

- Challenges in information management: Understanding that things are ambiguous and contractual. Ability to justify things to others is important.
- Subject of learning: Moving from individual information management and learning towards learning together, in networks.

So, this is the cultural environment for testers too and points to a broadening set of supporting skills. Obviously, we need to first understand the role of testing and quality management in the future, as the requirements vary in different tasks of different purposes. After all, one role of testing has generally been to be a balancing force to other professions' ambitions. Still, networking and innovation are examples of competencies that seem to be very relevant in any profession and any task.

Another relevant Finnish report is "Suomi tarvitsee maailman parasta insinööri-osaamista" (Finland needs the world's best engineering competence) published by Academic Engineers and Architects in Finland TEK (Mielityinen, 2009), based on large national collaboration by all interest groups. It concludes that while the professional core skills of engineers are naturally critical, much more is needed. Some items that the report sees important in the future:

- Creativity and innovativeness.
- Business skills.
- Usability of technology and productisation are essential.
- Risk management and an engineer's ability to see things three steps ahead.
- Sense of responsibility and ethics.
- Shared expertise, collective learning and facilitating skills.
- Problem-based thinking, reflection of own activity, collaboration.
- Ability to communicate own expertise to others.
- International and multicultural action.
- Understanding of differences in people as potential.
- Ability to stand stress and uncertainty.
- Ability to learn by doing.

This list of skills is close to the testers' world, as many testing roles are based on engineering culture. Thus, those competencies are very relevant to analyse further and provide a reflection base for assessing some specific domains.

Yet, we need surveys more focused on ICT. They often tend to be regional and thus are influenced by a given region's culture and history, company type, history and hopes for the future. One such is a regional survey made in Kainuu (Ahvenjärvi, 2011). It is interesting, but we need to be careful about generalising its views. It reports as the most important qualification in the future to be (in declining order): top expertise, proactive thinking and following of future trends, keeping competence up to date, and keeping in touch with technological development. What is notable is that team working skills were rarely mentioned, as well as the ability to listen to the customer. What can we learn about that? Each domain and each area has its own perceived challenges and we must not trust generalisations. Also, any survey notoriously has its own deficiencies that we need to assess.

Still, any views to the future need to be compared with today's situation. Ilkkala (2010) reports a survey of today's perceived competence need of ITC companies in Tampere region. In that, the most important skill areas were: basic ITC skills, teamwork and negotiation skills, working in the customer interface, basic skills in data networks, user interfaces and usability and database management.

In 2015, a survey was made in the Tampere city region (Vuolle & Alanen, 2016) that aimed in collecting information about the competence and training needs of companies in that area. The survey supported a goal of finding new employment opportunities for the large amount of ICT professional, many with a long experience. The survey first interviewed ten companies and the results were used in designing interviews for further 18 companies. At the final phase, an electronic survey was sent to a larger group of companies. The report concludes that the competences are on wrong technology areas. There is for example a need for front-end and Java developers, but there are plenty of unemployed embedded systems developers. There are simply not enough skilled employees on the key technology areas, such as web technologies. There are a lot of unemployed managers, who have lost touch of the actual software development and (automated) testing. Emphasis should be on refreshing their development skills and learning new competences. It should be noted that the unemployment situation in this region is partly a result of layoffs at Nokia and its subcontractors.

The main trends raised in the survey were:

- The needs for basic level programming language competences is the same as before.
- Need for competence skills in web technologies is on the rise. Efficient data storage data analysis and new scripting languages have emerged as important development tools.
- Integration into cloud services and operations management systems is in a large role in the industry.
- Linux, industrial Ethernet and embedded systems are rising in importance when the industry is moving towards IoT. At the same time, information security in emphasised at various levels.
- Programming practices are changing and functional programming and model-based programming (MBSD) were noted as areas where there will be a more common need.

- Test automation, continuous integration and understanding overall quality were emphasised. The quality of software products must be high and the basis of designing from the start.
- Utilisation of various standards and project manager certificates are only emerging, but fulfilling those is included in the requirement lists of foreign customers in project assignments.
- Solution based sales, presentation skills and practical language skills and cultural skills are areas in which the product developers are wished for improvement.
- Understanding overall businesses and processes – production, product development, solutions that produce savings – are in an important role. The production flow and keeping the end product in the scope of implementation are things that make also Lean and Agile high on the lists.

The report also includes a concrete list of important development technologies.

When we consider the various reports, we need to note that their goals and backgrounds can be different. The company surveys are focused on short term needs and practical tools and technologies and on the recruiters perceived needs and problems instead of more analytical thinking and emerging opportunities. In fact, early results of the latest Tampere region survey (Vuolle & Alanen, 2016) was presented in February 2015 in a seminar about software development competences[20] where it got criticism from the audience for being very tool-centric, the idea being that a professional should be able to tackle any technologies and tools if she has a good competence basis from her education. Indeed, recruiters are often criticised for tool-centredness and just thinking of the needs of the next project. There are always sudden turning points in regional situations when old structures collapse and it takes careful reflection to see to overall picture clearly and to find the essential level of focus for future. Yet, the overall business considerations and communication skills are common in all studies.

At this point we could have some reflection from the USA. Google is one of the companies that publish books about how they work. A famous book is “How Google tests software” (Whittaker & Carollo, 2012) and in 2014 they published a book about their general working “How Google Works” (Schmidt & Rosenberg, 2014). It is a business book and just documents how some people at Google see things and think about them and is expected to have been accepted and by the company’s marketing department. Still, it is interesting and worth assessing here, because anything Google says causes ripples – people tend to think that Google’s way is somehow the “best one” and applicable in other types of organisations. Every other organisation is different

[20] Nääsvillen Oliopäivät 2015 at TUT, http://www.cs.tut.fi/~systa/Oliopaivat2015/

than Google! There are similar challenges though: for example, a need for innovation affects every company.

In the book they describe how Google's employees present a perhaps unique type of worker:

"When we contrast the traditional knowledge worker with the engineers and other talented people who have surrounded us at Google over the past decade-plus we see that our Google peers represent a quite different type of employee. They are not confined to specific tasks. They are not limited in their access to the company's information and computing power. They are not averse to taking risks, nor are they punished or held back in any way when those risky initiatives fail. They are not hemmed in by role definitions or organizational structures; in fact, they are encouraged to exercise their own ideas. They don't keep quiet when they disagree with something. They get bored easily and shift jobs a lot. They are multidimensional, usually combining technical depth with business savvy and creative flair. In other words, they are not knowledge workers, at least not in the traditional sense. They are a new kind of animal, a type we call a "smart creative," and they are the key to achieving success in the Internet Century."

In the book they raise a concept of smart creative that they think describes the kinds of people that Google needs. It is defined as:

> "[smart creative is] a person who combines deep technical knowledge of his or her trade with intelligence, business savvy, and host of creative qualities."

And what are the characteristics of a smart creative according to the same source?

"A smart creative has deep technical knowledge in how to use the tools of her trade, and plenty of hands-on experience. In our industry, that means she is most likely a computer scientist, or at least under stands the tenets and structure of the systems behind the magic you see on your screens every day. (…)

She is an expert in doing. She doesn't just design concepts, she builds prototypes.

She is analytically smart. She is comfortable with data and can use it to make decisions.(…)

She is business smart. (…) She is competitive smart. (…) She is user smart. No matter the industry, she understands her product from the user or consumer's perspective better than almost anyone. (…) A smart creative is a firehose of new ideas that are genuinely new. (…) She is curious creative. She is always questioning, never satisfied with the status quo, seeing problems to solve everywhere and thinking that she is just the person to solve them. (…) She is risky creative. She is not afraid to fail, because she believes that in failure there is usually something valuable she can salvage. (…)

She is self-directed creative. (...) She is open creative. She freely collaborates, and judges ideas and analyses on their merits and not their provenance. (…) She is thorough creative. (…) She is communicative creative. She is funny and expresses herself with flair and even charisma, either one-to-one or one-to-many.”

And they note realistically: “Not every smart creative has all of these characteristics, in fact, very few of them do. But they all must possess business savvy, technical knowledge, creative energy, and a hands-on approach to getting things done.”

Now, let’s see how that list of characteristics look from the viewpoint of modern Finland:

- The list emphasises creativity and risk taking. Large companies can let individuals take plenty of risk, because the company has structures and mechanisms that control the risk. Actually, it is just because of the mechanisms and structures that companies need risk takers! Every force needs a counterforce. Small companies can’t afford that. They need to have people who take risks, but also manage them and every team needs to have a balance of risk taking and controlling it.
- “She is driven to be great, and that doesn’t happen 9-to-5.” If excellent people do not restrict the use of their energies, they are in serious danger of burning out. It may be that the smart creatives are like that, but we need individuals and organisations that manage their energies so that their work is sustainable.
- The list says little about teamwork skills, except noting willingness to collaborate and being “funny”. We could say from the Nordic point of view that humour and happiness results from a good working place and work profile. Finnish experts of organisational and occupational development usually emphasise those more than personal funniness.
- The ability to do, to make prototypes – or test systems or whatever – is indeed a critical competence in many situations, as modern organisations need to be able to build their own tools and products rapidly and everyone should have some applicable skills in that regard.

Overall, the list of characteristics matches many of the needs of companies. Yet we need to understand that diversity is one of the keys. An organisation full of “smart creatives” only would probably be a failure although not as likely as a company that consists only of traditional stereotypical software engineers.

Change-competence snippet 9	Need for new types of workers
Change caused by -> enables	Changing society, economic systems -> opportunities for broad competences
Competence implications (re: quality and testing)	Creativity #A Understanding overall contexts #U Business understanding #O #U Active, self-steered working for quality #A Broad flexible competence #O #U #A Multi-skilledness #O #U #A Personal competence development #A Creativity #A Risk thinking #U Role finding #A Social skills #A Collaboration skills #U #A Communication skills #U #A
Links with	<- Changing working life -> Finnish style challenged -> Changing engineering education -> Quest for multi-skilledness -> Better workplaces -> Living with contradictions

5.3.4 Changing engineering education

The changes in the society and operating environments of companies are reflected in the way people work in projects. That change is also visible in engineering education. Here we present one example of that change.

One part of engineering studies in universities has traditionally been a project work course, where student teams have been given a practical product development task to carry. The traditional courses are very engineering oriented, but on the side of those courses, as an alternative, innovation-oriented courses have been developed and they closely reflect the challenges of modern business and product development in many startups.

The author was during 2011-2012 involved in the development of one such course that is implemented in collaboration between Tampere University of Technology, University of Tampere and Tampere University of Applied Sciences and innovation platform Demola (www.demola.fi). The main characteristics of that course are described by

Pippola et al. (2012) and in condensed form by Kuhanen et al. (2012). Here we reflect on some of them.

The course is carried out in a system in which the companies order demonstrators of the new ideas from the student groups in order to weigh their usefulness as products or services. Learning that is essential for future professionals and learning to analyse and test the demonstrators is a critical testing skill of the future. International students have a strong presence in the teams, thus, the students learn to work in cultural diversity, which is very good learning. The university teachers have only a supporting role. It is emphasised that students learn by doing and also form their mistakes.

There are also workshops during the course, in which students can familiarise themselves with essential ideas and product development practices. Students themselves can select some of the workshops by voting, which forces them to reflect on their current knowledge and what new knowledge they would need the most.

During the course, students reflect their progress, activity and learning by writing weekly blog posts and the end report of their project. The outcomes are assessed by the real customer and in public pitching sessions, and not by the teachers.

Finally, in the teams, the students are encouraged to utilise all or their skills, not only the special skills needed in their primary role in the team. Ability to dynamically negotiate work in a team and to do a variety of tasks directly relates to what the working life needs now and even more in the future. At the same time, it provides total support for students' development as humans and flexible professionals.

The author analysed the differences of the Demola course and the traditional courses and the results of that are summarised in Table 17.

Table 17. Comparison of traditional project work course and innovation project based on (Pippola et al. 2012)

Attribute	Traditional project work course	Innovation project course
Uncertainty, risk level	Moderate risk	High risk, high uncertainty
Scope	Defined	Defined
Mental focus	Processes, routines, execution	Substance, business
Main quality factors	Fulfilling customer needs, total quality of action, re-usability of results	Value and re-usability of concept, new possibilities – creative thinking, product potential

Attribute	Traditional project work course	Innovation project course
Relation to tradition, rules, thinking patterns	Follow rules, use heuristics	Break rules, think differently
Main reusable result	Product, documents	Idea, conclusion, principles
Lifecycle emphasis	All equally	Concept, fuzzy front end feasibility study, proof of concept, marketing
Working environment	Closed, homogeneous, one culture, team work alone	Open space, networking, heterogeneous, multicultural, international, all teams in one space
Communication	Inside team, rhythmic with teacher / long cycle	Inside team, between teams, short cycle with customer/partner, networking
Language	Native language	English
Product rights	Team, licensed to customer	Team, licensed to customer
Skill set	Systematic project work, professional action, development & research methods, teamwork	Problem solving, teamwork, creativity, handling uncertainty
Learning experience	Project work, project management, how methods and theory work in practice, teamwork	Project work, team work, potential of creativity, intercultural working

As the projects do not intend to create production-ready systems, but demonstrators and prototypes, technical testing does not have a big role. Instead, the testing is concentrated on product validation and testing of ideas. Those are very essential domains of testing today and in the future.

Change-competence snippet 10	Changing engineering education
Change caused by -> enables	Need for innovation and product development skills -> new businesses
Competence implications (re: quality and testing)	National competence infrastructure development #A Understanding innovation #U Assessment and testing of innovations and product concepts #A Creativity #A Understanding overall contexts #U Business understanding #O #U Experiment design skills #A Prototyping skills #A
Links with	<- Changing Finland <- Changing working life <- Quest for multi-skilledness <- Innovation in product development <- New technology products <- Experimentation culture

5.3.5 Growing challenges for cultural competences

Especially in the engineering domains the cultural competences were traditionally – meaning before the turn of the century – quite simple. The testers needed to understand mostly professional cultures, the developers' culture and management cultures, as most operations were local and single-national. For many companies, international and global operations brought a critical new issue: understanding the co-workers and collaborators from other countries residing in the local country or foreign countries. Today, even a term of cultural intelligence is used in organisational settings (Sivasubramanian, 2016).

At the same time, as product technology evolved from basic engineering to something having more features and advanced user interfaces, designed for a larger variety of uses, product development become more customer-centric. That posed another level of cultural awareness: there are quite many different customers and users and we need to understand how they perceive the products, how it reflects their cultural features, values and habits.

These eras have very different requirements for all professionals. The growth in the requirements for cultural competences is visualised in Table 18.

Table 18. Cultural competence areas for testers in different product development eras (a small collection by the author).

	Local engineering era	Global engineering era	Global product business era
Cultural groups to understand	Developers Management	Developers Management Multinational co-workers in the same country Multinational co-workers in other countries	Developers Management Multinational co-workers in the same country Multinational co-workers in other countries Global customers Global users
Cultural issues to understand	Processes, ways of working	Processes, ways of working National characteristics Intercultural communication and working	Processes, ways of working National characteristics Intercultural communication and working National product preferences for design excellence Cultures of new domains

Change-competence snippet 11	Cultural competences emphasised
Change caused by -> enables	Raise of abstraction level, more communication, global networking -> shared competences into use
Competence implications (re: quality and testing)	Cultural skills (national, occupation, domains) #O #U #A Cultural adaptation #A Communication skills #U #A
Links with	<- Changing Finland <- New external operating environment -> Finnish style challenged -> Need for new types of workers -> Networked communication

5.3.6 Changes and implied competences for this section visualised

A visualised summary of the change-competence snippets in this section are visualised in Figure 40.

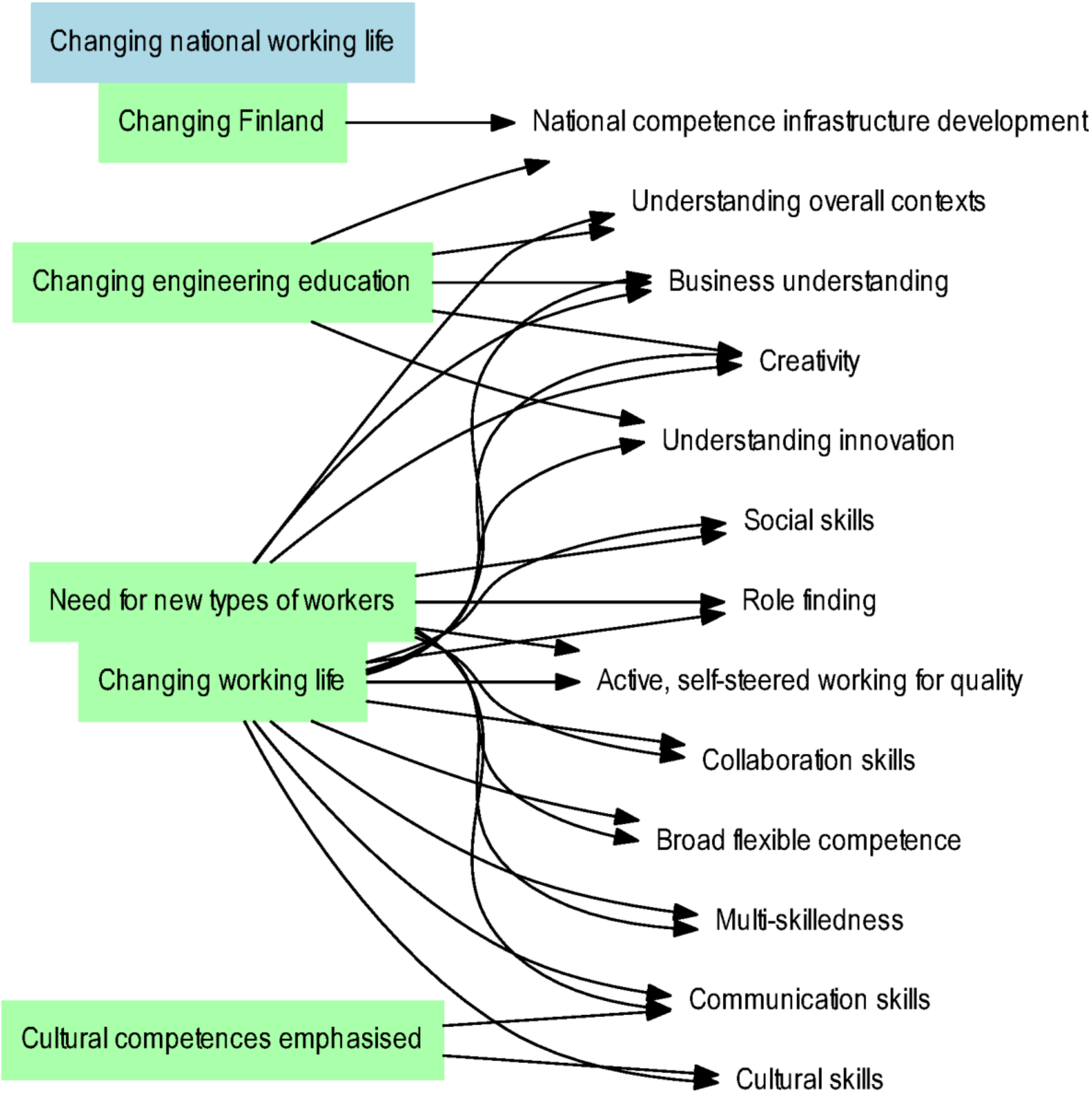

Changing national working life
Changing Finland
National competence infrastructure development
Understanding overall contexts
Changing engineering education
Business understanding
Creativity
Understanding innovation
Social skills
Need for new types of workers
Role finding
Changing working life
Active, self-steered working for quality
Collaboration skills
Broad flexible competence
Multi-skilledness
Communication skills
Cultural competences emphasised
Cultural skills

Figure 40. Visualised summary of the change-competence snippets in this section. To make the graph fit the page with legible font size, only the competences are included that are associated by at least two changes in this section.

5.4 Changes in the structure of the economy

5.4.1 The organisation types and sizes of companies

Finland was known as a country where the economy in ICT was built around huge and large companies. The most notable of those was Nokia. Such companies built around them large supplier networks of smaller companies that produced among others software development and testing services. That made life easy competence-wise. The smaller companies were given processes and tools to use and competence creation in testing was centred on the processes. As the companies could concentrate on a small number of customers, the technologies and testing tasks remained quite stable and easy to manage.

Now the economy has normalised (in relation to most other countries) so that the days of the giants are gone. There are startups, small companies and middle-sized companies that are independent in their actions. They are free to choose their organisation models, processes, technologies and all details of testing (unless they work in regulated areas). That produces variation and dynamic changes in the utilisation of technology and challenges for acquiring and developing competences. Smaller companies have lower resources and organisations are used effectively, making it difficult to arrange for example training for personnel, not to mention full training programmes. This emphasises the need for professionals to actively learn themselves.

Change-competence snippet 12	Smaller companies
Change caused by -> enables	Normalisation of society -> more dynamic economy
Competence implications (re: quality and testing)	Broad flexible competence #O #U #A Multi-skilledness #O #U #A Adaptability and flexibility #U #A Independent problem solving capability #A Competences usable in various process models and contexts #A Personal competence development #A Forming and promoting practices #A Active, self-steered working for quality #A
Links with	-> Changing Finland -> Need for new types of workers -> Finnish style challenged -> Quest for multi-skilledness -> Emphasis on real competence -> The startup phenomenon

5.4.2 Companies' external operating environment

External operating environment is continually changing. Some essential change factors include:

- Global competition. At the same time as more Finnish companies than ever operate internationally via the Internet, they get competition from every country in the world.
- Global operations. Many companies have some global operations. Software development or testing is often off-shored.
- Changing value chains. New modes of collaboration, component inclusion in services and products, platform arrangements and so on.
- Changing technologies, platforms. Companies need to be ready for any platform to change at any time. The mobile industry sees currently often changes of that kind. This leads to an understanding that our competences must not be tightly tied to some platform.
- Risk management is more essential than ever.

Change-competence snippet 13	New external operating environment
Change caused by -> enables	Global economy -> opportunities
Competence implications (re: quality and testing)	Risk thinking #U Cultural skills (national, occupation, domains) #O #U #A Networking skills #A System and system of systems thinking and testing #U #A Platform-agnostic skills #A Comparison testing #A
Links with	-> Relation to change -> Cultural competences emphasised -> Modern risk management -> Emphasis on real competence -> Networked communication

5.4.3 From products to services

The cloud is one example of turning products into services. Instead of providing installable information systems or application they are offered for use in the Internet on subscription basis. Even common desktop applications are turning into that mode, including Adobe's image editing and publishing applications and Microsoft's Office suite. On the domain of machinery, Kone Corporation is known for expanding its portfolio first from elevators to their proactive and reactive maintenance service and after that to intelligent overall systems from managing the "people flow" in buildings and areas (Kone, 2016).

This is a move where the core product is expanded to meet the real needs of the customers on a more value-added level. After all, "building businesses" don't need elevators as such – they need fluent people flow between places. Similarly, machine builders are expanding from pure machines to fleet control, production logistics and similar.

This provides a real need for expanded competences, as the product stack grows. The main focus of technology is no more electro-mechanical system technology, or software controlled local level automation, but customer and user-oriented intelligent information systems and monitoring systems that are integrated into the customer's businesses more and more directly. Even the core technical technology developers and testers need to consider the overall systems and customer experience at all levels of the system. For that to succeed, teams need people who have multi-disciplined knowledge, deeper understanding of the customers' business and understanding of customer experience, and competences in distributed information systems – not to mention security competences, as the integrated systems are prone for attacks.

Testing of those systems needs more rigour than traditionally. The systems offerings may be combined with service level agreements and should a large logistics system fail, the compensations can be huge. The requirements are similar as in the Finnish culture were the paper mills. When they have a disturbance that stops them, every second costs a lot.

So there is a need for high rigour combined with multiple disciplines that supports product definition and quality assurance. That is obviously a question where a large organisation is needed – small teams of individuals are not sufficient. And that emphasises communication and collaboration skills.

As those kinds of systems are by necessity tailored for each installation and will have a long lifespan, configuration management for the overall system is an issue that reflects highly on testing too. Every test must consider carefully the configuration to be valid. The system integration actually helps in this as when the information systems are controlled by the manufacturer, it is known how the systems are run.

Change-competence snippet 14	From products to services
Change caused by -> enables	Managed products, raise of abstraction level, Industrial Internet and IoT -> more added value
Competence implications (re: quality and testing)	Understanding overall contexts #U Business understanding #O #U Customer-centredness #O #U #A System and system of systems thinking and testing #U #A IoT-related competences #A Information systems and integration competences #O #U #A Risk thinking #U Risk, safety and reliability analysis #A Security assessment and testing #A Competence development focused on business needs #U #A
Links with	<- Digitalisation -> Platform economy and API economy -> Business understanding for all -> Modern risk management -> Machine industry turning into software industry

5.4.4 Platform economy and API economy

There is a now lasting phenomenon where digital platforms emerge that allows various parties to participate in providing value to a main service, to offer their value-adding products in the environment, thus turning traditional markets into an organic entity. Depending on the domain, this may have far-reaching effects for how societies work, how the economy works and how we individuals work. Kenney & Zysman (2016) describe many of the economic consequences.

Examples of the platforms include Amazon as a marketplace for all willing to sell their products, web browsers as platforms for marketing plugins that offer enhancements to the primary product, mobile apps ecosystems with their stores and selling mechanisms contained in the apps themselves, Google and Facebook that offer platforms for building platforms (!). In the so called "real life", Uber is an example of dynamic platform where aspiring tax drivers can join in the service network. Cities offering open data for local companies are building a platform that is a digital city, digital public government.

In Finland, there has been concerns about our abilities to work in the platform economy, exemplified in a report commissioned by the Prime Minister’s office that in the title asks if Finland is missing the train of platform economy (Alisto et al. 2016).

The platforms need various technologies, such as cloud-based collaboration environments with information and communication systems, tools for integrating the vendors' offerings into the platform – the cloud environment and the customers' environments (such as mobile phones). Many of the technologies are Internet-based.

There are three distinct relations companies and people can have with the platforms. First, they can build a platform. How that would be done in various domains is an interesting question and a challenge for product innovation! Secondly, they can commercially participate in a platform, finding a role in the ecosystem. That can be selling products in the ecosystem, providing consulting services or development services and so on. Third, they can just use the ecosystem. The third type is nothing special, we all do that practically every second.

In general, a platform ecosystem is built on existing components – cloud, information systems, APIs and so on. As it is a rich system for others to participate and requires a carefully thought-out business idea, solid structures (such as overall system architecture), definition of roles, division of work, control and power and so on.

All that support the integration of the services into whole that works fluently for the vendors and customers alike. A platform also resembles a community such as open source communities, and those are traditionally difficult to build. Of course, any commercial platform has the economic incentive of joining it. All in all, creation of a platform can be a very demanding task, but still a modern alternative for building a supplier network. It is, however, just a rearrangement of sharing, innovation and deployment logistics, but at best represents a change of mindset from control to enabling.

Tiwana et al. (2010) define the core concept in softwarwsoftware platform-centric as the following:

- Platform: The extensible codebase of a software-based system that provides core functionality shared by the modules that interoperate with it, and the interfaces through which they interoperate.
- Module: An add-on software subsystem that connects to the platform to add functionality to the platform.
- Ecosystem: The collection of the platform and the modules specific to it.
- Interfaces: Specifications and design rules that describe how the platform and modules interact and exchange information.

- Architecture: A conceptual blueprint that describes how the ecosystem is partitioned into a relatively stable platform and a complementary set of modules that are encouraged to vary, and the design rules binding on both.

The interfaces, are currently called shortly APIs, even though many of those are not strictly that. For ease of integration of the various activities in digital platforms, APIs are in central role:

- Retail platforms needs APIs that companies can link into their inventory systems.
- Software platforms are built on APIs for integrating the third party components. Besides that, Software Developments Kits are often provided and used.
- Service platforms in the cloud again need APIs that integrate the various subsystems together and to the tools the customers use locally (such as any software development environments).
- Geographical data services obviously need to provide an API for querying the data for presenting maps and/or integrating other information to the data sets.
- Platforms that mediate work need APIs for linking with resource management tools, personal "marketing tools" such as LinkedIn and social media systems.
- Social platforms obviously need APIs that can be used for publishing and searching information as part of other platforms and APIs that allow for integrating third-party services – starting with marketing.

In a broader sense, platform-specific programming languages are also an API as much as any platform libraries. From history it is known that difficult languages sometimes greatly hinder developers joining the ecosystem (the now extinct Symbian is famous for that).

Development of good APIs should be nothing new. That is what programmers do and assessment of programs built upon internal APIs is the very core tradition of software engineering. But the APIs need to work in the broader architecture and support decomposition, modularity, and design rules (Tiwana et al. 2010).

From the quality perspective, platform economy has interesting practical issues.

- Quality of the platform concept, including analysis and simulation and assessment of customer and user experience.
- Quality of the infrastructure, including system architecture, infrastructure software choices. Architectures are critical and need to be subjected to thorough evaluation.
- Security.
- Quality of the participating tools, including communication tools, development tools, integration tools.
- Quality of the APIs: usability, security, extensibility, compatibility with external systems and so on.

In the quality assurance the very ideas of an ecosystem provide interesting challenges. One of those is the issue of shared ownership and also control. It may be difficult to define who is responsible for quality, who should coordinate testing, who can demand quality from which partners and so on. That call for understanding the overall platform, own business and its responsibilities and most of all a customer-centred view to the quality of the whole. Customers don't care how the mash-up is created, they just wish to get value for their money and time and whoever in on the frontline, must respond. That responding starts in the planning of the participation in the ecosystem and own quality assurance and ends in responding to the problems that arise. Being responsive will often require good monitoring of the system, as it is vulnerable to other parties' actions. Monitoring then results in actions against any anomalies with diagnostic testing and testing of any corrective measures.

Because a collaborative platform may be new to a company or even on the domain, the slightest errors may be critical. Thus, the quality requirements are high. That makes things even more demanding.

Change-competence snippet 15	Platform economy and API economy
Change caused by -> enables	Managed products, raise of abstraction level, need for added value while limited internal resource -> more added value, growth
Competence implications (re: quality and testing)	Understanding overall contexts #U Business understanding #O #U Customer-centredness #O #U #A System and system of systems thinking and testing #U #A Information systems and integration competences #O #U #A Architecture evaluation #U #A Risk thinking #U Understanding permission, security, privacy #O #U Security assessment and testing #A UX and usability testing #O #U #A Quality advocacy #O #U #A
Links with	<- Digitalisation <- From products to services <- Industrial Internet <- Big Data -> Small inexpensive apps -> Business understanding for all -> Modern risk management -> Machine industry turning into software industry

5.4.5 The startup phenomenon

During the recent years, the emphasis of ICT has turned away from huge companies and their subcontractor networks to smaller companies that develop their own products and independent business. The main driver for this was the crisis of Nokia, but that can be seen just as a trigger that released forces that were already growing. Obviously, there are many kinds so that bundling of them up to one class is problematic. Some of them are founded by experienced experts who have all the competence and processes in good shape, some of them are more of a craft shop, perhaps a coding-oriented firm, and the sizes also vary – some start with the entrepreneur and some with a larger staff. Still, they are all startups, with their idea and a challenge to get the business started.

Giardino & Paternoster (2012) made a thorough study on startups and describe their challenges as follows (direct quote):

- The most urgent priority of software development is to shorten time-to-market.
- Startups do not apply any standard development methodology: the closest development approach undertaken by early-stage startups tends towards the Lean startup methodology.
- The greater part of engineering activities of startups is focused on the implementation while only little attention is given to more conventional activities (project management, requirement specifications, analysis, architecture design, automatic testing).
- The first release of the product includes only a limited set of well suitable functionalities focused on user experience.
- Engineering activities are supported by low-precision artefacts and collaborative tools integrated with the development environment.
- Characteristics of the founding team are the most important determinants of speed.
- The initial lack of structures and processes negatively affect the performances when the company starts growing.
- Startups bring the first product to the market in a very short time and practitioners are satisfied with the adopted software development strategies.

Of the issues in the list, test automation will always raise questions. Startups often emphasise test automation, but it is generally known that test automation is not the best focus point when everything is changing all the time – just like it usually should be in a startup until they understand their business idea and their product. Before that, focus should be on understanding the customer and exploratory testing should be preferred in the rapid testing of the new and changed features, but after that, the technical infrastructure needs to be developed to support growing of the business and the product.

It should be noted that this describes the startup culture as it is – not as it could be. After all, it is a quite new culture and most likely something that will evolve. Definitely one of the evolutions is the idea of internal startups; a "company within a company". Some large companies are currently (as of 2016) trying this as a response to their normally slow and tradition-tied operating models and cultures. This development is aided by some good experiences and available guidance (Märijärvi et al., 2016). From the quality-creation perspective this should be a good setting, as there is the parent company's expertise and technical infrastructure available, and there are also more expectations for quality due to the brand of the parent company, unless the internal startup is branded separately.

Now, the question is how does the role of “tester” fit into a startup company? One often stated premise is that startups do not have testers, but that fact does not mean that they should not have testers. A tester is not necessarily a job post, but an orientation of some person. Here are some ideas of good characteristics of that role by the author.

Attitudes are important. For a startup, the first customers are more valuable than gold. There is nothing without them. Therefore, the tester must also be very customer-oriented. She must do and test what is important to the customer in any situation. It is a question of the whole product, not just the “application” part of it, in other words also of the quality of the services, their quality and efficiency need to be assured. This also means an emphasis of the company’s own processes.

Utilisation of resources is critical, as there are so few of those. When there are not many people, one must be a versatile expert. "All what one must know, one must know" – and also do.

Testers need understanding about the whole business. Because there are not the time and much hands and everyone is very near the business, that business must be well understood so that it will be known what is important and what must be prioritised. Still, one must restrain oneself from “mastering” the business and the entrepreneur, but be a strong sparring partner and questioner.

Maintaining the dialogue in the company is one task for a tester as a producer of information. A startup is a big question of the hope. The testing must help the becoming of the hope. The tester must work so that the company is prevented from shooting itself in the foot. Therefore, there must be different viewpoints and approaches between the tester and the managing director. The passion and the motivation can be a totally common resource.

Everyone must bring something special to the competence pool. The special testing competence must be genuine core know-how and not for example information about the process models of the bigger companies.

When there are no experienced test managers around, the tester must have initiative and self-guided. Supervision must not be expected. Initiative and creativity and introduction of new ways of action (there may be none at the start) represent a high level of competence.

A tester can bring quality leadership to the company. No matter what the organisation model is like, somebody must maintain quality leadership in the daily pressures and so the courageous testers have their role in here. Actually, the quality leader should be the entrepreneur, but she can be more like an inventor or marketer by nature.

Operation models. The companies must always find their own ways to work and a startup can find its excellence also through them. The operation models must be light and must grow only as when the demands are raised – not by standards. Someone must take care of that growth of maturity. A quality-conscious tester is an important asset when the company takes the steps toward growth.

If and when a startup uses the Lean Startup method or some other such method for the clarifying of the customers' needs, wishes, behaviour and preferences, it is good to be somewhat a researcher. In other words, one must think how testing and experimentation are brought into every interaction with customers.

The majority of the beginning firms are not very long-lived. It means that even the personal risks are high. So the tester must really work for what the company needs to succeed, not just do testing.

The tester’s own identity requires collegial relations. There may not be another tester in the same company. So one should network and join tester communities.

So it is possible to use many kinds of skills which does not mean a need for universal geniuses but that, the skills and operation styles which are found, one should not make under the bushel because something stereotyped imagines. Still, one particular thing requires notice. Giardino & Paternoster (2012) note that user experience is just about the only product quality factor that matters. That emphasises that the key personnel should not only include the skills of user experience design but also skills of user experience assurance – doing evaluations, carrying out customer preference tests, conducting user tests. The aim here is not only finding out if the product is “good”, but more importantly, if it is being developed in the right direction. That brings us to the concept of Lean Startup.

The Lean Startup approach (Ries, 2011) is based on this. In the context of this dissertation its main characteristic is that every new system version is an experiment and thus the testers need to be involved in planning how to gather new information from every encounter between customers and systems. The idea is to learn by developing new versions and to actually validate the learning by experimenting with users. The validation may be done by observing users or by making measurements of their use of a system. This requires experiment design, implementation and analysis skills that resemble the skills of a scientist. When we consider the competence sets in the occupations in companies, testers are obviously the ones that should have similar competences. Often testing just proves the quality of something “as such” and not compared with something else. Here we need to be able to show the differences between the product and the previous version or alternative versions. Such comparison tests are sometimes done in the usability and user experience cultures and as user experience is essential to startups, the user experience assessment competences seem to be very critical, but may need to be further developed in the areas of for example metrics usage. Obviously, configuration and deployment management are essential – we need to know who is using what so that the metrics are reliable. Those competences should be found in some other team members – usually the developers. To make all this work requires very intense collaboration, pointing out the importance of team work skills.

Change-competence snippet 16	The startup phenomenon
Change caused by -> enables	Changing society, need for innovation -> new companies, new products, new economy
Competence implications (re: quality and testing)	Focusing and prioritising actions at each startup phase #U #A Working under insecurity and change #O #U #A Business understanding #O #U Multi-skilledness #A Personal competence development #A Process development #O #U #A (when changing into growth mode) Understanding of possibilities and alternatives in testing #U Team skills #A Creativity #A Right timing of actions #A Making compromises in quality #O #U #A Comparison testing #A UX testing for feature development #O #U #A
Links with	-> Effective work in small, smart companies -> Finnish style challenged -> Agility and flexibility -> Need for new types of workers -> Quest for multi-skilledness -> Business understanding for all -> New thinking on defect costs during application lifecycle -> Rethinking the goals of testing and quality assurance

5.4.6 The rise of the game industry

The game industry has become a new important domain in the software industry as Finnish games have had a streak of success that the domain seems to have followed with a lasting successful industry. Neogames, the hub of Finnish game industry, has estimated (Neogames, 2016) that in 2015 the Finnish game industry had a turnover of 2400 million euro, which was around 25 % of the turnover of the whole ICT domain.

But isn't it software product development such as any other? Yes and no. It has similarities to many other development contexts, but some special characteristics that make it essential to think about it seriously.

Games are an area where user experience is clearly emphasized and the essential element under that is "playability" (Korhonen, 2016). According to Korhonen the games should be developed using user-centered design, the principles of which are described in the ISO standard 9241-240 (ISO/IEC 9241-210, 2010), in Finland also published as an SFS standard SFS-EN ISO 9241-210. That situation emphasizes user experience design skills and skills of evaluating and testing the user experience, including the all-important playability. The usability culture is known for using and recommending analytic evaluation methods, such as checklists and heuristic evaluation using heuristics list such as the classic by Nielsen (1993 & 1994). Those are very relevant at all levels of assessment, from concept to details. Game industry is however, traditionally a craft industry where design is largely tacit in nature. Creativity is emphasized, just as being part of the youth culture. Thus, systematic assessment methods are not used as widely as they could be for maximum benefit, and that is one cultural opportunity. Note that the characteristics mentioned don't work against proper automated technical testing, such as unit testing and any testing done in the continuous integration process or in any automated publishing process. The settings in game developer companies emphasise the need for having creativity and systematic approaches can co-exist and doing, managing and developing that is not trivial.

Because of the age of the industry, the companies are also often startups or small sized, which makes them susceptible to the same problems as any other similar companies, such as the need to be innovative while managing any technical or other debts, to later manage growth and product line expansion, to handle constant change and so on.

Korhonen (2016) who studies using expert review for games, notes that it is not often mentioned in game design literature. Note, however, that the situation is often the same: design literature just doesn’t mention assessment or testing, yet, those exist in the reality and are written about separately. Practitioners just need to combine in their practices and workflows the disciplines of design and assessment. Practitioners can't afford the narrow focuses of researchers or consultants.

When it comes to using heuristic evaluation (or expert review) for user experience or playability, Korhonen (2016) notes that there is no common description for what playability consists of in games and what the heuristics would be. Therefore, there would be a need to transfer a company's domain knowledge into the assessment tools, which would absolutely be a very beneficial exercise in knowledge transfer and could be recommended for all domains. It makes sense to base such domain-specific list to an existing one (Korhonen, 2016, lists those) and tailor it based on in-house design vision. An in-house heuristics list would provide clues to the question every designer or testers should be able to answer: what makes this kind of product fantastic? It is however critical that any tools exposing the design ideas cover the whole "development stack" from conceptual key ideas to the usability of the implementation.

The assessment practices and small tool developments should be considered, as the competition in gaming is so fierce that games need all the help they can get for a breakthrough. It is already a common saying that a game company was an overnight success after tens of failed products!

Note that many prefer calling the heuristic evaluation expert review, because it cannot be carried out by just anyone. There needs to be expertise in using the methodology and the heuristics applied, but more crucially in the domain of the game type evaluated. For games, this is especially important, because games are so rich and their conventions are so tightly tied to the game culture.

There are also other challenges. Mobile games are sold at a very low price or even free, with the sales coming from in-app purchases. That has meant that there should be very little support costs, which means that the games need to be reliable and have no compatibility problems with the various variants of devices used. Traditional technical testing should therefore be as good as in any other industry. The compatibility issues are hard to test and the manufacturers of mobile phone games should be able to use for example cloud services where automated functional tests can be run on a wide range of devices, such as many different Android versions on different hardware. As the games are interactive in nature and have various interactions between elements, good exploratory testing skills are a basic requirement.

Obviously, when the games have in-app purchases or include third party advertisements, security or the games is essential. Games that use a well-known game platform as their basis have a better starting point for security, but others need to pay a lot of attention to security risk analyses and testing. Similarly, especially multi-player games should have no flaws in privacy. The visibility of other people's information through the game and its related systems should be evaluated. Access to the information in the player's device may not be a privacy risk indirectly as it opens more risk for the possible vulnerabilities in the game and its infrastructure. Some access to personal information can be desirable for making multi-player games social, so this is an area of careful management.

Many games have a server component and in the development of that, general quality management and testing competences are required. Indeed, the technology stack can be impressive, ranging from multi-device synchronized state management to augmented and virtual realities, so there can be a lot of technology to manage.

In summary, games are user experience -critical and benefit from being develop as such. But at the same time they have several, large areas of important characteristics that really form a challenge to the companies.

Change-competence snippet 17	The rise of the game industry
Change caused by -> enables	Global open mobile device software market -> new companies, room for innovation
Competence implications (re: quality and testing)	Broad flexible competence #O #U #A Multi-skilledness #O #U #A Team skills #A Multi-skilledness #A Role finding #A Understanding users #U Understanding quality and its practices #U Open-minded quality thinking #O #U UX and usability testing #A Collaboration skills #U #A Understanding permission, security, privacy #O #U Security assessment and testing #A Forming and promoting practices #A Working under insecurity and change #O Business understanding #O #U Configuration testing #A Making compromises in quality #O #U #A
Links with	<- The startup phenomenon -> Finnish style challenged -> Need for new types of workers -> Rethinking the goals of testing and quality assurance -> Information security and privacy -> Cultural competences emphasised -> Small inexpensive apps -> New technology products -> Need for personal understanding of quality

5.4.7 Changes and implied competences for this section visualised

A visualised summary of the change-competence snippets in this section are visualised in Figure 41.

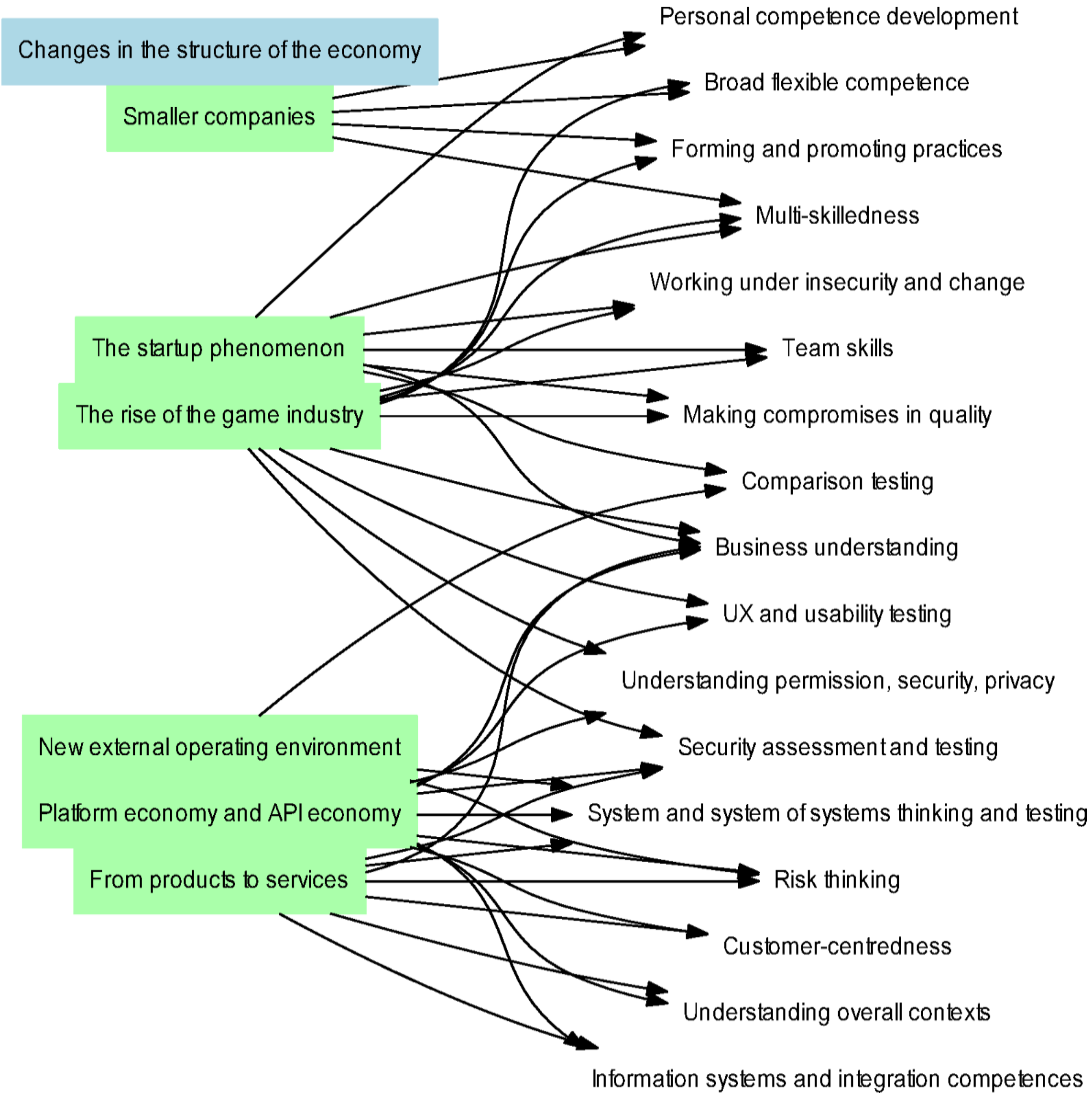

Changes in the structure of the economy
Smaller companies
The startup phenomenon
The rise of the game industry
New external operating environment
Platform economy and API economy
From products to services
Personal competence development
Broad flexible competence
Forming and promoting practices
Multi-skilledness
Working under insecurity and change
Team skills
Making compromises in quality
Comparison testing
Business understanding
UX and usability testing
Understanding permission, security, privacy
Security assessment and testing
System and system of systems thinking and testing
Risk thinking
Customer-centredness
Understanding overall contexts
Information systems and integration competences

Figure 41. Visualised summary of the change-competence snippets in this section. To make the graph fit the page with legible font size, only the competences are included that are associated by at least two changes in this section.

5.5 Changes in some businesses

5.5.1 From Western engineering culture to new Finnish culture?

The processes and many of our practices and cultures represent western ideals of excellence. Nonaka and Takeuchi (1995) have compared key elements of Japanese and Western professional cultures and their analysis can help us understand the barriers to agile in engineering companies.

Table 19. Comparison of Japanese-style vs. Western-style organisational knowledge creation from (Nonaka and Takeuchi, 1995).

Japanese culture	Western culture
Group based	Individual based
Tacit knowledge oriented	Explicit knowledge -oriented
Strong on socialisation and internalisation	Strong on externalisation and combination
Emphasis on experience	Emphasis on analysis
Dangers of “group think” and “over adaptation to past success”	Danger of “paralysis by analysis”
Ambiguous organisational intention	Clear organisational intention
Group autonomy	Individual autonomy
Creative chaos through overlapping tasks	Creative chaos through individual differences
Frequent fluctuation from top management	Less fluctuation from top management
Redundancy of information	Less redundancy of information
Requisite variety through cross-functional teams	Requisite variety through individual differences

The characteristics associated with Western culture do not necessarily favour modern agile development or organisational cultures or practices. But as we see those practices often succeeding, is our culture changing in deep level? Or do we see companies with truly differing cultures even after the current small and medium-sized companies grow?

In any case, the processes and practices are developed to be more agile, we need to address the cultural issues. Conflicts need to be identified and practices be chosen so that they suit the company culture, or the company culture also needs to be changed. The changing of culture can be hard. For a new culture, it is easier to start a new company. Table 20 presents more details to the cultural comparison that help us understand the challenges.

Table 20. Comparison of Japanese-style vs. European-style product development of high-end cars (Nonaka and Takeuchi, 1995).

	European-style	Japanese-style
Goal	Pursuit of superior performance	Adaptation to changing needs
Product appeal	Function (e.g., high-speed performance)	Image and quality
Product concept creation	Clear-cut decision at the initial stage, adhered to throughout the ensuing stages	Vague at the initial stage, modified and altered in ensuing stages in accordance with changes in needs
Flow of activities	Sequential approach	Overlapping approach
Ensuing process	Specific design targets fixed at the initial stage are pursued under a strict division of labour	Close cooperation among all departments concerned during the development
Organisation	Organisation according to function and often under a project leader with limited authority	Matrix- or project-team-type organisation under a project leader with authority over the entire process from planning to production to sales
Strengths	Conductive to a relentless pursuit of superior performance, function and high quality	Shorter lead time (3-4 years), high quality, and attuned to the needs in the market
Weaknesses	Longer lead time (7-8 years), high development costs	Risks of compromise on a low level; not conductive to an all-out pursuit of superior performance

For testing and quality management competences, these seem to be essential:

- Cultural adaptation skills – ability to identify cultures and to work in them.
- A personal toolbox than can be used in situations that vary culturally.
- Ability to work in such ways that boost the good characteristics of the culture at important situations and help avoid its risks. For example, if a culture is experimentation-centred, the testers need to be able to do experiments properly to avoid the risk on invalid experiments.

Change-competence snippet 18	Finnish style challenged
Change caused by -> enables	Stereotypical Finnish working style not sufficient in all conditions -> versatility in working
Competence implications (re: quality and testing)	Reflection on working styles #U #A Handling contradictions #U Cultural skills (national, occupation, domains) #O #U #A Cultural adaptation #A Broad flexible competence #O #U #A Multi-skilledness #O #U #A Collaboration skills #U #A Social skills #A Team skills #A Role finding #A
Links with	<- Changing working life -> Cultural competences emphasised -> Need for new types of workers

5.5.2 Competence requirement differences in various types of business

Traditionally, it has been noted that the requirements for testing may be different in product business and in developing tailored information systems on a project basis, and also in offering testing services. Here we briefly analyse the possible differences and also how they have been changed recently.

Traditionally, product business is based on getting “releases” to the market on some rhythm, say, every year. Many companies have turned their releasing strategy into more continuous style – because of agile processes enable that and because the market expect more continuous added value. In this regard, product business is more resembling project business where close an iterative working with the customer is the norm.

In testing service business, the style of working needs to reflect the needs of the current customer and the project. This applies to most elements of activity. Still, project business does not have a “customer” – just the market – although some key clients may be listened to more carefully than the others.

In project business and testing services, the customer chooses their partners. That requires proof of competences, which is by nature different than in a product organisation.

For a company’s own products, the company has (in theory at least) freedom to choose product and development and testing technologies. In projects and services, the customer may have a say in that. Usually, in testing services we need to be very versatile and be able to serve many kinds of customers with many kinds of technologies. This requires versatile technological skills and ability to rapidly adapt to new customer requirements.

Related to that is the issue of scaling. Products tend to be of similar scope, size and criticality in a company and that applies to project business too. Testing services have some variation, but service teams may be specialised in some kind of services for certain type of systems. Still, there are big and large projects that need to be handled properly.

Cultural environment can vary. Products are created in a company's internal culture. In projects, we may meet a very different customer, in a different field than the ones we have been working in. This requires cultural-adaptive skills. The same applies to testing service provision.

Dependability requirements are the same in all types of business. Testing must be something that internal or external customers can rely on.

The element of chaos. Some product companies may live in a continuous chaos, which may result for immaturity of even a conscious choice to enhance creativity. That is something that cannot be tolerated in service provision or in project work.

The main differences in the types of testing are collected to Table 21.

Table 21. Differences in various types of software business that influence testing.

Element	Product business	Project business	Testing services
Release mentality	Strong	Varies	Varies
Service approach	–	Strong	Strong
Agility	Strong	Strong	Varies
Proof of competences	–	Strong	Strong
Technological versatility	Weak	Medium	Strong
Ability to adapt	Weak	Medium	Strong
Scaling	Medium	Medium	Medium
Cultural skills	Medium	Strong	Strong
Domain knowledge	Strong	Varies	Varies
Dependability	Strong	Strong	Strong

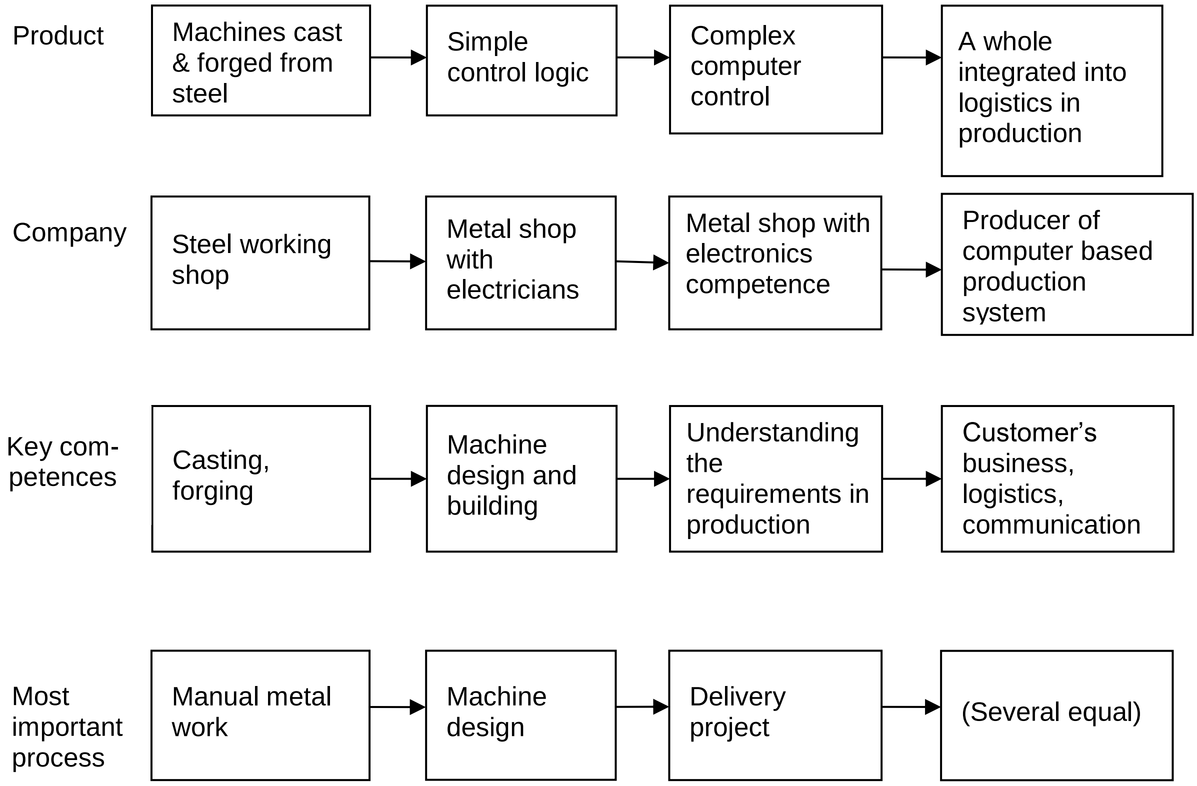
Element	Product business	Project business	Testing services
Possibility of chaos (or strong dynamism)	Strong	Weak	Weak

Based on the brief analysis – yet sufficient for this stage – the main differences seem to be few. Two important areas are service approach and mentality. Even those are changing in product business, because the ability to bring added value better than the competitors requires a strong will to understand and serve customers. That amplifies the need for lean startup type action and related testing. Another area is technological versatility. Product companies can handle their technologies more than the others, but this also relates to platform changes and similar issues. So, the key message here is to have a strong element of technology independent competence that can be adapted either in projects or changing business rapidly. In addition, there is the possibility of chaos. Product companies can have that luxury and that has been a trend currently, when maturity is out of fashion and agility and speed are emphasised. For testing, that means two things: 1) testing is a means to make the chaos tolerable, and 2) testing must adapt to the chaos.

Change-competence snippet 19	Competences focused on business type
Change caused by -> enables	Different businesses need different competences, business innovation -> effectiveness, good business
Competence implications (re: quality and testing)	Business understanding #O #U Understanding about competence #U Competences usable in various process models and contexts #A Broad flexible competence #O #U #A Multi-skilledness #O #U #A Cultural skills (national, occupation, domains) #O #U #A Competence development focused on business needs #U #A Understanding of possibilities and alternatives in testing #U Understanding overall contexts #U Risk thinking #U Personal competence development #O #U #A
Links with	-> Business understanding for all -> Quest for multi-skilledness -> Explosion of important quality attributes -> Cultural competences emphasised

5.5.3 From mechanical machine industry to intelligent software-driven machines

All business domains see transformation where the changes in environment and product technology turn the business into something very different. Sometimes industrial product business turns into service business (as in the case of elevators), sometimes the characteristic of the products change so that a company will need to rethink its operations and its identity. An example of this is the world of machine builders – one backbone of Finnish industry. The products were once seen as blocks on metal components working together, but with the introduction of computing and information systems, they have become more like logistic process controllers. The author analysed the change factors in that area in 2010 and the main changes are shown in Figure 42.



Product	Machines cast & forged from steel	Simple control logic	Complex computer control	A whole integrated into logistics in production
Company	Steel working shop	Metal shop with electricians	Metal shop with electronics competence	Producer of computer based production system
Key com-petences	Casting, forging	Machine design and building	Understanding the requirements in production	Customer's business, logistics, communication
Most important process	Manual metal work	Machine design	Delivery project	(Several equal)

Figure 42. Changes in machine building companies.

Changes like this happen gradually and they are noticed almost too late, requiring fast corrective action. Before that has been done, the products do not fulfil their potential or do not have the quality expected. The corrective action will include rethinking of the whole development system, including the organisation and processes for developing ICT elements in the products, new testing and QA processes, hiring new experts etc. There will be developments like that in many domains. Still, there are and probably should be domains that have different cultures and relationships towards technology.

When a company is for example, engineering driven, that “drive” is everywhere, in every activity. Likewise, when a company develops consumer applications, that orientation needs to be embedded in every person and every process.

That means that the new software intensive culture in a machine builder company should be different than in a media company. That difference reflects itself in the way quality is managed – what the quality means? What strictness is needed for success? How do we approach new technologies?

Change-competence snippet 20	Machine industry turning into software industry
Change caused by -> enables	Software controlled machines, production control, integrated information systems -> more added value into products
Competence implications (re: quality and testing)	Changing company-level competence profile #O #U #A Generic software quality and testing competences #O #U #A Process development #U #A Information systems and integration competences #A Business understanding #O #U Understanding overall contexts #U System and system of systems thinking and testing #U #A Understanding information security risks #O #U Security assessment and testing #A Data analysis #U #A
Links with	<- From products to services <- Platform economy and API economy -> Business understanding for all -> Industrial Internet

5.5.4 Changes and implied competences for this section visualised

A visualised summary of the change-competence snippets in this section are visualised in Figure 43.

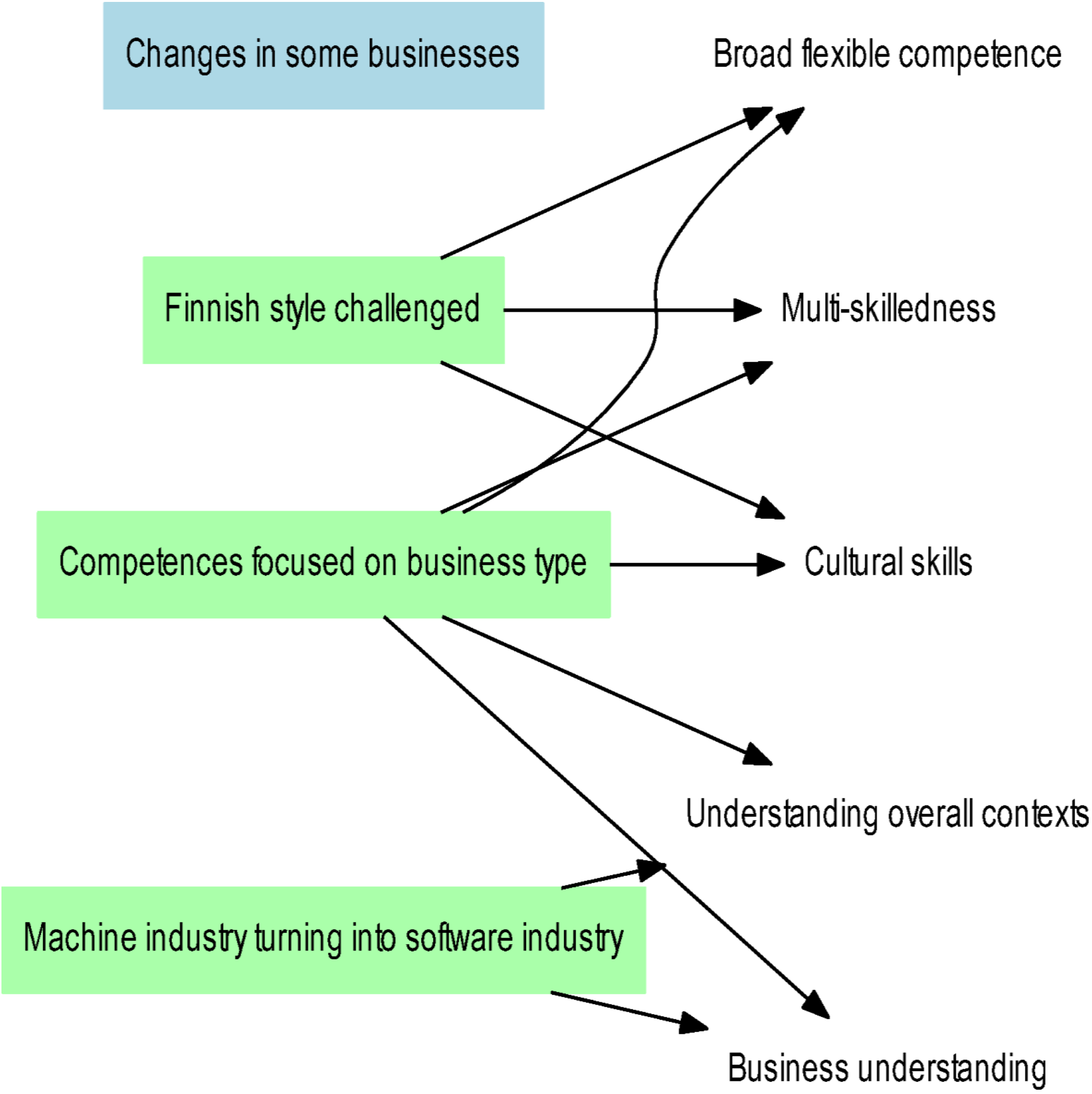

Changes in some businesses
Broad flexible competence
Finnish style challenged
Multi-skilledness
Competences focused on business type
Cultural skills
Understanding overall contexts
Machine industry turning into software industry
Business understanding

Figure 43. Visualised summary of the change-competence snippets in this section. To make the graph fit the page with legible font size, only the competences are included that are associated by at least two changes in this section.

5.6 Working style in companies

5.6.1 Companies' internal operating environment

Our traditional view to organisations has been to see them as large, with many units, plenty of people and plenty of support – and bureaucracy. That has changed and there is a trend toward smaller companies. This means a change towards:

- Smaller units.
- Just enough people for any given function or project.

- No separate QA organisations, but QA functions are integrated into project organisations.
- Less money to spend – must be very effective.
- At the same time a quest for effectiveness and demand for better quality.

These are simple changes, related to the resources of a company. There may be other changes more radical. One of them is holacracy (van de Kamp, 2014), which some new companies are reporting to adapt in 2015. It aims to help in the creation of agile and adaptable organisations and purpose-driven work. Its main ideas are:

- The organizational structure consists of self-organizing teams, called 'circles'.
- Individuals do not carry job titles, including management titles.
- Roles are defined with a clear purpose where they contribute to the organizational purpose and the aim of the circle.

Some of the changes were seen in a small study made in order to understand the phenomena in startups and small companies (Dande et al. 2014). In that study, patterns were identified that would characterise the unique ways of working in such companies. Patterns relevant to this discussion were, e.g., the following that show how companies look into empowering the workers being more relevant that traditional management heuristics.

- Flat organization [pattern #3].
- Bind key people [#67].
- Consider career expectations of good people [#52].
- Start with a competence focus and expand as needed [#59].
- Start with small and experienced team and expand as needed [#64].
- Outsource your growth [#12].
- Anyone can release and stop release [#36].
- Create the development culture before processes [#54].
- Small co-located teams [#2].
- Let teams self-select [#56].
- Sharing competence in team [#58].
- Have multi-skilled developers [#53].
- Keep teams stable in growth mode [#55].

Deeper that this goes a case study of Vincit, a quite new company that has got publicity for its characteristics in culture and management. Ylén (2015) focused on the main activity patterns of the company and discussed how they promote human agency. There were five distinct patterns. (1) Democracy as a practice was shown in a very independent workforce. There are directors, but mostly for official purposes. Workers do not have dedicated superiors, for example (besides what the project team dictates). All workers are equal. Most have even made their wages public. As an effect, democracy forces (2) voluntary personal development. Each worker must take responsibility for her own professional development instead of relying on courses offered by the company. This includes possibilities to choose what project one participates in and thus advancing own experiences towards own preferences and goals. Any such personal responsibility is aided by good motivation, and “vincitians” think that they are passionate and the company favours such people in recruitment. (3) Independent project teams can choose their practices, roles and everything else as they wish. This is in contrast to traditional companies that wish to harmonise practices to gain efficiency and quality. The choosing of practices and technologies is part of (4) shared experimenting, which is a traditional characteristic of learning organizations. The final practice (5) customer projects as a frame for human agency is a somewhat obvious-sounding thing in the world of software development, but it also emphasizes the focused activity in the company. The empowering principles sound humanistic, but at the same time they form a business strategy, where good customer-centred quality and business success is achieved by letting motivated and skilled personnel do what they think is best. The company admits that a culture like this is not for everyone and from the quality point of view, all the freedom would require people who have a mature attitude towards quality, good skills in designing and testing for it, and a solid quality culture in the company from the start. If and in what extent these kind of changes occur, they will bring a new kind of organisational environment to testing too. Testing will need to be “sold” to colleagues, good communication is essential and in general everything will be more dynamic.

Gary Hamel has always been insightful in these matters – and also provocative. In the foreword of a book about open organisations he lists characteristics of an organisation that is fit for the future (Hamel, 2015). The author grouped and re-arranged the characteristics in following way to make the list more digestible:

- Democracy: Leaders will be chosen by the led. Compensation will be set by peers, not bosses.
- Influence, status and recognition: Contribution will matter more than credentials. Influence will come from your value added, not your title. Individuals will compete to make a difference, not to climb a pyramid. Every idea will compete on an equal footing.
- Practices: Experimentation and fast prototyping will he core competencies. Decisions will be made as close to the coal face as possible. Control will be achieved through transparency and peer feedback.
- Organisation structures: Communities of passion will be the basic organizational building blocks. Structure will emerge only where it creates value and disappear everywhere else. Organizational boundaries will be porous. Resources will be allocated with market-like mechanisms.
- Collaboration: Coordination will occur through collaboration, not centralization. Commitments will be voluntary. Lateral communication will be more important than

vertical communication. Strategy making will be a dynamic, companywide conversation. Change will start in unexpected places and get rolled up, not out.

- Attitudes and mental patterns: Everyone will think like a business owner, and be just as accountable. “Why” will matter more than “what.”

The ideas in Vincit seem to have much common with what Hamel proposes and also quite well match the views into Finnish working life presented earlier by Alasoini, Järvensivu & Mäkitalo (2012).

One thing that effectivity of organisations requires is more holistic thinking of activities. For example, when thinking about developing testing, we must not think of optimising testing as such, but optimising the overall activity. It has been understood for some time, that and organisation may have a set of optimised processes, but the whole is not optimised. An optimised whole requires sub-optimal parts!

Therefore, process development and testing require understanding about the whole and an attempt to systems thinking: how the overall system works, what are the interactions, what will any changes at one process cause at others? The agile culture is by its integrated nature, supportive of this. This is more a mindset issue than something that could be tied to any specific activity.

We may see how all the new management and cultural ideas work out, also considering that the companies who are trying this are young and at a growth phase and later phases in their life may bring changes to the organisational design. It is also easy to think that many of the new ideas are a re-bounce from the years when the founders were working in large companies that had sometimes opposite practices and values.

Change-competence snippet 21	Effective work in small, smart companies
Change caused by -> enables	Changing world -> speed, effectiveness, innovation
Competence implications (re: quality and testing)	Process development #U #A Managing change with information #A Quality advocacy #A Active, self-steered working for quality #A Collaboration skills #U #A Team skills #A Networking skills #A Role finding #A
Links with	-> Need for new types of workers -> Finnish style challenged -> The startup phenomenon

5.6.2 Changing social systems

The social systems where testers work are continuously changing. Traditionally, testers have worked in line organisations and work groups. We use the term “group” intentionally, as this unit of organisation has rarely been a true team.

Testers have often been seen best to organise into separate groups, but recently their integration to the development teams has increased. At the same time, international collaboration with other units and subcontractors has increased, as well the internationalisation of Finnish workplaces. A new phenomenon is the networking of professionals in social media – discussion groups, blogs, etc., which opens up the interaction of testers tremendously. All this leads to growing requirements for social competences, such as intercultural competences, understanding social media and media reading skills, team skills, general “people skills” and language skills.

Change-competence snippet 22	Testers in development teams
Change caused by -> enables	New social systems in companies -> more integration, fast feedback
Competence implications (re: quality and testing)	Social skills #A Team skills #A Role finding #A Quality advocacy #A
Links with	<- Changing working life

Change-competence snippet 23	Networked communication
Change caused by -> enables	Networked, dynamic industry -> external social systems, information flow
Competence implications (re: quality and testing)	Social skills #A Networking skills #A Communication skills #U #A Understanding information security risks #O #U Understanding domains, contexts and situations #O #U
Links with	<- Pervasive communication -> Information security and privacy

5.6.3 Experimentation culture

There are signs that companies are moving towards a more experimenting culture. Popularity of the Lean Startup (Ries, 2011) method is one clear sign of that. A/B testing is another strong phenomenon. In Finland, books are appearing about experimentation, exemplified by Hassi, Paju & Maila (2015). The goals experimentation can be varying. In Lean Startup, the goal is to understand the customers' need by rapid experimenting, something that also traditionally was done by prototyping. But Lean Startup is more open-ended than the traditional practices, not based on a requirement specification or design concept to validate, but questions and hypotheses. Thus, it supports innovation and general sensemaking of a situation. The A/B testing in live systems is more about optimizing designs. Companies often advertise innovation sessions (by various names; most often hackathons) and where they try gain ideas for implementing new technology or find radical solutions to a business problem in session with technology providers or students in a couple of days instead of a much longer systematic internal R&D process. There are now plans to bring such experimentation into engineering curricula (Systä et.al 2016).

But particular methods are just add-ons may not the be properly integrated to the overall activity system and development processes. Fagerholm et al. (2014) propose a holistic model for a continuous experimentation infrastructure built around build-measure-learn blocks and which spans vertically all levels from vision and strategy to instrumentation of products. Holmström Olsson, Bosch & Hiva Alahyari (2013) see the movement to experimentation as a transition from traditional agile development into a system based on making lots of experiments on productions systems, utilising continuous deployment and immediate reaction to user feedback. Holmström Olsson & Bosch (2015) continue the development of approaches and describe a conceptual model for continuous validation of customer value based on a novel idea of hypothesis backlog. They all are approaches for building an integrated development system where all parts work for and are optimised for the purpose, but are also based on certain assumptions about the best overall mechanism of product development that may or may not fit specific circumstances.

The side-effects of experimentation and experimentation capability are various. One is an improved agility to respond to any changes. Experimentation also moves the organization towards a learning organization – a goal from already decades ago. It will also make the workplaces more dynamic and attractive to young professional.

There are many contexts for experimentation about the products. Experimentation can be done during system development (Lean Startup, various levels of prototyping – paper prototypes, mock-ups, functional prototypes, any trials during the fuzzy front end). Private and public beta releases are usually used also for experimenting how clients respond to features. Experiments can be done with product versions already on the market (A/B testing). New technical entries can be created in the hackathons and turned into marketable products by startup companies or internal startups in large companies that may be disruptive spin-offs of the regular product line and brand. The startups are often really based on experimenting, as in the beginning they are in a state of not knowing what they should do and experiments are the way to find out. On process level, experimentation can be trying new design practices and so on, those perhaps leading to internal standardization until replaced by other practices found by experimenting. This is also how Lean works.

One difference to previous times is that experimentation was done in special R&D units, but now there are less of those and innovation is integrated with product business units, the development teams take responsibility of it. Indeed, everyone can take part in experiments.

Experimentation shows a new mindset where the organization is open for ideas, failure is not seen as an error and actions are done also for learning and people's potential needs to be untapped. This is in contrast to the stereotypical "year 2000" culture, where experiments are minimized as they cost money, experiments only at necessary points in process (perhaps one prototype); there is no attempt to learn, just to validate and accept ideas and designs, experiment design is lacking, failure is abnormal and problems are errors and the main attitude is to understand first, then experiment in order to validate the ides – vs. experiment first, then understand.

On the organization design the new culture can be supported by multi-skilled people who can work together effectively, diverse teams, low and dynamic organization structures, open communication and freedom for collaboration.

To be effective, experimentation should not be by just ad-hocking of implementation. In the Lean Startup methods it is emphasized that there need to be hypotheses and questions that are asked in the experiments. The experiments need to be planned and executed properly. This includes tasks such as:

- Deciding the goals of the experiment – why, what? Forming hypotheses and questions.
- Designing the artefact to be tested, either by some systematic method or by creative techniques, including team's brainstorming.
- Implementation of the artefact is suitable form.
- Planning the testing arrangements – who, where, when?
- Choosing a reference to use as comparison (old version, competitor's product).
- Metrics design.
- Data collection plan, including feedback collection, measuring the use and instrumentation of the product and handling data ownership.
- Security and privacy plans for the experiment.
- Execution of the experiment, including facilitation, observation, recording and so on.

- Analysis of small and big data.
- Synthesis of the findings.

Even in the agile environments, those are just tasks in a very traditional planned experimentation, of which there are lots of guiding materials available, for example Wohlin et al. 2000. Engineering students should have a grasp of it from their education. Software testers may have lacking competences for proper experiment design. In usability and user experience testing, similar issues are often tackled and indeed, the product experiments are often about user experience.

New areas are the concern of permissions, security and privacy and data ownership, if used with live systems, and the emphasis on data analysis, not just observations. Many companies do not have the necessary skills and thus it is expected that the results of experiments are not as valid as hoped. Another concern is that companies would replace good professional design with experimentation, which would result in very sub-optimal products.

Thus, there is a need for better experimentation competences:

- Meta-competence of balancing design and experiments.
- Experiment planning and design.
- Prototyping skills.
- User experience -related competences.
- A/B testing requires high competences in configuration management and product deployment.
- Competences related to permissions, security and privacy.
- Data analysis skills.

At the organisational level there is still work to do in making experimentation more integrated into the daily work. For example, hackathons (in companies or for a group of companies) seem often to be arranged during weekends, which clearly is not a sustainable practice.

Change-competence snippet 24	Experimentation culture
Change caused by -> enables	Need for innovation -> validated ideas, concepts
Competence implications (re: quality and testing)	Evaluation of product concepts #A Critical thinking and presenting critique #A Prototyping skills #A Hardware-related skills #U #A Experiment design skills #A Doing proof of concept tests for technology #A Using exploratory testing for understanding the behaviour of technology #A UX and usability testing #A Understanding permission, security, privacy #O #U Data analysis #U #A Creativity #A Cultural adaptation #A Changing company-level competence profile #O #U #A
Links with	<- Innovation in product development <- Fast product development -> Need for personal understanding of quality -> Flexibility over maturity -> Changing engineering education

5.6.4 Agility and flexibility

The business domains where products are used will always change, due to emerging markets producing new opportunities and old markets dying. A single company will need to pivot its offerings every now and then. Effective pivoting can be difficult, if the company is tightly specialised in one domain, the domain’s culture and technology. The usage of generic technologies across domains helps here. That is getting easier today than during previous decades, as there are less specific technology stacks for example in devices than previously. Operating systems and development technologies have been converging and will continue to do so. This allows for horizontal technology thinking: using technologies that can be used in many domains, using basic components and device platforms that are “domain-agnostic” and easily ported to new uses across domains.

We need to support technical skills targeted for applications that are used across domains, not only on the needs of some domain. That enables rapid change.

Testers also need deep domain expertise, but also general understanding and meta-skills that aid in the transformations. For example, understanding how reliability and safety of systems are analysed is something that applies generic principles on all critical applications.

Change-competence snippet 25	Agility and flexibility
Change caused by -> enables	Dynamic environment -> business change to new domains (existing and emerging)
Competence implications (re: quality and testing)	Understanding of domains and cultures #U Domain-agnostic competences #A Cultural skills (national, occupation, domains) #O #U #A Adaptability and flexibility #A Understanding of possibilities and alternatives in testing #U Broad flexible competence #O #U #A Multi-skilledness #O #U #A
Links with	<- Relation to change <- Changing Finland <- Flexibility over maturity <- Need for new types of workers <- Testers in development teams -> Agile software development -> Lean -> Need for personal understanding of quality

5.6.5 Faster decision making

There are many change factors that affect decision making in companies. Those include:

- The general idea of emphasising speed, which should result in faster decision-making than previously.
- The increased relying on experiments, which is a positive thing as it provides empirical data for the decision making, and validation of product-related hypotheses.
- Smaller companies and units, which decreases the number of people participating in decisions.
- Agile culture, which may reduce proper reviewing of plans, but on the other hand ideally allows multifunctional teams to join in on the decision-making.
- Start-up companies that have a very different setting in their businesses than the established companies.

All of those have been discussed previously, yet there is a need to separately discuss the decision-making aspect and how it relates to testing.

Though the years, the understood goals of testing have changed from finding out defect to providing information about quality (nothing, though, that the general stereotypes see that still in a narrow frame). That information is used in making decisions – is a feature solid, can the product be released, does it meet the requirements and so on. Because of that, the information must be timely, factually correct, in a format that is usable and presented in a way that really delivers the message to the people utilising the information. Some of the information exchange happens informally in discussions, some in formal reporting and some in information systems. To succeed, testers need understanding about the overall process and the people – what information does this manager need right now and next week? – and communication and information presentation skills.

Traditionally, the test communications have been seen as neutral – just deliver the facts though experts have also emphasised that important messages need to be "sold", taking into consideration psychological phenomena. The discussions about those have been concentrated on delivering defect information to programmers and project management (see for example Kaner et al. (1999)). Today, and in the future there is a need to open the focus of delivered information to the quality of product concepts, and technological and functional features. In that domain, there are new challenges, operational and psychological. Some of them are so essential that a summary here is in order.

Operational challenges include the reduced time for decision-making. In a short time period there is little time to plan a test for gaining information. Testing should optimally be proactive and provide data that could be needed in a foreseeable decision. But creating information that might not be needed is against common agile and lean philosophies, so a balance needs to be found here with business sensitivity.

The idea of testing or at least rapid experiments needs to be sold to the management and that is a competence area in itself.

When there is little time, the tests or experiments need to be carried out rapidly. That takes good experiment design and execution skills, as the speed must not cause a low validity of the experiments.

When there are time pressures, the organisation is prone to various cognitive biases. In fact, every decision is biased, claims Arnott (2006). Lowe & Ziedonis (2006) note that:

> *"(...) start-up firms typically have not developed detailed policies governing decision making, causing entrepreneurs to be more likely to rely on simplifying biases and heuristics. Entrepreneurs developing technologies in emerging*

> *technological trajectories also must often act quickly with limited information on technical feasibility and market conditions to convince financiers, employees, and other stakeholders of the start-up opportunity's prospects. This further encourages entrepreneurs to rely on simplifying heuristics to speed decision making."*

Testing cannot remove biases, but needs to aim to reduce their effect. The biases are essential to understand and mitigate in the future, because speed amplifies them. At lower pace there would be more time to ponder things, get more information and recognise the biases.

Some essential biases in product development include, picked from a list provided by Arnott:

- Confirmation bias. Often decision-makers seek confirmatory evidence and do not search for disconfirming information.
- Desire bias. The probability of desired outcomes may be inaccurately assessed as being greater.
- Overconfidence The ability to solve difficult or novel problems is often overestimated.
- Test bias. Some aspects and outcomes of choice cannot be tested, leading to unrealistic confidence in judgement.

Every new product development setting has lots of confirmation and desire bias. When there are novel ideas in the company, the managers and developers would obviously love to see them as successes and are naturally optimistic and confident (see Lowe & Ziedonis, 2006). Carelessly made tests may "prove" all the expectation, even if the reality was something else. This is where good testing and experiments are needed, carried out by proficient testers. Suitable tester independence from development can help the tester to handle the bias. Confirmation bias is very much related to author bias, where the designer or implementer is blind to the problems in her creation. It is mentioned in ISTQB syllabi as one motivation for the tester's independence. In safety-critical development, independency is required in product validation for various reasons. Independence can also help with availability bias (Mohan & Jain, 2008), where knowledge is gathered from where it is found most easily. In product development, that may be the previous projects. But in changing conditions that can be dangerous, as the old knowledge may not be up to date or otherwise sufficient any more. An external tester can bring valuable knowledge with her. Of course, the independency has its drawbacks too, if it means less directly applicable contextual understanding. So, as always, situations vary and the organising should be made carefully if there are options.

Note that there are other, closely related biases and authors use different terms in describing them and that this is not meant to be an exhaustive discussion of the phenomena.

Anchoring bias (see for example Mohan & Jain, 2008) is a common bias also related to the same phenomena. If for example, someone "decides" that the product is as good as a competitor, that forms a mental anchor for all participants and thus turns expectations and interpretations to support that idea. One needs to be careful with anchors and they should not let them guide test design.

For any situation in which the assessment of the product by itself is in danger to be interpreted through biases, comparison testing is valuable, as it can in the same context have a clear yardstick for any results – and quality is always relative. That obviously raises cost and effort.

Sometimes it can be thought that a feature cannot be tested before release. That might be true, but if good testers are challenged to find a way to test the feature, they might be able to do that. Sometimes "cannot be tested" may mean that it cannot be tested with test automation, but in that case it could be tested manually. In safety-critical development, when something cannot be tested outside production environment, it is subjected to careful, methodological analysis. That is something that should be considered also on non-critical development. (Note also that if the non-testable thing is technical, product architecture should be revised to improve testability.)

In summary, the competences that this discussion raises up are professional testing skills and especially the ability to do valid experiments rapidly. That is supported by understanding about the decision-making processes and the psychology involved. Using that understanding, the effect of cognitive biases can hopefully be minimised.

Change-competence snippet 26	Faster decision making
Change caused by -> enables	Dynamic environment, fast business -> rapid reaction, fast action
Competence implications (re: quality and testing)	Business understanding #O #U Communication skills #U #A Experiment design skills #A Prototyping skills #A Business and product concept level testing #A Quality advocacy #A Risk thinking #U Critical thinking and presenting critique #A Comparison testing #A Right timing of actions #A Dependability #O #A

Links with	<- Relation to change <- Flexibility over maturity <- Agile software development <- Lean -> Need for new types of workers -> Need for personal understanding of quality

5.6.6 Maturity models losing relevance

A traditional view to understanding organisational capability has been the concept of “maturity”. This is based on the idea that the better an organisation is the larger number of important processes it can execute professionally. This thinking has been put into use with maturity models or frameworks, most notably Capability Maturity Model, CMM, and its successor Capability Maturity Model Integration*,* CMMI, (CMMI Product Team 2010) first in the field of software engineering, and later in software acquisitions and services. In the field of testing, similar models include Test Process Improvement, TPI, (Koomen & Pol, 1999) and Test Maturity Model integration, TMMi (van Veenendaal, 2012), which closely resembles CMMI in its structure. The maturity levels of TMMi are presented in Figure 44.

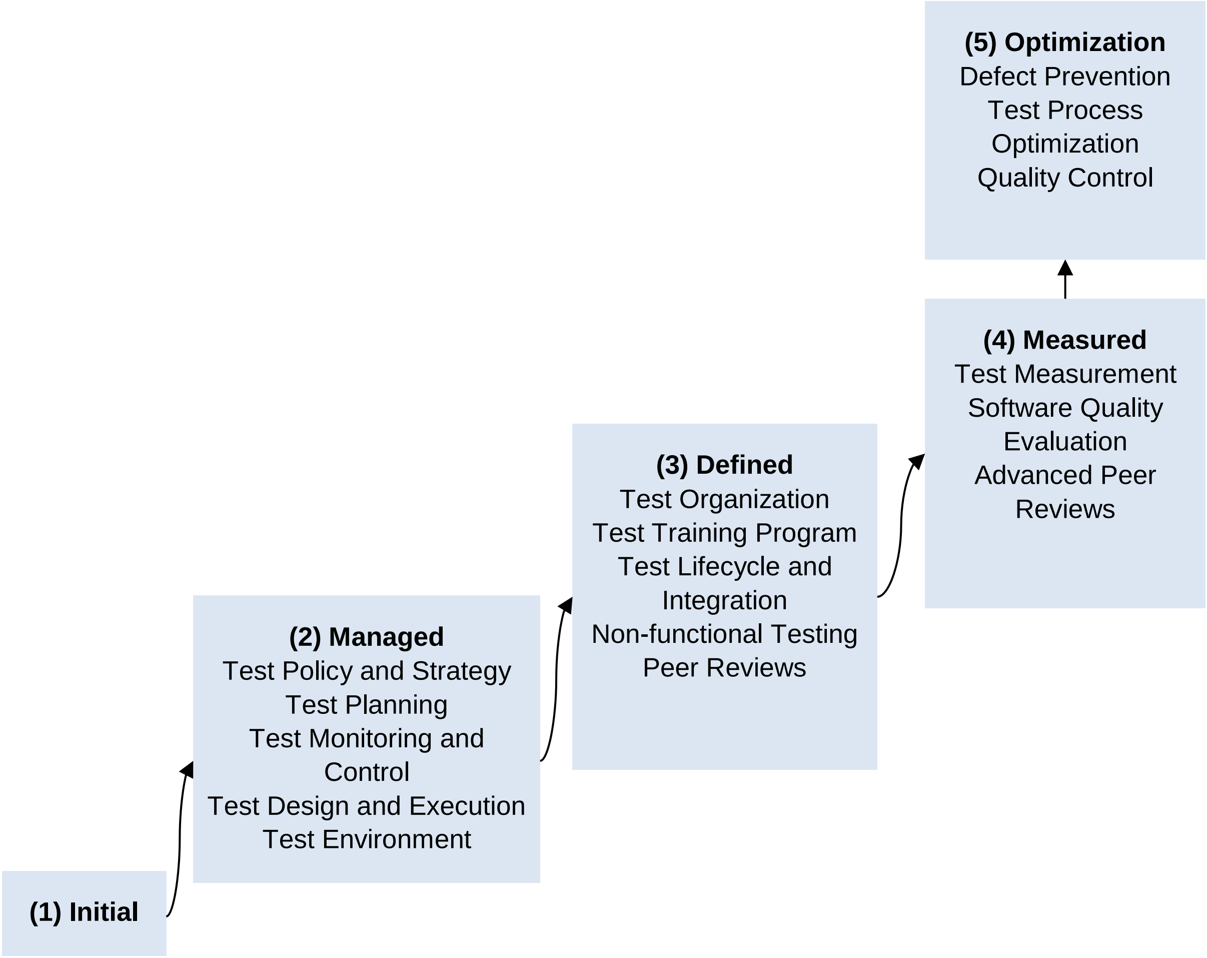


Figure 44. The maturity levels in TMMi (Veenendaal, 2012).

The author has previously in his training materials presented this critique to the maturity models. The maturity models are based on general truths, but the world of software development and testing is diverse, and best practices will depend on the company's situation and the basic ways of activity. The maturity models do not necessarily get into each organization's characteristics and success factors. The aim of the organizations does not necessarily have to be "mature", but to be the best kind for its business! Instead of maturity "optimization", more important is often the adaptability to new situations. The optimization gives wrong signals about the temperament of the development. The maturity models' goal should be to show in a clear way the organization's maturity level, often exaggerating it characteristics in a caricature way. The maturity models tend instead to excessive perfection and then they no longer are able to show the “big picture”. Instead of clarifying the situation – most importantly to the management – they raise up issues that are minor. And when a maturity model is so complex and abstract that it is no longer understood, it can no longer be used for guiding the organization’s future.

If fact, the agile culture has largely neglected the maturity models, as they are seen not to be relevant. Still, many consider that they may provide valuable input when used wisely. However, in the context of this dissertation, the main lessons are:

- Pure “maturity” is not a value as such.
- The ability to adapt and change can be more important than maturity.
- If order to build an organisation’s self-understanding, the development frameworks need to be simple and concentrate on issues relevant in the organisation’s context. Being generic is not sufficient.

Change-competence snippet 27	Flexibility over maturity
Change caused by -> enables	Dynamic environment
Competence implications (re: quality and testing)	Understanding about competence #U Adaptability and flexibility #U #A Broad flexible competence #O #U #A Multi-skilledness #O #U #A Understanding overall contexts #U
Links with	<- Agility and flexibility <- Emphasis on real competence

5.6.7 Changes and implied competences for this section visualised

A visualised summary of the change-competence snippets in this section are visualised in Figure 45.

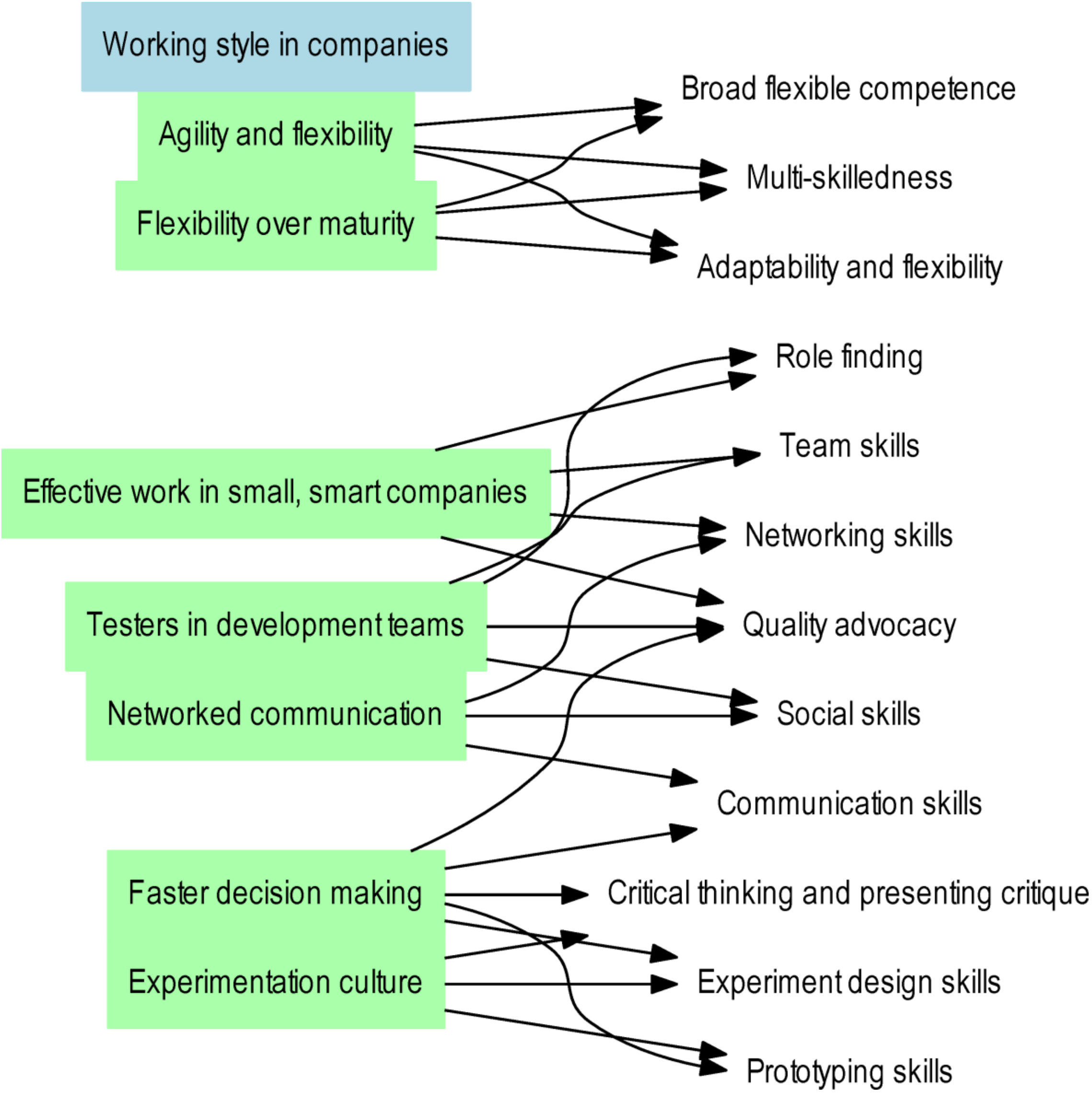


Figure 45. Visualised summary of the change-competence snippets in this section. To make the graph fit the page with legible font size, only the competences are included that are associated by at least two changes in this section.

5.7 Relations to competence in companies

5.7.1 Quest for multi-skilledness

Multi-skilledness is often emphasised for various reasons. First one is simply getting everyone something to do. When work is done in teams, there is not a constant need for other expertises other than software design, programming and low-level testing. Specialist architects, user interface designers and system level testers have work only at times during the process. This is a change from previous practices where testers might form a team and serve several projects. A startup might not even have more than one team and one project. Of course, the situation also reflects the situation where the other competences could be used, but are not. For example, if testing is not done that much, where would a tester be used?

Other reasons are that when multi-skilled people with differing backgrounds are doing tasks together, a team gets a larger knowledge base for the task, which increases the team's abilities for making sense of the situation, more innovation, better designing and better identification of risks and problems – and shared learning. Multi-skilledness is a key means for building diverse teams that produce excellent results! Both are very essential reasons and for example in startups are very critical needs for the future success of the company.

A very often mentioned model for a person's preferable competence profile is a T-shaped one. Madhavan, The vertical bar of the T refers to the high primary competence of the person and the horizontal stroke to the less competent familiarity of other disciplines, either in the scope of a team or things in general. Grover (1988) analyse the effect of T-shapedness for communication and knowledge creation and find it very valuable. Morten & Oetinger (2001) discuss the profile for managers. The horizontal line refers in their description to the competence required to spread across company hierarchy borders and thus the ability to share information. Others have continued on the same line, emphasising communication, such as Barilo et al. (2012). Tippins & Sohi (2003) emphasised that there is a need to export knowledge between parties for organisational learning – a goal deeper than just communicating and T-shape promotes that.

Oskam (2009) noted that the T-shape is beneficial for interdisciplinary innovation, which is a critical issue for Finland. Barile, Saviano & Simone (2014) analyse deeply the issues related to T-shaped innovators and see that such innovators are characterised by wishful thinking, lateral thinking, open-minded gifts, knowledge-seeking capabilities, and social intelligence, combined with analytic thinking that produces vertical learning.

Spohrer et al. (2010) specified the vertical line to mean "deep problem solving" ability. They describe the benefits of the T-shape to include the abilities to understand the vocabulary of disciplines, to describe problems that they may not be able to solve, and to reason about problems with experts who can solve them. In this regard, testers should be T-shaped in all areas of testing. For example, they may not be experts of security issues and security testing, but should be able to identify areas where it needs attention and then raise the issues with experts.

Note that the serifs of the T are important – they are not visible in the sans-serif fonts used here – as they point out to some smaller competence areas that contain actual skills, yet not expertise.

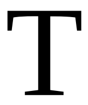


Figure 46. The serifs of T hanging from the horizontal bar have a meaning.

Buxton (2009) notes that to complement the T-shaped people, there should be I-shaped people too. They are the people who can solve the hardest problems and are needed in both development and testing roles. Those can for example be the hard-core engineers who are absolute experts in testing of critical real-time systems and other advanced things that really require complete focus.

When we combine two I’s we get H-shaped (or Pi-shaped (π) and sometimes A-shaped[21]) professionals, who have deep knowledge and skills on two disciplines. That would be very valuable as it would allow two personal identities, full ability to take various roles and role identities. And still, the horizontal line in H and Pi implies that the two disciplines would have a relation between them that is also familiar to the person. The only limitations to the vertical disciplines come from the individual’s interests.

There is also the Dash-shaped (-) competence, which refers to generalists. Those also have a place in product and systems development, should have their expertise outside the team’s operative actions. They could be coaches, mentors and so on and are also used for connecting people with deeper profiles. Indeed, they can often be found in communication roles (Chydenius & Gaisch, 2015).

We could think of one more competence letter! The “T”s, “I”s and “H”s are very static, standing still in one position. How about Ö-shaped competence? The shape means encompassing a knowledge area, but ready to roll ahead with the dynamics of the contexts, to move it according to what is needed. The shape also suggests a possibility to expand. And the accents represent spikes outside the primary domain; bringing in seeds of new ideas. This shape might not be ready for general use, but its idea of movements, dynamics on competences is critical. Besides the continuous dynamics of any context, this is related to the idea of life-long learning. It is quite possible to learn a new, strong I every ten years.

Most people should have several skills spanning in some way more than one disciplines at least in a T-shape. At the team level, the "T-shaped" competence profile

[21] A-shaped is rarest, but used for example by Leonard-Barton (1995).

is used to mean that there is one core competence that forms the basis for a role in a team, but that is supplemented by a couple of secondary competences. For example, a tester might be specialised in testing, but have secondary competences in implementing software, user interface design and similar. The opposite arrangement (which some Agile subcultures seem to think is a good idea), is a coder-centred team, where the focus is on programming and the only core competence is programming and the programmers may have sufficient other competences. There is evidence that we really need diverse teams and great competences in many areas. The days are gone when it was accepted that any programmer can create a great user interface or that only low-level functional testing produces sufficient testing.

The practical discussion is usually focused on being able to "do" things. Even more important can be the ability to innovate and to communicate outside the team.

As practical examples in we can identify the following:

- Functional testers can expand their competences towards assessing information security, usability and user experience, and reliability engineering.
- Testers can become programmers, but their mindset is not always suitable for innovative designing.
- Testers could do requirement specifications and user research.
- Testers could manage continuous integration and deployment systems and take care of product packaging and deployment.
- Software developers can expand their low level testing skills to all areas of testing.
- Everyone in the teams can expand their business context knowledge.
- Everyone can become a “company-level” expert in some critical technology.
- Everyone can become a facilitator of team’s work, customer sessions, safety analysis sessions, reviews and so on, and also a Scrum master and similar.

The competences related to this are two-fold: Testers should carefully think of what secondary competences they should develop that would help in their career. Other professionals should think whether testing-related competences would be suitable secondary competences.

Companies should coach their personnel and help employees develop their secondary competences and support them using those in a safe way.

The competence portfolio synthesis in composing a team should take the secondary competences into consideration. This is essential for self-forming teams where the key people need to understand the broadness of competence needs. The team leaders should have a broader than I-shaped competence profile.

Change-competence snippet 28	Quest for multi-skilledness
Change caused by -> enables	Effectiveness, smaller companies -> collaboration, dynamic organisation
Competence implications (re: quality and testing)	Broad flexible competence #O #U #A Multi-skilledness #O #U #A Personal competence development #O #U #A
Links with	<- The startup phenomenon <- Need for new types of workers <- Business understanding for all <- New technology products <- Testing of intelligent systems -> Changing engineering education

5.7.2 Business competences for everyone?

Understanding business is often mentioned as one important new competence area. It does not mean that developers and testers should be able to make a business plan or to manage a business. They just should understand the company's business to be able to support it and its priorities and to effectively get information from the business people that is needed for development and validation of designs and implementations.

Business competence for testers would include things such as the following:

a) Business in general:

- Understanding that this is not a game, but a matter of succeeding in big things.
- How things are thought about in business?
- What does "quality" mean in the business context?
- What are business risks like?
- How are the businessmen different from product developers and testers?
- What is the world of product manager / PO like? What does she do, think, what pressures does she work under?[22]
- The whole of the activity, systems, ecosystems.

[22] About the viewpoints of other parties (in Finnish): ”Testaus organisaatiossa – eri osapuolten näkökulmia laadunvarmistukseen and testaamiseen” (Vuori, 2010e).

b) Culture of the domain:

- The generic nature of the domain – is it strict engineering or something more free-form?
- What is the communication like – reports or chatting?
- Are the actors assurers or risk takers? What kind of things characterise the domain?
- What are the customers like, what do they need?
- In what actions must we be “sharp” and in what we can be more relaxed?
- Knowing the essential concepts and terms of the domain. What things are so obvious that the customer does not even mention them?
- The enablers and limitations of the domain. Legislation and important standards.
- Ecosystems.
- In the context of national cultures, cultural skills and training are often talked about. This is a similar area.

c) The customer of testing – the business actor:

- The central goals of business, the mission.
- What goals do the organisations, units and individuals have?
- Why does the customer need systems? What is their "beef"? How do they use the systems? To what kind of operations, how do information and money flow?
- What is important for the customers in their everyday? Is speed, quality, comfort or safety (for example) above others?
- What is the use environment of the systems like, other systems?
- Understanding about risks related to schedules and taking systems into use?
- What is essential in starting a new business (with an information system)? What things will happen, what all must succeed, what definitely must not fail?

d) The systems that are developed and tested:

- What is their role in the business processes?
- Which functions and characteristics are really important when they are used for business purposes? Where can compromises be made?
- What kinds of risks are there if systems do not work? What things could cause interruption of business? What are the information security risks (information, functions)?
- What kinds of decisions are made about the system? What information is needed for that? When is that information needed?
- What could be the problems, when the number of customers increases or the volume of business grows?
- What kind of requirements are there for the system. Does the law say something? Are there product or process standards?

- Who knows more about things? What is the oracle of things or is there any?
- What kinds of systems are usually used in the domain? How is their usual – that is, expected – level (of quality)?

Change-competence snippet 29	Business understanding for all
Change caused by -> enables	Testing needs to support business -> better testing, better information -> better business
Competence implications (re: quality and testing)	Business understanding #O #U Understanding of domains and cultures #U Business understanding #U Understanding customers #U Understanding users #U
Links with	<- Quest for multi-skilledness <- Smaller companies <- Changing Finland <- Testing in every process <- Innovation in product development -> New thinking on defect costs during application lifecycle

5.7.3 Scaling and adaptation of competences

As already briefly noted, competences and capabilities need to scale. To assess this, we need to think of some dimensions where the scaling can be required. It should be noted that the needs for scaling depend very much on the context.

The main dimensions of scaling are project size / testing volume, system criticality, technical diversity and distribution of work. The project size is an obvious thing that varies. There are projects and systems of every size and adapting to those requires competences both at personal and organisational level. People will, for example, need to be able to handle large amounts of things to test in large projects, but also change to a different mind-set for small in-house projects, such as developing and testing of a small utility. Large systems need the skills of handling large amounts of issues without losing track of the whole. The issue becomes even more challenging, because large systems tend to be more complex. There are not just more things, but there is more technical diversity. There are many different kinds of things, with many types of advanced – or just plain strange – interactions. So, the ability of handling complexity is required as well. The traditional way for that is to use tools, such as test management tools and application lifecycle management tools. Fluent using of those is a competence area in itself. Also, large systems need plenty of effort to test, so test automation skills get more and more important, the bigger a system is. These challenges are not just at personal levels. At team, unit or subcontractor level, there must be similar ability. Small and large systems need to be handled. Large systems often start as small, so it is good to have processes, skills and tools that can scale at the same rate as the system grows.

Criticality of the system is something that varies from project to project. It depends on the context where a system is used. When the context changes, the criticality of the system changes. For example, a simple in-house system might be taken into global use in a business critical process. Individuals and organisations as whole need to be able to detect the changes in contexts. More critical systems require more advanced development and testing methods because of the risks and because some mandatory standards may demand it. The ability to detect important changes in criticality and how development and testing should respond to it is an important “meta competence”, because the change in criticality may cause a need to utilise new competences – by the same people or by some other people who need to get involved in the project.

Distribution of work is also something that associates to the size of the system, as large systems can seldom be developed in one location, but the development needs to be distributed globally, to different countries and different cultures. Handling that is not easy.

The main thing to note is that all of these issues magnify constantly:

- Systems get bigger and bigger.
- Systems get more complex.
- Systems get more diverse.
- Systems get more critical for business or safety.
- Systems are developed in a more distributed way.

The ability to handle all this can be a real challenge. The essential competences for testing related to scaling are:

- A scaling “toolbox” that can adapt to changes in volume.
- Compatible tools and methods that scale with criticality growth.
- Information systems that are easy to start with, but can handle large amount of system elements and testing information.
- Mind-sets that are not “locked” in some typical situation, but which can help us see what the current situation is like.
- Styles of using resources, that scales.

Change-competence snippet 30	Scaling of competences
Change caused by -> enables	Changing businesses, domains, growing companies -> successful lifespan for companies
Competence implications (re: quality and testing)	Scaling personal toolbox #A Platform-agnostic skills #A Understanding domains, contexts and situations #U Scaling resource management practices #A Risk thinking #U Process development #O #U #A Business understanding #O #U
Links with	<- The startup phenomenon <- From products to services <- Machine industry turning into software industry <- The changing requirements of technical software systems

5.7.4 The business processes where testing is used

The business processes where testing is carried out really influence how we see it. For example, in software development, testing is often carried out as “neutral” activity, but in some cases it is presented as a means for example validating and verifying product features and designs. This happens often in a regulated environment, where systems need to be formally accepted before being marketed. On the other hand, when companies use testing to ensure that an information system fulfils their needs, they almost never talk about verifying and validating anything. Testing is just a neutral tool that produces the necessary information for making decisions.

Testing as a conscious activity is seeing a growth of application areas and because of that, also a growth in disciplines and vocabulary. This new richness is at the same time being controlled by standardisation and certification schemes. Still, when testing is being applied in a process, one very special competence is the ability to position it in the context correctly and to communicate the purpose of testing correctly.

Change-competence snippet 31	Testing in every process
Change caused by -> enables	Moving from engineering to product development -> better business, lower risk level
Competence implications (re: quality and testing)	Business understanding #O #U Business and product concept level testing #A System and system of systems thinking and testing #U #A Process development #U #A (for integrating testing and experimentation into business activities)
Links with	-> Business understanding for all

5.7.5 Integration of quality management

During the process era of software development, it was a norm to have a framework for quality management. The whole society expected that every company would have a formal quality management system based on an international standard, most usually the ISO 9000 series and in that particularly ISO 9001 (2015) and additionally software subcontractors were expected to implement CMMI maturity model (CMMI Product Team, 2010).

Many companies saw ISO 9001 as a baseline for their quality management, just much as many implemented it only because their client demanded it. It is noteworthy that ISO also had a special standard 90003 (ISO/IEC 90003, 2014) that would give guidance for applying ISO 9001 in software development, but that was very little known in the industry or even by quality consultants who promoted ISO 9001. Situation was much the same with CMMI. It was usually considered quality bureaucracy and it was seen that levels above 3 were hard to reach and would not reflect a rise in collaboration capability.

When companies started a transition to agile development, those frameworks were usually neglected and focus turned to proper implementation of agile methods[23]. This is partly positive, as the focus is in the real, holistic activity of the teams instead of external evaluation models. At the same time, it is expected that the assessment of process quality is performed in sprint reviews and such, instead of quality audits made by an auditor team. The competences needed for the “new era” of quality management are related to personal and team level self-reflection. Abilities are needed for understanding the quality or action and any deficiencies (or strengths) in it, producing

[23] Naturally, the developers of CMMI see that CMMI has a role in Agile too (Glazer et.al 2008).

information for the management about the quality of action and a base for trusting the activity.

When it comes to the priorities in selecting quality related practices, the new focus on unique competences means that practices should not be selected or prioritised based on external frameworks, but by the actors' own understanding of what practices are really beneficial and will produce real value for any process.

Exceptions in this are the safety-critical domains where the mandatory practices may be defined in safety standards and auditing against those is a natural part of organisational life. In those environments it is often necessary to have explicit "quality assurance testing", whereas in other domains testing is just testing and produces not only "assurance", but other support for business and development teams in a balanced whole.

A return to the quality framework is not to be expected during the recent years. Yet, the ideas of ISO 9001 are very relevant and relate to the needs of companies. First, the standard emphasises quality culture (although it does not use that term) of solid management, processes that fit together and activities that help produce good quality and customer satisfaction. Collaboration in the overall scope of a company is much raised in the new concept of devops, described in novel form in Kim, Behr & Spafford (2014), that emphasised the collaboration between product planners, developers and hosting and maintenance personnel. The collaboration between parties and optimising the whole is a key factor in any quality management approach.[24]

Change-competence snippet 32	Integrated QA
Change caused by -> enables	Varying contexts, team-independence, ISO 9001 experiences -> independent thinking, selecting most suitable practices for context
Competence implications (re: quality and testing)	Understanding quality and its practices #U Understanding domains, contexts and situations #O #U Process development #U #A Quality advocacy #A Active, self-steered working for quality #A

[24] The full list of the principles of ISO 9001 is: customer focus, leadership, engagement of people, process approach, improvement, evidence-based decision making, and relationship management.

Links with	<- Flexibility over maturity <- Agility and flexibility <- Need for personal understanding of quality <- Rethinking the goals of testing and quality assurance <- Testers in development teams

5.7.6 Changes and implied competences for this section visualised

A visualised summary of the change-competence snippets in this section are visualised in Figure 47.

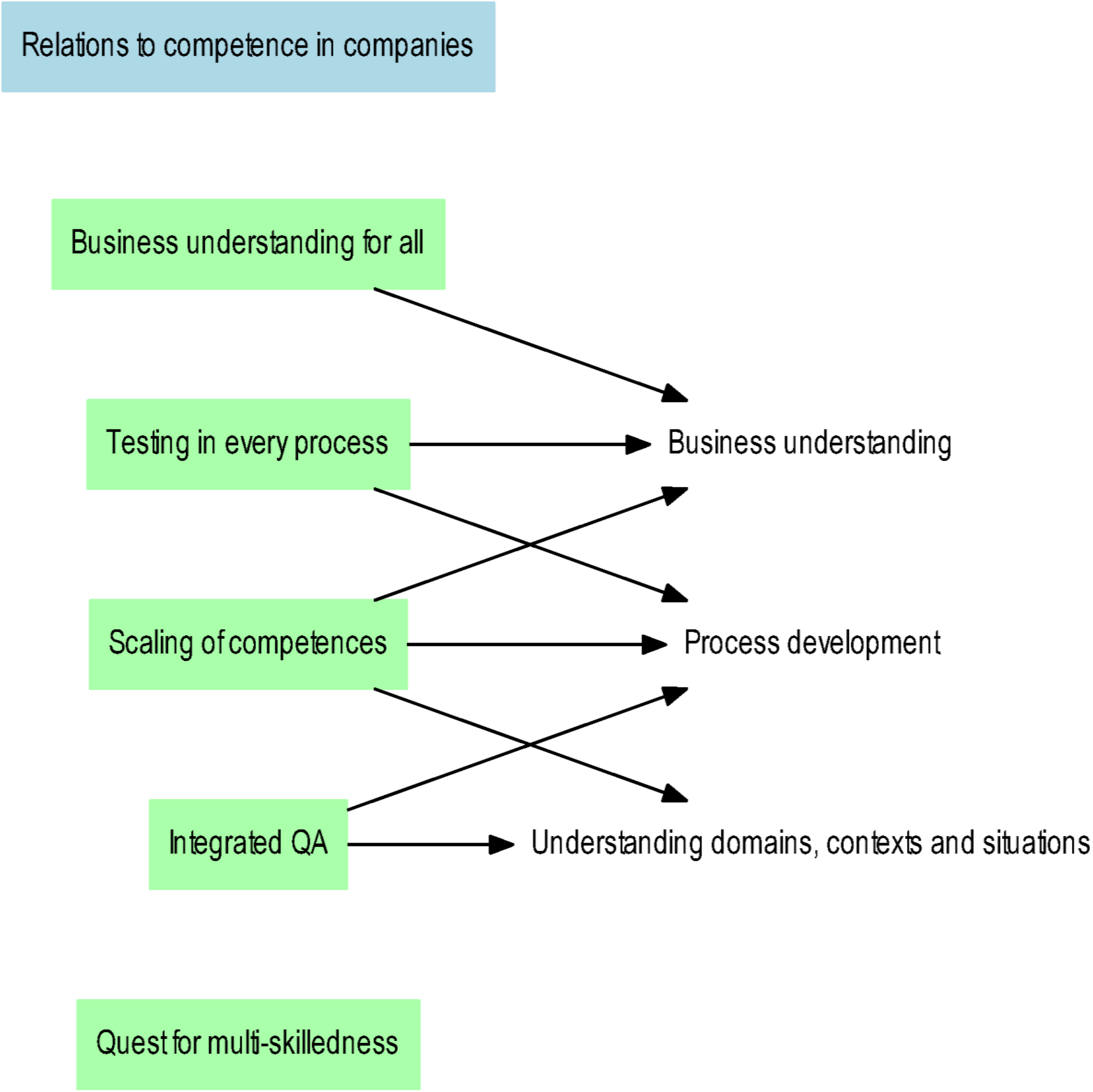

Relations to competence in companies
Business understanding for all
Testing in every process
Business understanding
Scaling of competences
Process development
Integrated QA
Understanding domains, contexts and situations
Quest for multi-skilledness

Figure 47. Visualised summary of the change-competence snippets in this section. To make the graph fit the page with legible font size, only the competences are included that are associated by at least two changes in this section.

5.8 Changes in product technology

5.8.1 Industrial Internet

As Finland is a country with plenty of industrial heritage, Industrial Internet is particularly interesting new phenomenon. It became a big “hype” around 2014 and is gradually becoming more and more understood. Mostly it is, like Internet of Things (IEEE, 2015), based on connectivity of things, but in an industrial setting, where the hopes and challenges are different. Basically, one wishes to be able to control and monitor all devices in an industrial system, to collect data from their use and so on. Obviously, the new technological infrastructure would enable the creation of new types of system concepts, more dynamic systems and businesses that utilise the new streams of data from the connected systems.

When it comes to the quality and testing challenges, there is a publication made by a consortium of big system vendors that presents a reference architecture for Industrial Internet (Industrial Internet Consortium, 2016). Its picture of the reference architecture demonstrates the challenges – there are many tiers, various components with functionalities that may be new to the designers and implementers.

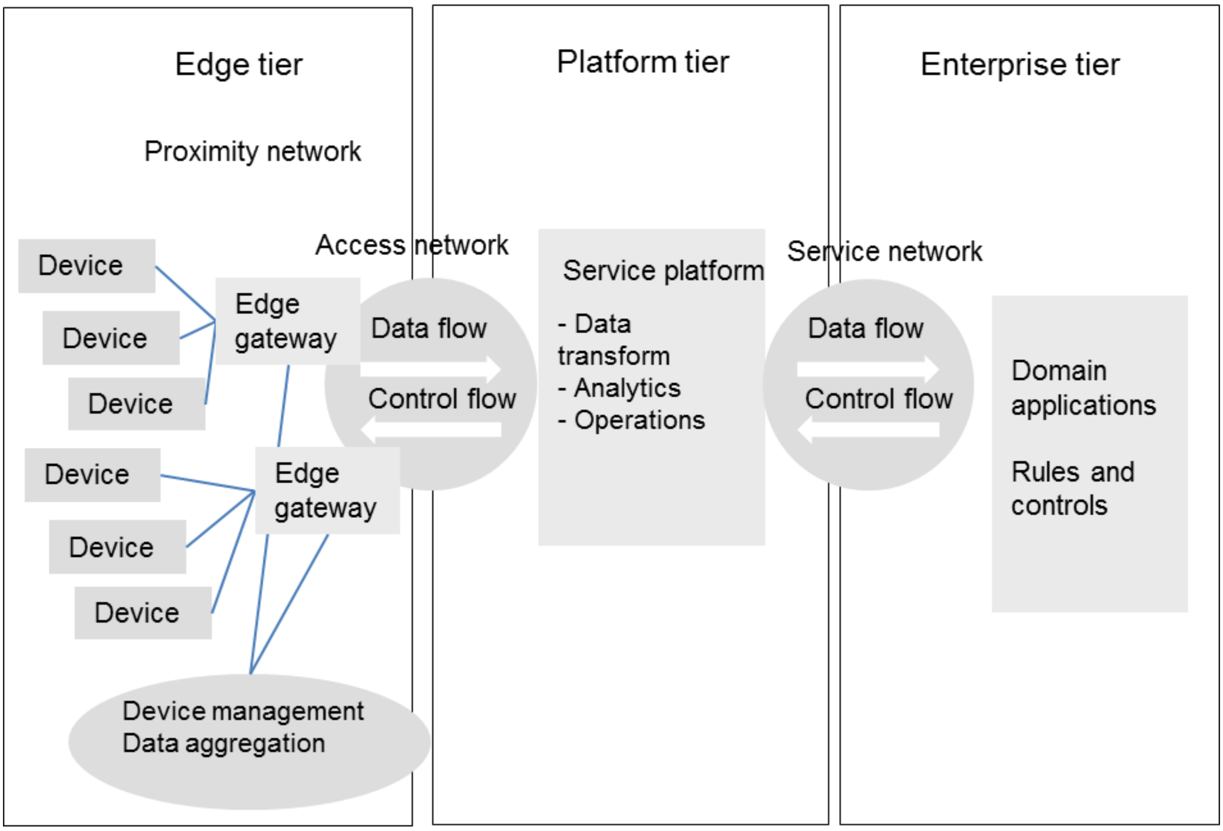


Figure 48. One view to the reference architecture for Industrial Internet system based on Industrial Internet Consortium (2016).

The document lists key system concerns, the areas that need special addressing in design and validation:

- Safety.
- Security, trust & privacy.
- Resilience.
- Integrability, interoperability and composability.
- Connectivity.
- Analytics.
- Intelligent and resilient control
- Dynamic composability and automatic integration.

There are challenges for the competences, as even traditional safety-critical industrial automation systems demand a quite large set of knowledge and skills. Here, especially issues related to integrability, interoperability, dynamic composability and security are very demanding from both design and testing perspectives. It is very clear that the verification and validation of this kind of systems require the combined competences of many different professionals.

Finland has great hopes for being a major global player in this field and there certainly is much potential for that. A government report (see Ailisto et al., 2015) offers these 15 recommendations for the public sector:

Leadership and Implementation

1. Create a Finnish story and wake-up
2. Lead change as part of the government platform
3. Appoint a person in charge to co-ordinate the implementation of the industrial internet

Market access

4. Make use of innovative public procurement: 5% obligation
5. Support the partnerships of different types and sizes of companies
6. Carry out focused market intervention

Markets and business models

7. Clarify the rules of game for data ownership and management
8. Reduce regulation and reform taxation
9. Specify common platforms and standards at the EU level

Competence

10. Start adult education at different organisation levels
11. Reform the education programmes at universities
12. Support self-initiated education
13. Derive best practices from business models

Technology and platforms

14. Ensure a cyber-safe industrial internet
15. Produce a rapid-testing environment

It is good news that this issue is taken seriously and will hopefully lead to action. Certainly, the education and training sector has taken action. A Google search 15.1.205 found several training arrangements in Finland, such as a training programme in a university's professional development unit, several short courses provided by training houses and IoT recruiting programmes. Inclusion in regular curricula of education institutes is expected to increase rapidly.

Change-competence snippet 33	Industrial Internet
Change caused by -> enables	Internet technologies, low cost of communication technology -> added value by intelligence, monitoring, maintenance
Competence implications (re: quality and testing)	System and system of systems thinking and testing #U #A UX and usability testing #A Risk, safety and reliability analysis #A IoT-related competences #O #U #A Hardware-related skills #U #A Understanding information security risks #O #U Information systems and integration competences #O #U #A Security assessment and testing #A Data analysis #U #A Architecture evaluation #A Doing critical technology assessments #A
Links with	<- Big Data -> Platform economy and API economy -> Multi-device systems with new interaction styles -> Testing of intelligent systems -> New technology products -> Information security and privacy

5.8.2 Big Data

Big Data is a general paradigm for understanding that there could be a lot more data than is traditionally used in business or in monitoring businesses. The ideas are now relevant due to two main reasons: 1) commerce in Internet is capable of producing much more data about the behaviour of customers than before and 2) the connectivity of devices is enabling them to send huge amounts of data about their operation, enabling monitoring, diagnostics and prognosis about their future performance and informing of any needs for maintenance or notification about changes in the devices.

In Finland, the phenomenon has received plenty of interest and studies have made of its impact. For example, the Ministry of Transport and Communications published a report about Big Data in Finland (Alanko & Salo, 2013) with the goal of "to explore the current Big Data phenomenon with regard to how it is perceived and how it is expected to develop", which notes that

> "The report clearly points out two issues. On the one hand, for many organisations Big Data is a new concept that is difficult to grasp. On the other hand, it was generally recognised as a strategic force of change and a prerequisite for future

> competitiveness. In the near future, investments in research and training will increase and new projects will be launched.
>
> Almost without exception Big Data as a phenomenon was considered interesting and significant. An increasing shortage of experts, enormous diversity of software services related to the phenomenon, and explosive growth and diversification of source data were considered the most significant challenges in bringing Big Data to the core of everyday data processing."

Big Data is important from the viewpoint of testing, because obviously the collection and analysis of data is yet another feature that needs to be validated as functionally correct, effective and secure. This means for example, extra effort on testing the endpoints and analysis systems in Industrial Internet systems.

The data can be used as a tool in testing too. Again, in Industrial Internet, we can consider the vast device data to be an advanced form of logging of every device, collected in harmonised data formats and by common protocols. That will very useful in the testing of the overall system. Traditionally the approach to system under test has been to take snapshots of system behaviour (a collection of test cases run a given time. Long term testing obviously generates a collection of data series. The new ideas of experiments and A/B testing are based on collecting behaviour data and accessing its variations when the system is used in varying designs and configuration. In an experimental and learning mindset, use of a system as such is always at the same time a test.

From a defect tracking perspective, it is important to combine defect data to the behavioural data, data about system’s states and history. The data will also allow us to identify potential system states that could be problematic before they really develop that way.

The implications for testing are a possibility to utilise data analytics, data science skills with the data and yet more complexity demands more advanced testing skills and rigour. For that to be possible, some of testers need those skills. The essential skills include:

- Instrumenting the systems.
- Efficient and secure data collection. Consider for example Industrial Internet applications.
- Data analysis in theory and practice.
- Using analysis and reporting tools, including Excel, focused tools and programming languages such as Python and the special language R (The R Foundation, 2015), which is currently gaining in popularity.

So the role of big data will be twofold: it is an element in the systems to test and a tool for testing. The testers doing this will not need to be “data scientists”, but “data practitioners”. The data handling is just yet another tool in their personal toolbox.

Change-competence snippet 34	Big Data
Change caused by -> enables	Connectivity, sensors -> monitoring, prediction, testing
Competence implications (re: quality and testing)	Instrumenting of systems #A Efficient and secure data collection #A Information systems and integration competences #O #U #A Data analysis #U #A Using analysis and reporting tools #A Understanding information security risks #O #U Security assessment and testing #A Architecture evaluation #A
Links with	-> Industrial Internet -> Platform economy and API economy -> Information security and privacy

5.8.3 The cloud as a platform for systems and testing

Cloud computing is mainly a new delivery paradigm – deliver software as services, not as products. In the cloud, there is a pool of resources that can be rapidly taken into use with minimal management effort or service provider interaction. The NIST definition framework (2011) presents the essential characteristics: on-demand service, broad network access, resource pooling, rapid elasticity and measured service. Service models include software as a service (SaS), platform as a service (PaS) and infrastructure as a service (IaS). The publicity level of a cloud can be private (an organisation’s own cloud), public cloud, community cloud, or a hybrid of those.

There are two main aspects to cloud for testing: testing of cloud application and systems, and using cloud for testing.

According to Riungu-Kalliosaari et al. (2012) there are many practical implications to testing. For actual testing the cloud provides more efficient performance testing, quicker testing and more realistic test results. For testing services delivery and support the cloud gives better availability of testing tools and options, improved developer–tester communication and enhanced service delivery for vendors. There are also challenges, such as that the cloud-based testing requires testing of additional aspects and parameters. Security-related issues are a major concern, especially in test data management. There are also concerns about domain knowledge and budgets.

The author made a short analysis of the applications of cloud in testing, which is shown in Table 22.

Table 22. Issues in cloud testing.

Issues and criteria	Notes
Benefits	
Scalability	Automatic scaling possible. Get more capability and installations as you need
Rapid start of testing	Turn-key platforms Turn-key whole infrastructure Application ready to use No own computers needed Business projects can control the environments, not the ICT department No need to wait for budget – pay for what you DO
Cost efficiency	Renting – use as needed No investments No own maintenance Centralised, costs shared with many parties
Mental concentration	Concentrate on testing, not environment management More mental room to be creative
Availability	No reservations, just use
Faster execution	(If tasks can be divided to multiple computers)
Ease of outsourcing	Equal accessibility
Isolation	Testing isolated from production systems
Agility in changing plans	No commitments to infrastructure
External service	Can demand more than from own IT Service level agreement essential
Potential problems	
Total control of environment	Fixing problems in environment Adding own applications, tools, drivers
Information security	Access control Data Auditability
Application integration	Integration of external services in main platform
Stability of versions, configurations	Configuration management
Legacy systems	Systems that are not “cloud compatible”
Lack of standards	Compatibility, de facto standards, choice of “ecosystems”

Issues and criteria	Notes
Performance	Latency Connectivity Throughput Uncontrolled variability
Quality of cloud vendors	Auditability Reality behind adverts Garage companies Continuity Handling of growth Keeping the service level agreement promises
Wrong choices	Hype! Getting clouded Wrong expectations Lack of planning
Behaviour rules for hard testing	Risk of breaking the cloud Terms of service
Uses	
Configured project platforms	Generic platforms Company-specific
Test management	Bug management Test management system, test scheduling, test running, test logging Configuration management Document management Requirements management
Test design and execution	Testing tools as a service Test design and execution as a service Automatic test design as a service Standard test suites
Test environments	Virtualized OS environment Real, hosted computers, remote testing, farms Expensive environments, even mainframe testing environments Just in time deployment – as tests are deployed, any operating system, any installed application set, as many as needed
Test types	Performance testing, test client farm made to choice Security testing, safe sandbox Usability testing (manual) User experience testing, A/B testing Other test types

Issues and criteria	Notes
Generic information management	File management Office tools Calendars
Testing as a service	Human cloud: tester resources, crowdsourcing Manual testing services Automated testing services
Types of cloud	
Public	Security
Private	Need for own management infrastructure – benefits?
Community	Management issues
Hybrid	Finding a good architecture
Application under test	
Information systems	
Workstation applications	Cloud as virtualisation platform
Mobile applications	Device farm usage
Cloud systems	Test in development Test in production
Utilization stack	
Test techniques	Generic test techniques apply
Practices and test processes	Workflows for test deployment
Platforms and tools	Tools compatibility with cloud environment Availability of tools from vendor
Goals for testing in cloud	Understanding of why the cloud is used in the first place
Vendor business opportunities	
Choosing a good service portfolio	Platforms, applications, tools Testing services Monitoring and measurement services Consulting and training Part of cloud testing infrastructure, "Testing mashups"

Let’s also mention one great opportunity in cloud testing. It provides possibilities to offer testing environments and infrastructure to companies as turn-key service. Those could be used in an ad-hoc manner and scaled as required. This all could be very important to startups and lower their competence requirements for test environment management.

Change-competence snippet 35	Cloud testing
Change caused by -> enables	Cloud -> dynamic test environments, low investment, new testing opportunities
Competence implications (re: quality and testing)	Understanding cloud systems, their possibilities and problems #U Managing of test environments in the cloud #A Deployment and automation skills #A Understanding information security risks #O #U Learning new testing tools #A
Links with	-> Virtualisation -> From products to services -> Fast product development -> Information security and privacy

5.8.4 Virtualisation

The cloud can be seen as one type of virtualisation, but virtual environments are used locally even more and more in the forms of virtual machines and packaged OS environments such as Docker. Even hardware is often virtualised at least in testing. For testing, virtual machines are a fantastic invention. Where traditionally physical computers needed to be used, hard drives slowly copied and applications and data installed for a test round, a virtual machine can now be deployed in seconds that is automatically configured for the application under test, its configuration, the OS configuration, test data and so on.

Every developer and tester should now have skills for understanding virtualisation and every organisation should have some people who can create the systems for creating virtual machines as needed for a test round, test session or even for a small test set.

Change-competence snippet 36	Virtualisation
Change caused by -> enables	Virtualisation technology, computer capabilities -> fast deployment of environment, hardware and OS-agnosticism
Competence implications (re: quality and testing)	Understanding virtualisation #U Virtual environment design and implementation #A Virtual environment deployment skills #A
Links with	-> Cloud testing -> Fast product development

5.8.5 Multi-device systems with new interaction styles

The future of our environments will be full of Internet and local network connected intelligent devices that work in collaboration with us humans and other devices. The devices use heterogeneous, both open and proprietary technologies and are constrained by security and safety concerns. The device environment is dynamic – new devices are brought into the environment and old ones removed from it in an ad-hoc manner. Finally, the devices use new means of interactions – spatial gestures, gaze etc. A role in this will also be by “liquid software” (Hartman et. al, 19996), where the execution of software currently run can be moved ad-hoc to another device.

That kind of an environment is prone to various problems that need to be found during development, including:

- Technical device compatibility.
- Which devices will respond to humans’ actions and communication?
- Coordination of action between devices.
- Moving control and data (such as a video feed or a phone call) between devices.
- Handling and stopping any undesired or unsafe activity.
- Correct detection of human gestures and other means of device control.

All in all, the range of necessary testing is broader than with current systems and the testing arrangements are more complex. This requires higher levels of technical competence in the creation and management of the test system. Understanding of technical systems and socio-technical systems is critical for creating good overall system simulations. For creating automated tests, simple scripting is not sufficient for orchestrating the whole, but some types of model-based testing are needed.

One challenge is the combination of disciplines. Creating a solid system requires testing using many viewpoints, such as safety, user experience, security, functionality, low level human-computer interaction and so on. Finally, a new mind-set is needed to handle the dynamic nature of the systems, as the system configuration can change at any time, and control of the system will change between devices.

Change-competence snippet 37	Multi-device systems with new interaction styles
Change caused by -> enables	Device interactions, IoT -> collaborative device systems
Competence implications (re: quality and testing)	System and system of systems thinking and testing #U UX and usability testing #A Testing of complex interactions #A IoT-related competences #O #U #A Experiment design skills #A Doing proof of concept tests for technology #A Security assessment and testing #A Risk, safety and reliability analysis #A Using exploratory testing for understanding the behaviour of technology #A Hardware-related skills #U #A Configuration testing #A Installation testing #A Architecture evaluation #A Configuration management #A
Links with	<- Industrial Internet -> Information security and privacy

5.8.6 Changes and implied competences for this section visualised

A visualised summary of the change-competence snippets in this section are visualised in Figure 49.

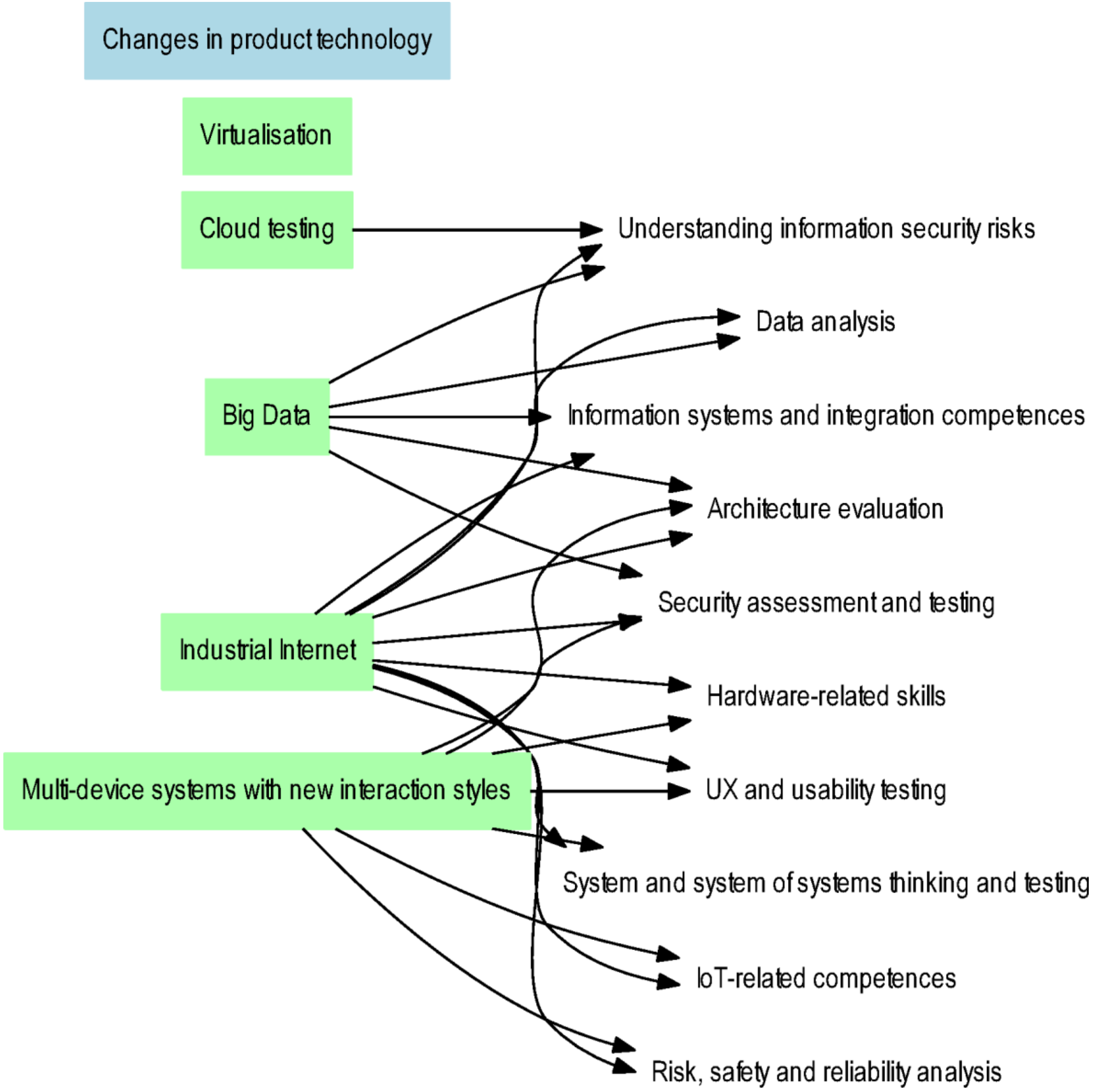

Changes in product technology
Virtualisation
Cloud testing
Big Data
Industrial Internet
Multi-device systems with new interaction styles
Understanding information security risks
Data analysis
Information systems and integration competences
Architecture evaluation
Security assessment and testing
Hardware-related skills
UX and usability testing
System and system of systems thinking and testing
IoT-related competences
Risk, safety and reliability analysis

Figure 49. Visualised summary of the change-competence snippets in this section. To make the graph fit the page with legible font size, only the competences are included that are associated by at least two changes in this section.

5.9 Changes in product requirements

5.9.1 Explosion of important quality attributes

The important quality factors that are actively tacked in projects have changed during decades and it is logical to think that this development will continue. There are many reasons for that. Software systems have a broader role in our world and that role is becoming even more pervasive in our environments and lives. Software systems have an effect on everyone, doing on most anything and they clearly have become more critical than ever. Through networks, systems are more visible, gain more users but also misusers, which increases the possibility of system failures. At the same time, systems have become more complex, more difficult to develop and maintain, and that increases the potential for failures in the systems we use. All this is a continuous learning process. We have collectively learned about the characteristics of software system and now see their characteristics in more diverse way. For example, systems are more seen as providers of value than just technical systems.

Thus, we are able to see clearly new quality factors that were not perceived or conceptualised before. They add to the volume of the elements of overall quality. That in turn creates new collective challenges – there are much more things to understand, to design for and to assure than previously.

Just some examples:

- Usability as a concept was more widely used only from around 1995. The classic book Usability engineering (Nielsen, 1993) was an eye-opener for the whole software development industry. Before that, the approach was only in handling traditional computer ergonomics – layout of the screens, font sizes etc.
- User experience came into discussion around 2005.
- Information security was very rarely mentioned during the 1990’s, but become understood around year 2000 to be essential in all systems.

It is clear that we will have even more surprises in this regard as time goes on.

Change-competence snippet 38	Explosion of important quality attributes
Change caused by -> enables	Quality attributes expand -> need to assess for total quality -> better products
Competence implications (re: quality and testing)	Understanding changing nature of quality #O #U Understanding of product, product culture, businesses and their needs #U Open-minded quality thinking #O #U Customer-centredness #O #U #A Quality advocacy #A
Links with	<- Innovation in product development <- The changing requirements of technical software systems <- Need for personal understanding of quality -> Business understanding for all -> Competences focused on business type

5.9.2 The changing requirements of technical software systems

Obviously, one very critical issue is how software systems will evolve? After all, understanding the systems that are tested and the quality of which is being assured is essential. This is obviously a very big topic, ranging on analysis of software trends to futures studies about the future of technology. Many of the most important issues are generally “known” and do not require validation by literature references. Those include the growth of size and complexity of software systems, with the obvious challenges to testing and quality assurance. Others, like the architecture of software systems, are such that they need analysis by other means than in this review.

Complexity is one important factor that we will look shortly here. It is generally understood that it is one of the main factors behind system failures. Systems are getting so complex that understanding all interactions between system elements are difficult. We see complexity on two levels. Systems are getting complex with many elements, dynamic compositions and complex interactions. At the same time, the system elements are getting complex. Consider a small element in an Industrial Internet system. It may contain a full software stack from JavaScript front end application to a full local database system and many different sensors. The technology density is huge. Miniaturisation is also in effect here – small does not mean simple anymore (as we actually have learned from smartphones already at the turn of the century).

The issues of complexity are thoroughly analysed in Dvorak (2009) in the context of space flight software. The report emphasises the utilisation of randomness in testing, explaining that the principle is known under a number of different terms, e.g., as “fuzz testing”, “stochastic testing”, and “probabilistic testing”. It is based on two principles: we cannot understand all interactions and states of a system in operation, so we need to use randomness to test all situations that can be executed.

The important thing here, from the viewpoint of competencies, is that random testing was once thought to be the worst testing technique, but currently there is a return to that – mostly due to the complexity issues. Still, Dvorak emphasises that understanding of the system, its design and functioning is essential to manage complexity and it is essential also for the testing. Random testing can only be applied efficiently when it is based on understanding of its application and its limitations.

One view to complexity is the requirement for robustness. In complex systems, any component needs to assume that any other component that it is interacting with can do just about anything. Appendix H of Dvorak (2009) describes how good engineers are proud to produce components that can tolerate any use and misuse. That is something that mirrors to the testers also: testing really needs to aim at breaking the software. This is something that is already often understood in tester community, but usually not by management.

Another issue is the size of software. Ebert & Jones (2009) describe the trend of software growth in the context of embedded software. The problem with the size of software is that it results from more lines of code, more function points, more functionality, which all means that there is more to test at all testing levels. At the same time, time and budget resources are not growing. So that implies two things: testing needs to be more effective and it needs to be prioritised. It was understood early on that software can never be completely tested, but that idea referred originally to the mathematical impossibility to test all possible inputs of functions and similar. Now, after the software systems have become more complex, there are often many modules that cannot be tested due to practical reasons. This requires new thinking from testers.

Two issues related to both complexity and size are technological change and diversity. New software systems contain new technologies with which the tester needs to get familiar fast. Technological diversity is often required in safety-critical systems, meaning that testers need to understand various means to implement some functionality. Literature of these issues is not reviewed at this point, however.

It must be noted that the requirements vary greatly depending from the viewpoint. The issues above are technological ones, but the view to what is essential can be very different when we look into how marketing sees things. For example, startup companies don’t even have a mind setting that considers complexity and size. As Giardino & Paternoster (2012) demonstrate that user experience can be the almost only thing that matters – a startup needs to get customers interested in the product fast.

Thus, the technological requirements, customer requirements and product business requirements vary greatly in different situations. The testers need some generic “mental tools” and approaches that will help them tackle what is important and spend less effort on issues that are not. One part of that is the understanding of the business at hand, but understanding as such is not sufficient. Some methodical approaches are needed that work on the level of hands-on testing.

A relatively new paradigm is risk-based testing, which is basically the idea that the priorities of testing are based on the risk involved in the functions (or requirements or similar) that are tested. Approaches to it are described by for example Bach (1999) as a tester’s general approach and Redmill (2004) from the point of view of traditional risk analysis and Rosenberg, Stapko and Gallo (1999) from a code level perspective. Risk-based approaches are increasingly used in practice and that means that testers need to understand risk, be able to participate in risk analysis and to be able to prioritise tests based on risk. The more critical a system under test is, the more essential this is. Risk-basedness is also a tool for developing the whole approach to testing. Products and projects that have a high risk level should obviously have better testing than those that have a low risk level. This is shown in practice in IEC 61508-3 (2010) standard which displays a tremendous growth in the required actions in verification and validation as the system’s risk level (called in this instance “safety integrity level”) increases. Similarly, the IEEE test documentation standard (IEEE, 2008) presents a risk based approach to the requirements of test practices and test documentation. Such approaches in safety-critical domains are not new, but now these ideas are becoming more common in generic software development domains too.

Change-competence snippet 39	The changing requirements of technical software systems
Change caused by -> enables	Complexity, risks, nature of systems -> better, focused testing
Competence implications (re: quality and testing)	Understanding of product, product culture, businesses and their needs #U Understanding changing nature of quality #O #U Understanding systems’ requirements #U Understanding technical systems #U Understanding complex systems #U Risk-based testing #A Robustness testing #A Open-minded quality thinking #O #U Cost-benefit thinking in selecting quality practices #O
Links with	-> Modern risk management -> Integrated QA

5.9.3 Products with new business models in the Internet age

Much of today's software is sold as mobile applications in device manufacturer specific internet stores. Common for those "apps", is a very low price and often limited functionality. As a result, the expectation for quality may vary. Yet, manufacturers require any applications to be sufficient quality and require some formal testing process to be carried out before accepting the application to the store. So, this model of delivery will not lower the quality standards, unlike sometimes perceived. In fact, testing can sometimes be more thorough to meet the requirements of the store than it would be without those. The costs of testing are complemented by the expectations of large sales – which of course may or may not happen, just like with other types of sales. However, as the mobile applications are usually quite compact, the amount of testing is not so great – but that varies. For example, a complex camera application is by no means a trivial thing and its price tag of like three euro hides its development complexity.

Change-competence snippet 40	Small inexpensive apps
Change caused by -> enables	App culture, app stores, heavy competitions, low cost -> business possibilities
Competence implications (re: quality and testing)	Risk thinking #U Business understanding #O #U Making compromises in quality #O #U #A Cost-benefit thinking in selecting quality practices #U Business and product concept level testing #A
Links with	-> Business understanding for all -> Agility and flexibility -> Modern risk management

5.9.4 Testing of new technology products

Every moment some new technologies are introduced that will be important in the future. Sometimes they are new manifestations of old ideas, such as the Internet of Things (IEEE, 2015), at times paradigms that have been maturing for a long time, such as the artificial intelligence, which is now beginning to change our world after a long promise. At the same time, our operating environment changes. All this produces challenges for the testing of the products and systems and therefore it is time to take look into all this. Text in this section is adapted from Vuori (2015).

New technology always fulfils some shared dreams. Mobile phones connect people, the industrial Internet does the same to the machines and robots replace the human being in dangerous work. So a positive attitude to the technology prevails and nothing else would make the materialising of innovations possible. Sometimes this leads to blind hype and assessment of potential risks and problems is forgotten. At the worst, problems can be taboos about which one is not “permitted” to talk in the field’s communities.

In a mature organisation, it is therefore good to have different kinds of people and various points of view. The testing people represent the point of view of risks and are a dialogic counterbalance to the innovators and product managers. A good result is created with people's teamwork. The dreams must be described with scenarios and stories (and some people indeed must particularly enthuse!) but as their counterweight, risk analyses must be done from various viewpoints – the customers and users of the new technology, the technology itself and the business. The risks can be more concretely uncovered when prototypes are made for products that will implement the new technology and they are assessed by testing. If some people have claimed that “testing is dead”, it is clear in the context of new technologies that especially skilled testing is needed much more than before. One challenge of the testing is to show when the new concept is not ready for the market. When that is understood early, plenty of money and time will be saved.

Because technology as such is extremely fascinating, products that use new technology are developed in a technology-driven way and use and culture are forgotten. Technology has a large value as such! It is thought if a technology could be utilised and not what would be the best way to fulfil people's needs. There is nothing wrong in this, but it requires complementing points of view during the product development from the testing side, at least usability testing should be carried out.

Technology centeredness is not unique to the adapting of hype technologies. Sometimes even traditional information systems are described with class diagrams instead of describing what will change in the business processes and users' world.

The technologies meet obstacles during the road to the market. The scheme represents the global situation and individual applications of technologies can, of course, become successes, which usually is helped if they are made for a well-focused purpose. Various testings and evaluations help in the creation of such success stories.

Evaluation and testing of prototypes are a traditional way to study a new technology product. Testing can be performed with pilot customers or with end users. The testing of user experience is essential, as it determines a lot of the product’s success. The products always need a reference to which they are compared. Only then can the buyer understand the product and can make choices. Therefore, it is important to perform comparison tests with competing products. It should be noticed that the reference product does not need to implement the same concept, as long as it has the same purpose, in which case one can estimate the ability of the product to succeed in some objective of the users – and a reference product can thus be a dipstick for the customers. The manufacturer can partly decide itself what the reference product on the market is and push that idea in its advertising.

Nowadays it has been realised that it is not best to be the first in a market just because the customers need something to compare the product with. From the point of view of quality this is great, because the first product in the market helps developers to understand the concept and works as a dipstick in the comparison with the company's own product.

Innovations' path to market

It is expected that the technologies go to the market in different ways through the populations which react in different ways to technology. Their classic description is the definition of Rogers (2003) of the different adapters of innovations, who are supposed to take technology into use in order. Description below is picked from Wikipedia (Wikipedia, 2015):

- Innovators. 2–3 % of the population. Daring, educated, have several sources of information.
- Early adopters. 10–15 % of the population: social leaders, popular, educated.
- Early majority. 30–35 % of population. Receptive once they have become convinced of the benefits of adopting the innovation, have several separate social contacts.
- Late majority. 30–35 % of population. Sceptical, traditional, lower socio-economic status.
- Laggards. 10–20 % of the population. Resist new innovations actively, neighbours and friends are the main information source, are afraid of running into debt.

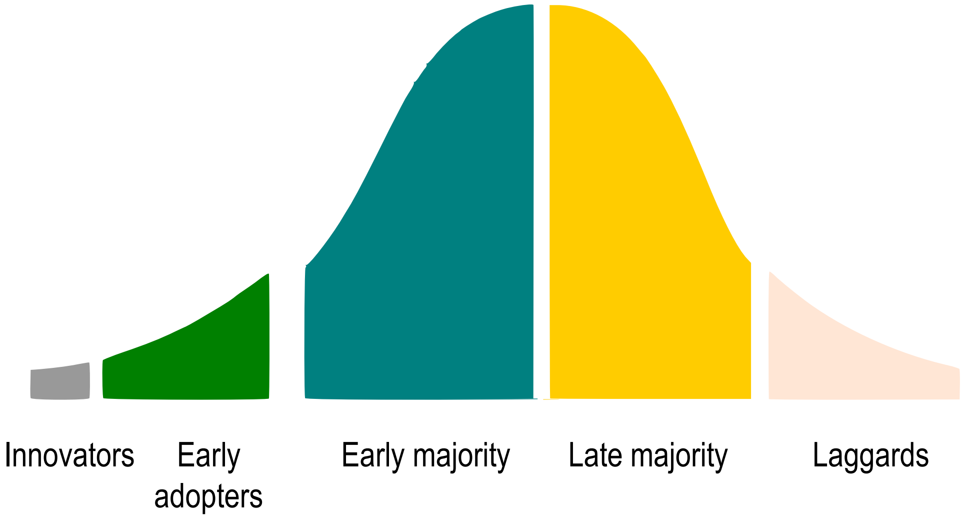


Figure 50. Adopters of innovations

For the ones excited about technology, the traditional quality of products does not matter. It is essential only that the new technology just works so that they can associate themselves to the promise of the technology and get a view into the future. This is also a form of quality of the first realisation of the product, but in very different from to what one usually means with quality – correct operation, usability, safety etc. When we are trying to reach the mass market (the early majority), the quality requirements rise radically. The technology must work as well as products usually do, and better than the old competing technology does. A conversion must be made of dreams and aesthetics into a value. At this stage, good functionality, reliability, usability and safety are needed. In other words, good design, implementation and testing, whereby information related to the qualities is produced and it can be found out how well the development has succeeded.

This growth of demands has not always been self-evident. It has often been thought that all goes nearly “by itself with hard work” from population to population, if we only get to the beginning of the path. Many companies think that if two familiar key customers are satisfied with the tailored application that was made for them, it also suits all others, if just a brochure is made for it. Becoming the favourite of the early majority is very demanding. Moore (2013) has discussed this theme. This challenge is similar to the challenges of startups that want to scale their product business after a few key customers to serve a larger customer base. See discussion about the startups’ situation in Dande et al. (2014). In that situation, they will find that more activity and competence are needed for “assuring” quality. The challenge is amplified by the fact that startups can have as the core of the product very new technology, which is still maturing. Then they really need to wake up to the issues and renew product development and product management to meet the requirements of the next phase of the company.

Many kinds of reforms are needed:

- Identification of the success factors of the following stage of the product.
- Assessment of competences and practices.
- Expansion of the scope of testing and other quality assurance activities and reassessment of their focus points.
- Tuning of the staff to meet the new needs. [25] Leadership, communication, training, special tasks.

All this changing of thinking is very difficult. So, one needs to refocus in things at three different situations:

1. When finding the concept and the main features of the product.
2. In the change situation, when the expansion of product or product line is planned.
3. In the situation of bigger volumes – more customers, more deliveries, more features.

A big part of the focusing is the understanding of what is not worth doing. When we are only searching for the best concept or try to get the first product out to the customers, building an automatic testing infrastructure for the continuous deployment pipeline will not accelerate that work at all. Instead it is accelerated by the intelligent testing whereby it is understood what the product should contain and how to make the business scalable. The time to think about test automation will come later.

Engineering and product development

The developing of technology products is usually implementing type of engineering. Characteristic of that work is focusing on the technology on the low level and doing testing with minimum effort. For maturing the new hype technologies, low level focus is needed for making the systems robust, but concentrating on that means less focus on other things. We already mentioned the users' world, more generally it is a question of planning at the product level, about thinking of the concepts. If we do that, we can see better the things that need to be tested. This change is promoted by the thought of heterogeneous teams which belongs to the culture of the agile development. When the team does not consist only of five similar coders, but also people who think about security and user experience, these “new” things are easier to get into the agenda. It is not so easy to fall into the old mode, where a potential world-beater’s usability is assessed only by making a usability test (and so late that it cannot influence the design).

[25] I was about to write “to new challenges”, but in our culture it means looking for a new job.

The second important change – which is not only rhetoric – is Minimum Viable Product thinking together with Lean Startup methodology (Ries, 2011). The basic idea is to make the smallest product that makes sense, which will then be tested and thus it is learned what the customers and users expect and what kind of product could be good. This is essential because the disruptive new technology products are never sufficiently understood and without the understanding the development work will fail. Actually, it is wrong to talk about the understanding products, because what is really needed is the understanding about customers: what do they need and want and how a product which corresponds to the needs and desires can be created? Therefore one of the new concepts of the product development is indeed "customer development" (Cooper & Vlaskovits, 2010).

The further development is made using variations and by testing them. This sounds like a traditional prototyping and in some extent it is that, repackaged – if we talk about very managed "scientific" prototyping, which companies' everyday life is rarely. The process is directed by clear hypotheses and a view about which things one wants to get more understanding about. For the testing, this produces many new possibilities and challenges in competences.

Technology complicates the world

Technological innovation seldom replaces old technology but make an addition to it. That results in, for example, the following effects:

- The products and systems become more complicated.
- Interactions between the technologies and different systems increase.
- The developers of products are on the learning curve all the time.
- The new technology is not mature and therefore the new products are immature.
- There will be new features in existing products with the new technology.
- Sometimes product paradigms are created with the new technology and the whole organisation must reform its thoughts. For example, the machine building companies can find their essence in the provisioning of logistics systems instead of assembling metal parts to form a machine.
- Sometimes the products produced by the new technologies are so different that it is difficult to determine the relation to them. Is for example an intelligent human-like robot a tool or a social actor?

Sometimes several hype-technologies meet in the one and same package. For example, an intelligent robot may combine new sensor technology, new interaction technology, artificial intelligence and new types of information management. Such a whole is especially challenging.

On occasion the hype technology may simplify the world. Functional programming languages have been for a long time coming and now they would seem to be coming closer to the mainstream. With their help, for example the handling of the rich data in the information processing systems should be easier than before.

Challenges for testing

All this causes many kinds of challenges for testing. The testing of intelligent systems requires a change in the starting points for the testing. The target to be tested cannot be seen as a deterministic automatic machine but as an intelligent subject. Studying the behaviour of such requires psychological and sociological skills. However, the tester must not respect such targets as much as they respect intelligent humans, but still an attempt must be made to “break” the targets. In other words, find the mistakes and problems in their behaviour.

When interaction between the systems increases for instance in homes, there will arise a need to perform testing of the whole system. Otherwise the problems in the interaction will not be found. This a challenge to the test environments – how to build an environment that every other equipment needed? Likewise, it is a challenge to the test automation – how to build an environment where we can stress the heterogenic whole?

When the diversity of the platforms and technologies and the dynamics of the structure of the systems increase, the systems will be more prone to have problems. System elements need to be made more robust than at present. Therefore, the testing of individual elements needs to be done more thoroughly. This raises the amount of work and expenses at all testing levels. The same demand applies to the overall systems. If the communications of the microwave oven get muddled by a software update or it starts to do denial of service attacks to the systems of the home, the situation must be controllable.

If all the things are connected to the Internet either directly or through central devices, the significance of information security will be huge. A cracker can paralyse the whole life of an individual, family or home. The security risks must be analysed for every product, solutions must be studied and their security must be tested. The analysis needs to be updated when the system changes. The risks apply to the distinctly confidential data but also to everything else – it is easy for the criminals to find out the ways of life of the family from the usage profile of the devices of the home.

We could formulate the essential principles of testing like this:

- Risk-based attitude. We are dealing with a new thing that has many kinds of risks on both business and technology. The risks must be identified so that they can be controlled.

- New technology always ties people more to technology. Therefore, the analysis of the risks of the users and customers is as important as the analysis of the risks of the manufacturer. What will happen if the technology does not work?
- Behaviour of the new technology is not understood. Therefore, it must be studied with exploratory testing, with an open mind.
- Because the different interaction phenomena and the quality of other technologies of the environment are not known, every component of the system must be as robust as possible with the help of testing.
- There are new information security challenges in all new technology. The challenges must be studied and solutions must be analysed and tested.
- New technology is more dynamic. More life cycle thinking is needed also for the testing. In addition to installation and taking a system into use, the updating of company's own products and their reaction to changes in the surrounding technology must be tested.
- All in all, it is good to be suitably paranoid.
- The success of new products often depends on user experience. Therefore, it is important to test it well in addition to designing it.

Creating understanding

Introduction of the new technology requires creation of shared understanding about issues related to it, on which designs and decisions can be made, the developing of product concepts can be directed and measures related to the quality can be planned. Company's own analysis and thinking are important because the sellers of the new technology naturally present only its positive sides (*Coke adds life* vs. *Coke rots your teeth and makes you fat!*).

The ability to read the hype is indeed important. It is good to know how to read from between the lines of market messages. A few keywords, using a slightly humorous tone:

- New = Mature only after a time. Unfamiliar to users. Does not fit the culture.
- Versatile = Complex. Difficult.
- Intelligent = Has information security risks. Difficult.
- Integrated = The working of the whole is challenging.
- Quicker = More expensive. Drains more current from the battery devices.
- [Brand] = Tied to a certain ecosystem and to a closed technology. Does not work with your own telephone.

At this stage, we must point out that the products of the new technology have three important levels:

1. New technology on an abstract level, forgetting the qualities of its implementation. On this level its general features, importance, desirability, risks and general situation in the market now and in the future are visible.
2. Components and implementations that are available. Their maturity, availability and applicability in company's own products.
3. Company's own product concepts and products which adapt the new technology.

These all the levels are worth assessing systematically. This evaluation work connects the ones operating among a product all. Examples:

- Surveying of technologies. What is the situation of potential technologies? How do they mature? What kind of applications are coming to from the technology suppliers? Are the claims about the advantages reliable? Which situations do they hold true in? What kind of known problems are there? Are there alternative expert opinions around?
- Analysis of the applicability. How is the technology suitable for company's own product palette? What must be revised to adapt it? Does it produce added value or just complexity? Is internal know-how enough or is external help needed? Are the licences suitable? Do the advantages correspond to the extra costs? Are there conflicts with the licences of open source code? Etc.
- A risk analysis of the technology from a practical viewpoint, as a part of company's own product concept. Taking into consideration understanding of the technology, its maturity, reliability, desirability for customers, safety, scalability, product lifecycle phases etc.
- A survey and evaluation of available implementations. For example, buying of available products and testing them.
- An experimental project in which the functionality of the technology in company's own technology palette is tested before the taking of it to the products in practice.
- After that, sensible product development where there is no absolute value on the technology but only the overall quality of products counts. Now testing and evaluation of all kinds of properties is needed. Often neglected is the comparison with the other products, as quality is always relative and acquisition decisions are choices between alternatives.
- Analysis and testing of user experience is central at this stage. It is good to study architecture and technology choices in the technical analysis of the product so that the scalability of the product will be understood – either into different product variations or to a larger amount of the functionality.

Change-competence snippet 41	New technology products
Change caused by -> enables	New technology -> new concepts, disruptive products, new business
Competence implications (re: quality and testing)	Understanding technology #U Evaluation of product concepts #A Understanding overall product lifecycles #U Critical thinking and presenting critique #A Doing critical technology assessments #A Hardware-related skills #U #A Understanding what qualities are the most critical at each phase of the company's lifecycle phase and the product development phase #U Risk, safety and reliability analysis #A Using exploratory testing for understanding the behaviour of technology #A Experiment design skills #A
Links with	<- Innovation in product development <- Fast product development <- The startup phenomenon -> Testing of intelligent systems -> Experimentation culture

5.9.5 Testing of intelligent systems

We have come to an age, where systems become more and more intelligent, having some autonomy in working with humans and other systems. Artificial intelligence has taken great leaps recently (see an American review of the status in Stanford University, 2016) and together with developments in sensors, computer vision and other technologies, enables the development of new intelligent and sometimes autonomic systems. The development of those is an area of opportunities, but obviously has many challenges., too..

As an example of challenges in testing such systems, we will look into testing of human-like robots (a drawing of one is in Figure 51). The text is mostly extracted from Vuori (2014a).

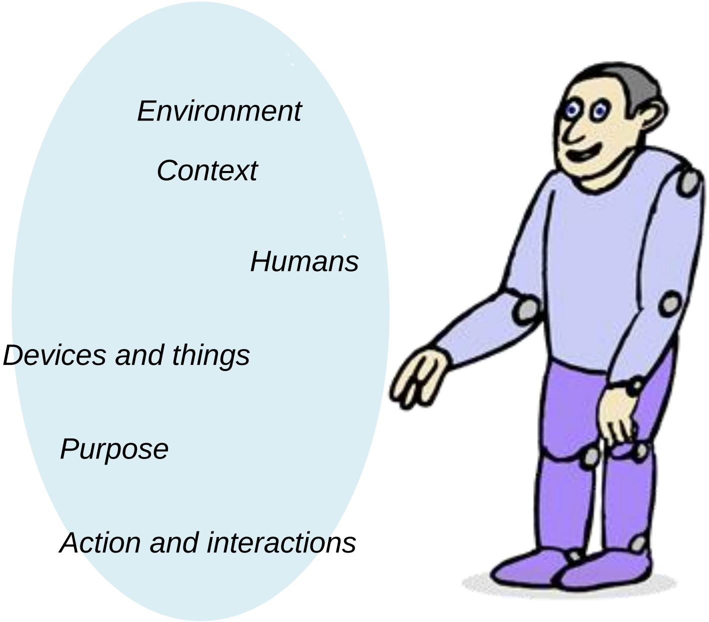


Figure 51. A human-like robot.

Of course there will be many types and configurations of the robot, having very different characteristics:

- Some targeted to be simple physical aid, able to do simple tasks – like lift things for the elderly, or a vacuum cleaner robot.
- Some targeted to be a communications and memory system for the user.
- Some are meant to be for various kinds of personal company and pleasure.
- The size may vary (midget-size is still human-like).
- Some are clearly more safety-critical than others.
- Autonomy will vary – executing simple commands versus doing tasks independently.
- Ability to learn will vary. Some are programmed by the user or the manufacturer or someone else, but some can learn new things itself.

For the sake of assessing a “worst case” – or a “best case” – we will here consider the most advanced do-it-all robots.

Table 23. Characteristics of human-like robots and their influences on the implementation of robotic systems.

Characteristic	Influence
Is a new thing	People have different expectation about them and there will be surprises.
Is human-like	May lead people to expect an unrealistic level of human-like understanding from them. Causes unfounded trust. Thus makes life more pleasant, but may cause problems with unproven technology. First guidance from when robots were introduced was: do not humanize them, remember that they are machines.
Is physical and moving	Has presence, may cause hazards by moving or blocking movement of humans.
Can lift and move things	May cause hazards by acting on wrong things or dropping things or taking them to a wrong place.
Has an advanced sensory system	Can recognize things much better than any living thing and can communicate in many ways
Is intelligent	Intelligence will be a great aid, but can be dangerous.
Can have personality	Despite the warnings above, a robot clearly can have a personality and that always means some quirks.
Is a software system	The robot’s behaviour is based on software. Software makes them suitable for a task and context, and differentiates different robots.
Is networked locally and with the world	The robot can “know everything” – and also reveal everything.
Is technologically diverse and complex	The things are hard to develop and test.

The whole environment where the robots operate is interesting. In that things happen in parallel, in non-deterministic manner. The whole system is practically unknown and changes often as new devices; people and robots join and leave the collaboration. All participants communicate in diverse ways and may have various roles in any activities (starting them, participating actively, monitoring etc.). Also, some of the elements may and will be malicious and their reliability will be unknown. Various types of networks for a basis of the communication between the robot and other actors and they cause many potential risks. This calls for “paranoid” security and robustness strategies both in design and testing.

The next table outlines the most essential testing types for the system. Note that in this kind of presentation, the system elements are not independent – for example the control system cannot be separated from sensory system and the “intelligence system”.

Also note that here we discuss the software and behavioural aspects and not much the testing of the physical robot.

Table 24.Testing of various elements of the robot system.

Element	Test types (most essential)	Special challenges
Overall system (robot in action, in environment, in collaboration, as part of systems).	Concept testing (analysis, simulation, mock-ups).	Validating that the robot concept is the best one for the context, goals. Validating that the robot has a cultural fit to where it will operate.
	Functional testing.	For automated testing: Environment simulation, programmatically created user gestures, voice commands… For model-based testing: Modelling of environment (elements and behaviour) – including devices and humans. Use cases / stories for both humans and the robot. Exploratory testing important due to complexity. Testing the operating logic in a simulated environment vs. testing the physical robot in the real world. Changing environment setup. Need a paranoid approach to how other system elements behave.
	Safety testing.	Need a thorough risk / safety analysis for a test basis. Testing requirements from safety standards – advocate advanced techniques, such as model-based testing. Safety is related to security too – consider dangerous remote control of devices.
	Security testing.	Low level of trust in any system elements.
	(Regulatory) validation testing.	Unclarity of the regulations and their interpretation, unclarity of what standards are applicable.
	Performance / capability testing.	–

Element	Test types (most essential)	Special challenges
	Compatibility, co-existence testing.	Testing of the diverse technologies and variations in the collaborating environment.
	User experience testing.	Need to assess overall relation between the humans and the robot – is it as planned?
	Localization testing.	The whole behaviour, the meaning of control gestures, behavioural rules – it can really be cultural testing of cultural fit (by no means checking of translations).
	Upgradeability.	Testing of updating of software or hardware.
Control system	Functional testing.	Testing of movement in practical spaces.
	Reliability testing.	Reliability analysis as a test basis.
Intelligence systems	Testing of logic and decisions.	All deviations, non-determinism, context data.
Sensory system (perception system)	Functional testing.	(Depends on sensor). Variation on input – gestures, sound and ambience
	Reliability testing – defective sensor etc.	–
Safety system	Functional safety testing.	Testing requirements from safety standards (such as SFS-EN 61508 series) – can be very demanding! Needs safety / reliability analysis for basis.
Communications system (technical)	Functional testing.	–
	Reliability testing.	–
	Performance testing.	Including load, stress testing.
	Security testing.	–
Human interface (user)	Usability testing, analysis.	The new ways of interaction can be difficult to validate.
	Analysis and testing of human errors.	Must test for human errors thoroughly (voice, gesture commands).
	Obedience testing.	Who is in control, when many humans are present (or TV is on).
	Functional testing.	Exploratory testing is critical – need to have almost a “psychological” approach.

Element	Test types (most essential)	Special challenges
Human interface (programming & configuration)	(The same as for the user interface).	
	Security testing.	Who can program / configure the robot? Consider remote control.

Testers are humans and as such they are prone to the same psychological phenomena as the users of the robots. They may tend to treat the human-like robots with awe, respect and care. That is the enemy of good testing. Good testing should aim at breaking the software (though not to breaking the physical robot) and that obviously requires that we do not care about its well-being. The more in trouble the human-like robot gets in testing, the better! So we need pay attention to the testers' attitudes.

Another phenomenon is that people extrapolate their testing approach from history and previous projects in "just enough" manner – if nobody complains about the inadequacy of the approach, it must be ok. But when the systems under test take a leap in challenges – new concept, new level of complexity, new types of systemic interactions, large amount of new technology, a mix of different development cultures – the whole testing should be reassessed. It would be an error to think them as just another programmable device, yet another type of automation or a more serious toy.

Indeed, they can be very different. Traditional products usually employ strict rule-based programming, which makes the behaviour predictable and thus relatively easy to test. But when a device is a learning one, the situation is different, as the device learns continually and if it is to work with humans, it needs to learn from the interaction with humans. Especially, if the system is safety-critical, the configuration is supposed to be frozen for testing and deployment, but that is clearly not the case. The safety strategy is difficult to make inherently solid and one must resort to the separate safety architecture to – hopefully – catch the device doing dangerous things, and make that as solid as possible. But an intelligent system will produce surprises by design. In general, the testing related to learning should assess how the robot learns right things, learns them right, does not learn dangerous things, validates its learning from the uses as appropriate, and despite its own intelligence, obeys the user.

What could be the strategies for testing the intelligence? The first challenge is to try to makes sense, with exploratory testing about the logic the system uses. To reveal the intelligence, the test scenarios for the robot's behaviour need to be open-ended. The tester must not respect the intelligence, but lead it to troubles to expose how it behaves in those. One needs to really tax the system. There is a need for almost psychologist's competences. One must always suspect the intelligence and try to find its limits. Conditions and situations need to be varied to see how the intelligence deals with different sensory inputs. It is essential that the tester needs not know the actual implementation of the intelligence, only the behaviour is various situations. There is more need for good test models that models of the artificial brain.

Analysis of possible misuses, either intended or not, is required for all systems and intelligent systems in the human environment are immediately subject to playing with and making them do things the designer did not plan them to do. Here, testers can collaborate with designers in the analysis, and will as well as they can, test the system for such usages and test the controls against them.

With any complex systems, testers may find it difficult to find a place to start. The traditional division of the system to layers helps here. With robots, layers can be identified such as:

- Sensors and actuators – logic – behaviour.
- One device – device pairs – flock of devices.
- Accuracy, force – speed, fluency or action – ability in scenarios and use cases.
- Isolated simple testing of an issue – simple situation combined with other elements or actors – complex interaction.
- Local activity – local area / context / network – global situation.
- Software – integration of software and hardware (incl. user interface) – overall product.
- Etc…

This is in part related to the technology stack and in engineering it is natural to validate the stack layer by layer. Safety standards also emphasise such approach, but have at the highest level not "technology", but the overall product or system more as a behavioural entity.

All this requires better than normal testing competence, preferably more than one tester with complimentary competences. For example, UX testing competence – not just usability testing – and understanding of automation systems and safety-critical systems are especially essential to have in the core team. Security analysis and testing skills can often be “outsourced”. The hardware-related testing competences will depend on the nature of hardware development and sourcing. In a context like this, an important meta-competence is the ability to understand what kinds of competences are needed in the development and testing and to be able to reflect one’s own competence against that.

The testers need also be familiar with design guidelines, such as SO/TS 15066 (2016) for collaborative robots and the relevant generic or domain-specific safety standards.

One of the functions of testing is learning about the things that are developed. Good testing cannot be “execution”, but an attempt to understand the new things, to gain insights to act upon.

The area of intelligent robots combines competences of various disciplines, both in development and testing and quality assurance, see Table 25. Those rarely exist in one company, placing challenges for both education and organizational design. The collaboration between people of various disciplines obviously emphasis team skills.

Table 25.Testing area and related disciplines.

Testing area	Discipline
Physical action, actuators and sensors Safety	Industrial automation
Product concepts Cultural fit Usage patterns User interfaces User experience	Industrial design User experience design
Software technology Data handling Information networks	Information technology
Information security	Information security
Intelligence	Artificial intelligence

Change-competence snippet 42	Testing of intelligent systems
Change caused by -> enables	AI, robotics -> human-like robot, intelligent software systems
Competence implications (re: quality and testing)	Broad flexible competence #O #U #A Understanding new products and systems #U Evaluation of product concepts #A Understanding complex systems #U Testing of complex interactions #A Security assessment and testing #A Hardware-related skills #U #A Safety management #U #A (on safety-critical domains) Risk, safety and reliability analysis #A Understanding innovation #U Open-minded quality thinking #O #U Understanding users #U UX and usability testing #A Team skills #U
Links with	-> Information security and privacy -> Need for new types of workers -> Modern risk management <- New technology products <- Multi-device systems with new interaction styles

5.9.6 Changes and implied competences for this section visualised

A visualised summary of the change-competence snippets in this section are visualised in Figure 52.

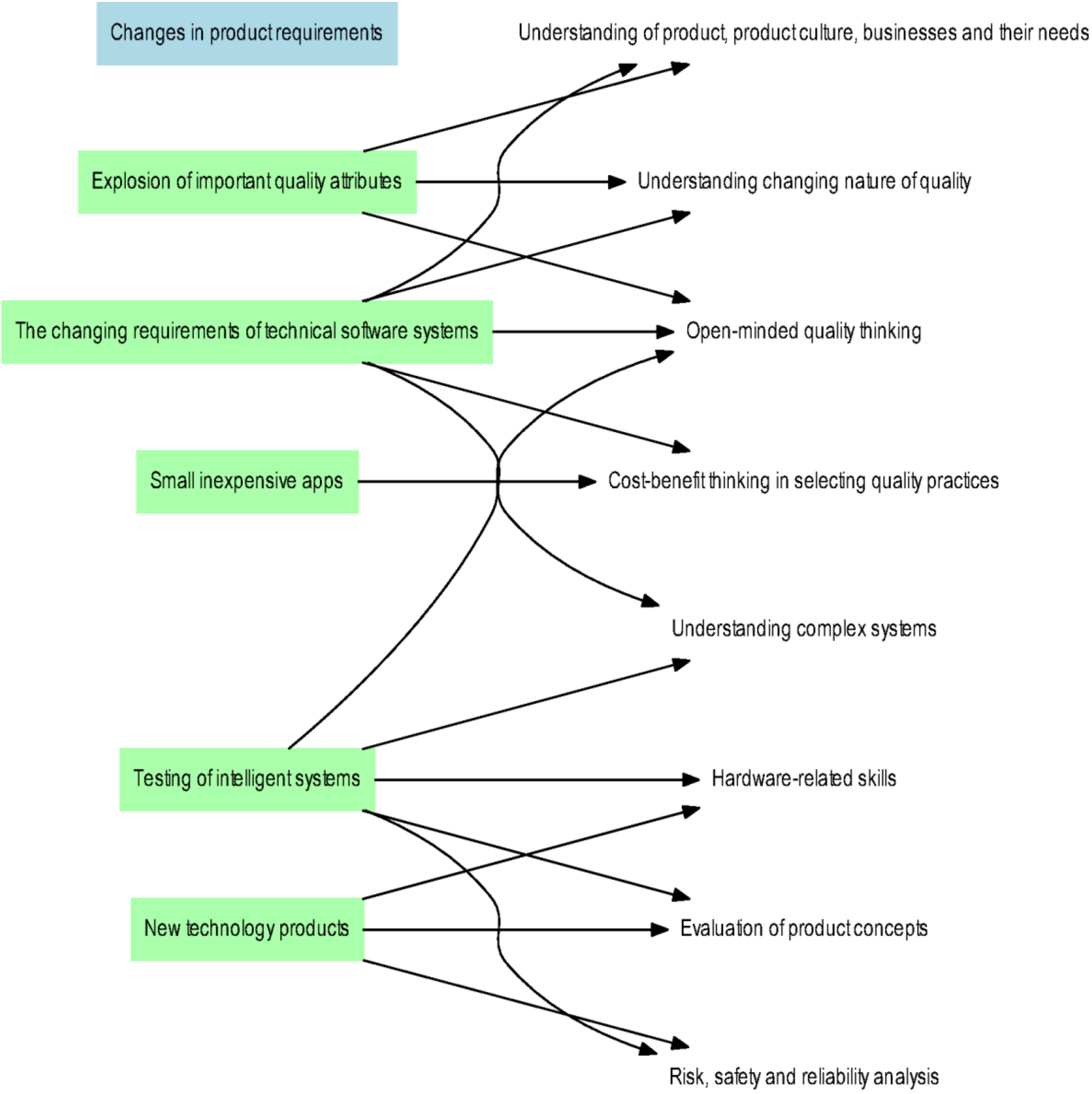

Changes in product requirements
Explosion of important quality attributes
The changing requirements of technical software systems
Small inexpensive apps
Testing of intelligent systems
New technology products
Understanding of product, product culture, businesses and their needs
Understanding changing nature of quality
Open-minded quality thinking
Cost-benefit thinking in selecting quality practices
Understanding complex systems
Hardware-related skills
Evaluation of product concepts
Risk, safety and reliability analysis

Figure 52. Visualised summary of the change-competence snippets in this section. To make the graph fit the page with legible font size, only the competences are included that are associated by at least two changes in this section.

5.10 Software development process changes

5.10.1 Innovation in product development

Successful innovation is absolutely critical for any product development. One big part of that is the first phase of development, where the company tries to find out what to develop. What concept could be the best one for the problem or opportunity at hand? Let’s look into the process of product development and the various ways it can be started. This list was made when the author was involved in the development of university education in the innovation project work context – an innovation oriented version of a project work course (InnoPilotti, 2011):

1) Closed / old concept – systematic process: Requirement specification -> design -> tailoring -> incremental development.

2) Open concept – systematic process: Needs / goals -> concept design -> detailed design -> implementation -> developing maintenance.

3) User-centred design – systematic process: Understanding needs -> specifying context -> design / evaluate cycle -> developing maintenance.

4) Participative design – systematic process: Scoping of the problem -> users build the solution (mock-up) aided by designer and process -> implementation -> developing maintenance.

5) Open concept – innovative process: Visionary concept birth (genius) -> key characteristics -> fitting to users -> renewals for generations.

6) Open concept – innovative and systematic process: Scoping -> criteria -> ideation of alternatives -> evaluation, selection, development -> implementation -> developing maintenance.

7) Lean start-up: Triggering idea -> core need -> minimum implementation -> testing -> variation -> birth and understanding of concept -> making it deeper and wider.

8) Organic development: Existing implementation based on concept -> incremental transformation -> becoming of another concept -> development.

9) Order – delivery: Customer's requirement list – implementation.

10) Director decides: Director makes a specification -> implementation.

11) Copying: Competitor's implementation -> own implementation.

12) Updating of old: Current version -> implementing wishes for improvement -> new version.

13) Fitting of old product to new context: Current version -> fitting to new target group / price position etc. -> detail design -> implementation.

Of the types, only some are so open-ended at the starting phase that they allow concept level innovation. Lean Startup (7) is a common approach to more innovative development, but for some decades there have been an idea of “fuzzy front end” to the product development (see for example Koen et. al, 2002 and Miller, 2002), in the list this maps to number 5. In this thinking, the development process is divided into three areas: the fuzzy front end, the new product development process, and commercialization. The first phase is the fuzzy phase, which is chaotic, unpredictable and uncertain. We don't know what we should start developing seriously and when. In practice the phases may not be separate, but there is a flow from chaos to systematic development. The classic funnel model visualises that in Figure 53.[26]

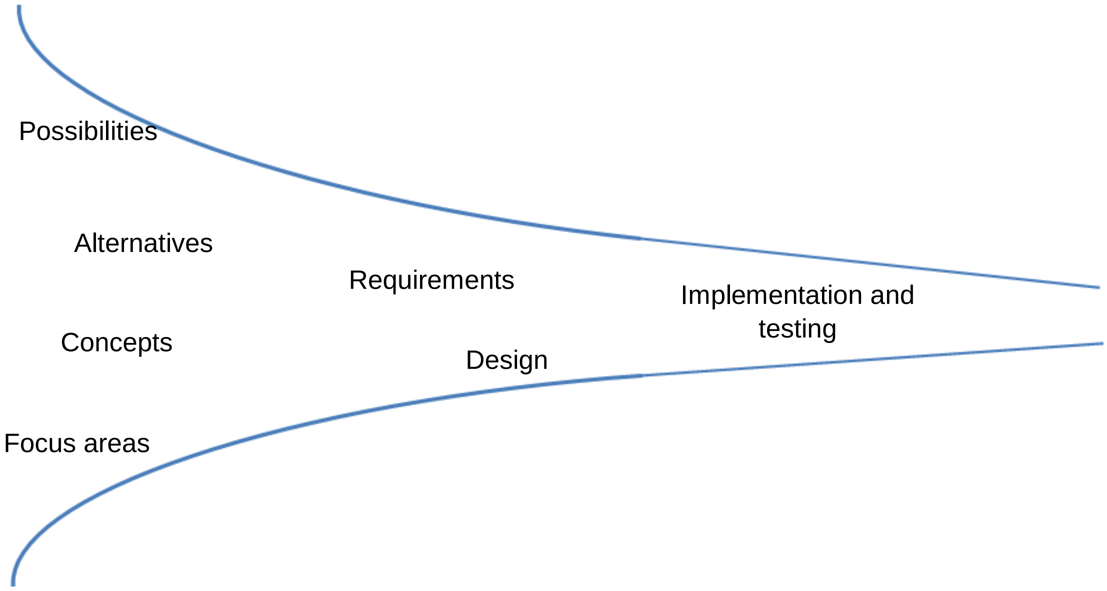


Figure 53. The funnel model that visualises moving from the fuzzy front end to the systematic development and gaining focus.

Yet more important than a process is the activity system of innovation, because that forms the living, adaptive system where the innovators work, where things are thought, developed and tested. Its elements are presented in Figure 54.

[26] This kind of funnel or cone picture is quite common. This one is by the author.

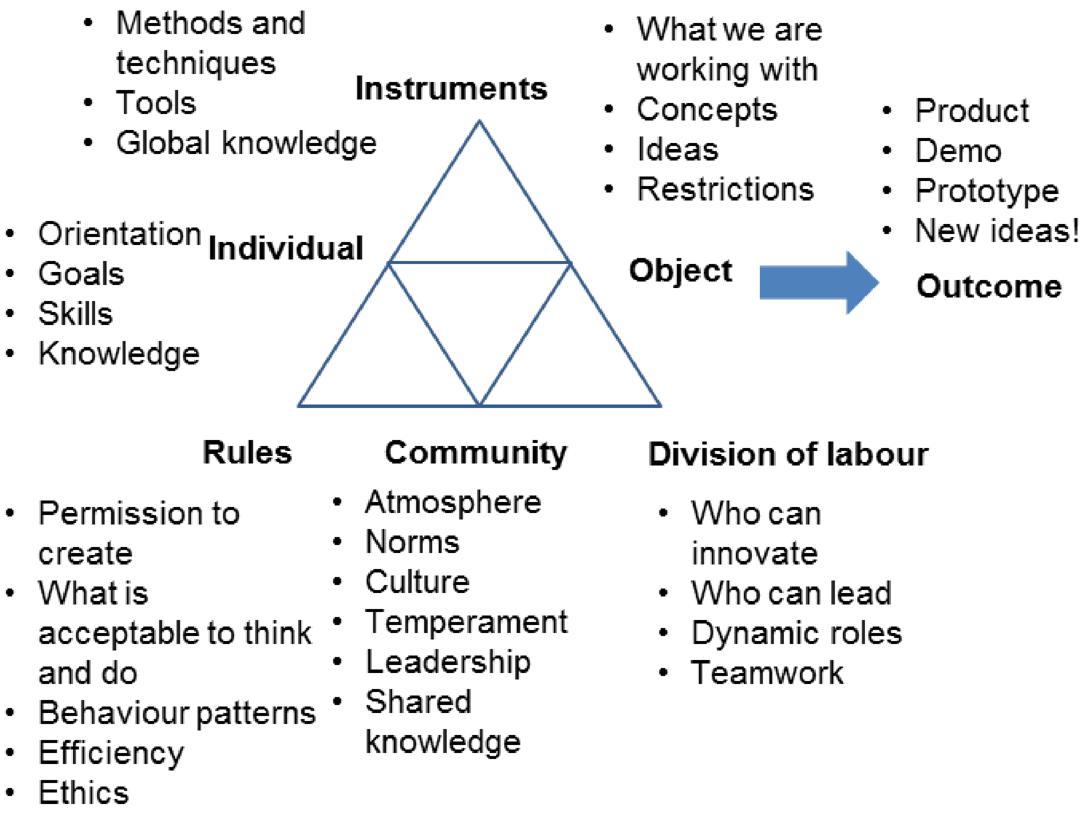


Figure 54. The activity system of innovation (structured by the triangle model used in action research), adapted from Engeström (1999). This time it is used for describing innovation.

Traditionally, when testing is discussed, everything is located in the linear development process phases, which proceeds rationally from concept development and requirement specification to technical design and so on. This is the time when inventions happen. Making inventions should be supported by available testing skills, which are utilised deeply integrated into experimentation.

Of course this is usually informal work in the teams, except in those where this phase is done in dedicated research centres, but those are rare nowadays.

Change-competence snippet 43	Innovation in product development
Change caused by -> enables	Need for new products, disruption -> new business
Competence implications (re: quality and testing)	Understanding innovation #U Understanding of product, product culture, businesses and their needs #U Understanding overall product lifecycles #U Understanding the product development paradigm #U Evaluation of product concepts #A Critical thinking and presenting critique #A Prototyping skills #A Doing proof of concept tests for technology #A Experiment design skills #A Understanding of needs of development #O #U Doing experiments with users #A UX testing for feature development #O #U #A Working under insecurity and change #O Team skills #A Understanding about the company's business #U Understanding customers #U Understanding about ethics #O #U Doing ethical assessment #A
Links with	<- Changing Finland -> Industrial Internet -> Changing engineering education -> Experimentation culture -> Fast product development -> Rethinking the goals of testing and quality assurance

5.10.2 Relation and response to change

Testing has traditionally been against change, as has the whole systems development culture. For the development processes, change has been problematic as the processes were based on large batches of features and designs, and making small changes would need special change requests handled by special means. If the projects would actively accept changes, they would often just run after the changes and not do much new feature development. This was a difficult situation, as experts usually agreed that software development is a learning process and the customers and developers really understand during the development and after the first implementations how the system should be developed. The development processes were, luckily, rarely of pure waterfall type, but consisted of several iterations, yet not on as fast rhythm as in today's agile development.

Testing would especially be against change, as changes would mean throwing away functional test cases that were carefully crafted against the designs that would not even be implemented. Similarly, carefully planned test automation script would need to be rewritten. But the Agile ideology and agile development practices changes the overall attitude towards change. Suddenly, people were positive towards change! This happened at the same time when companies saw that businesses need to be more adaptive, which would mean accepting changes fast – during development projects and not just after them.

So, in general, there was a new understanding that change is usually good, if there is a need for it. Change that is the result of bad planning or designs is also a good thing, if it is caught during the development. Still preferably one would of course like to have less such changes and more changes that result from learning.

There have been changes in development practices that greatly help with dealing with change and sometimes reducing it. Sprints, as used in Scrum (Scrum Guides, 2015), the very common project framework today, reduce pre-planning, make development more incremental and thus enable focusing on development based on very current understanding and changing of opinions without too much change on already made implementations. If there were to be changes, unit testing and continuous integration practices provide a safety net for making changes without too much regression risk. Moving from pre-designed test cases to exploratory testing would really enable changes from the perspective of testing. Leaner communication practices help in agreeing about changes without resorting to committees and special process flows. Yet the needs do vary; this is just the general situation.

In general, testing should embrace change when there are reasons for it and use agile practices to manage the evolving situations. Testing is also a critical instrument in validating a need for change. Usability testing, user interface testing and proof on concept testing should be used to find out whether an idea is worth implementing. Therefore, the competence needs are moving in two directions: the ability to assess ideas and the ability to work with the changes. Both require learning of attitudes and working methods and integrating the testing approaches more with product planning and design activities.

Change-competence snippet 44	Relation to change
Change caused by -> enables	Dynamic environment -> change offers opportunities
Competence implications (re: quality and testing)	Adaptability and flexibility #A Right timing of actions #A Short work-in-progress lists #A Understanding software engineering #U Reflection on working styles #U #A Rigour in collaborative keeping the platform robust and tolerating change #A
Links with	<- Agility and flexibility -> Agile software development -> Lean

5.10.3 Time and rhythm

In the discussion about SEMAT, it was noted that time is always a very critical element in all work in many ways and on many levels. On the business level, products have a window of opportunity that starts at a given time and may close at another time. Projects obviously start at some time and end at another – that is one criteria by which we call some activity a project. All process models may have a given rhythm based on iterations or development sprints.

Quality assurance and testing must adapt to those “natural rhythms”. Testing activities may be rhythmic themselves, such as doing system testing at the end of development sprints (thus having the same rhythm as development). Sometimes, they may have another rhythm that is harmonic with the development rhythm, such as doing testing for product launches at a more leisure pace than feature development. Those rhythms are slow. Faster rhythms can be found in daily work and obviously in the use of test automation.

Much of the rhythmic nature did not exist in previous decades. The only rhythm might have been the slow annual rhythm of releases of some slow evolutionary rhythm during a development process. Mostly the process was linear. In linear processes, timing is everything. The process moves ahead and testing must be able to “grab” an idea or artefact for testing at exactly the right time and deliver the wanted information to be available at exactly the right time when it is needed (or just a somewhat earlier – if information for decision making is produced earlier than that, the information can easily be forgotten and neglected).

Many processes combine rhythmic and linear characteristics. Almost all development projects proceed in linear fashion, but contain iterative rhythm. Even in Kanban-based processes there will be some stable rhythm if when the size of the tasks is similar and the development speed stable. “Continuous” deployment can be rhythmic to make things easier, while there is a capability of breaking the rhythm if needed. Humans and organisations simply are built for rhythm!

To understand the dynamism of the product development, let’s consider it as an acoustic system consisting of volumes and connecting pipes that have a length and a diameter. This is because even “continuous” activity is not truly continuous, but information and activity always moves in a rhythm. The system is presented visually in Figure 55.

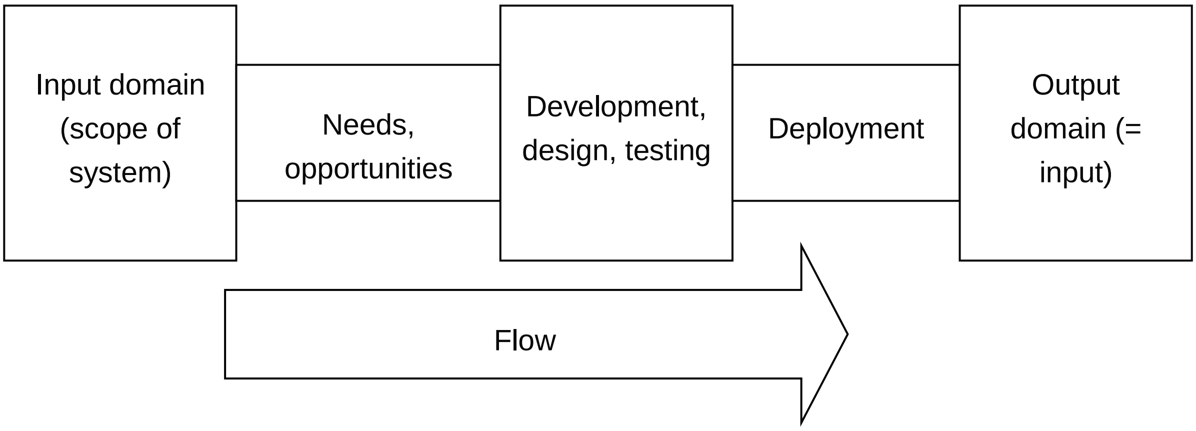


Figure 55. The development process as an acoustic system.

A thicker pipe can pass more information, but only if the frequency of the system is suited to the frequency it is driven. The basic ideas in acoustics are that a shorter pipe has a higher frequency. When it is connected to a volume, the smaller the volume is, the higher the frequency and vice versa. The thicker the pipe, the higher the natural frequency is and vice versa.

Using this analogy, we see that we can achieve higher speed if:

- Input domain is restricted. That helps us to better understand the requirements of the system and control the focus and size of the developed system. It is also easier to understand risks, design and execute the necessary testing. Smaller, more focused systems have fewer defects and require less testing than systems that have a wide, non-focused scope.

- There is a short path from customers to development. That way, information can flow fast; there are fewer errors in understanding things. This is why agile teams emphasise direct customer participation.
- The resources for development and testing are large.
- The deployment pipeline is short. Of course, it must not be too short, as it needs to contain the checks that the new functionality is something that should be passed to the customer. But many of the checks can be integrated into the development "volume".

Those are the basic acoustic principles. Those should be understood and considered when building workflows and processes. Indeed: the designs of development workflows do reflect the principles. As the overall speed of activities is rising, the importance of acoustic is increasing as well.

All in all, the actors in testing and quality assurance need to have time related skills. As a basis for everything, they need a general sense of rhythm. Secondly, they need ability to time their actions well. There is a time window of some optimal length for any activity, so a skill is needed to plan activities so that they can be performed in that window.

Change-competence snippet 45	Timing and rhythm
Change caused by -> enables	Agility, speed, efficiency -> flow, efficiency
Competence implications (re: quality and testing)	Sense of rhythm #U Right timing of actions #O #U #A
Links with	-> Designing new development lifecycles -> The next steps in software development lifecycles -> Agility and flexibility -> Agile software development -> Lean

5.10.4 Towards continuous delivery

Continuous delivery of software to production has been talked a lot during the early 2010's. Traditionally, software has been delivered to production incrementally, perhaps twice a year, after a batch of important enhancements or after a sprint as in Scrum (Scrum Guides, 2015). Perhaps the systems could be delivered even many times a day, after any small change. Humble & Farley (2011) describe this in great detail in their book.

The main value of continuous delivery is that when the changes are small (the “batch size” of changes), the introduction of the new system has a lower risk than when there are many changes. This could obviously be possible with an almost completely automated delivery pipeline, which may yet have phases for manual testing.

The competences required are very much related to test automation. Everything should be automated. Because the configurations that are delivered must be carefully defined, configuration management skills are essential. This all requires some non-technical competences:

- Disciplined attitude and very professional attitude. Every small detail must be correct for this to work.
- When to deliver is a business decision and required understanding about the customer’s business and needs.
- As the whole system is automation-oriented, there is pressure to automate everything and testers must carefully assess what kind of manual testing is still necessary and need to negotiate that to be included in the process.
- The paradigm presents acceptance testing as something that can be mostly automated. The idea that the software producer could define and carry out acceptance testing of behalf of the customer is dangerous, even though user acceptance testing is sometimes briefly mentioned as something that is not automated.

Change-competence snippet 46	Towards continuous delivery
Change caused by -> enables	Reactivity, speed of deployment, deployment risk control -> capability used for various business benefits
Competence implications (re: quality and testing)	Discipline #O #A Deployment and automation skills #A Configuration management #A Rigour in collaborative keeping the platform robust and tolerating change #A Process development #A (integrating manual testing into the workflow) Supporting deployment decisions with assessment and test information #A
Links with	-> Lean -> Fast product development <- Timing and rhythm

5.10.5 Fast product development

The general aim of fast product development is related to the goals being first in a market, bringing rapidly new value to customers and being able to utilise new technologies as soon as possible. The issues were discussed in Vuori (2014c) and some of the main ideas are condensed here.

First of all, we need to remember that there are many types of speed. Some of them are listed in Table 26

Table 26. Types of speed.

Type of speed	Description
Fast continuous speed.	Fast product pipeline. New products are brought to the market at a fast, yet sustainable, pace. High continuous speed of producing new value to the customer. For example, new features are published “constantly”. Continuous deployment. Regular updating of the product already in use, for whatever reason. Short time from entry of the development process to the exit of the deployment. Note that the velocity of output does not necessitate that the time spent in development is short – as analogy consider a highway that can be hundreds of kilometres long, but “outputs” cars every second.
Fast projects.	Fast time to market. The time from idea / concept to market entry is short, meaning that the product development is done at a fast pace. Fast velocity in large projects keeping their schedule reasonable. Rapid pivoting of a product. If a product needs refocusing, it is done rapidly.
Fast response.	Fast response to any action from competition. For example, when a competitor publishes a new feature, own product is updated to match that rapidly. The “browser wars” between Netscape and Microsoft in the 1990’s is an example of that and really emphasised high speed in software development for the first time. Rapid reaction to emerging needs. When a need for a new feature emerges in the clientele, it is met with rapid updating of the product. Rapid response to problems, threats and risks. Rapid bug fixes and hot fixes. Rapid response to changes in collaborating systems, including other systems in the overall architecture, social media APIs.
Fast change of the platform or the ecosystem.	Fast introduction of products to another operating system or device platform (besides old platform – that is, turning into multi-platform provider). Fast porting of products to other environments. Fast changing of the primary platform – abandoning old, embracing new.

So, there clearly are different types of speed and they require different competences: the ability to have a high average velocity is a very different skill that the ability to take a spurt as needed. It can also vary whether the process is dominated by a rhythm or more linear logistics.

Risks of high speed

Maximum speed is not optimal, if the speed compromises the business mission. For example, buses from Tampere to Helsinki could make the trip faster if they didn't take any passengers on board between the cities. High speed does produce benefits, but there are also risks in it. For example, continuous updates lower the desirability of new product versions, making it harder to market and sell them. And if speed is not managed, all forms of quality can suffer. The latter is due to practices not being developed suitably for the speed. Examples:

- If testing is not good enough, new product releases can contain defects.
- Trusting too much on test automation can let defects slip past that would have been caught by manual testing. If automated regression testing is not sufficient, trivial regression failures can happen. For higher development speed, both automated regression testing and manual testing need to be improved.
- If the product architecture is not designed for agile changes, it will rot and the product will be harder and harder to maintain.
- If the new features just cumulate and add to the feature count, the product will gain complexity and will be buggy and hard to maintain – just like any current "high end" product.
- If documentation processes do not have the necessary resources, documentation will lag behind, causing problems to the users.
- Quality of user interfaces will often suffer. Important functions are forgotten to include in new revisions and usability assessments and testing are neglected – not to mention a good design of the new functionality.
- When startups evolve to a point where they start growing, gaining more speed in every activity is essential. If practices are not adjusted or changed to meet the new needs, problems will arise.

Clearly there are many risks in getting up to speed and various types of testing can be used to control some of the risks. Yet, the most important thing is that speed is not the goal, it is just a tool for something – maximised customer satisfaction, keeping products ahead of competitors etc. That is the biggest risk: trying to reach speed for speed's sake, without considering why it is done and what are the things that would cause the biggest benefits for business.

Existing technical debt and process debt hinder speed improvements and working at higher speed will adjust the accumulation of technical debt. So in every speed improving activity there needs to be a focus to various kinds of debts.

One of the challenges is to be able to have the speed at the long term. At one phase of a company the focus of speed may be in concept level innovation, and in other phases in “feature logistics” – giving small new things to a varied clientele. Allocating the always too few resources to wrong things can be hazardous. For example, if a startup puts all its energy in the development of a multiplatform continuous deployment engine, it will have less energy to use in the critical task of developing desirable products. Good, fast processes that are in place, will tie the personnel into existing things, not allowing for mental renewal. That is agility gone horribly wrong.

One of the problems is that engineering mentality may start to drive business speed, not just enabling it. Speed of product development is a business issue at heart and thus needs dialogue from various viewpoints, such as customer needs, business opportunities, product development and innovation processes, testing as provider of information, production capabilities and enablers, quality management and risk management.

Deployable does not mean desirable

A mental risk is to see product development as “logistics”, as operating a value transferring machine. In that mind-set, speed records are broken what people think less about what is being produced – humans being humans. Testing gets narrower focus, moves from validation to verification. Just like when travelling fast, one concentrates on checking whether we are on the map where we should be, rather than whether it makes sense to be on that road.

One must remember that software development is not serial production. Every new feature, every new build, every “item of value” is something unique and must be assessed carefully. There must be time for good validation, because it cannot be left for the customers. It cannot be done in a speedy deployment pipeline, as it requires an integrated whole and peaceful mental environment.

Sometimes the levels of quality are presented that any new feature must pass, for example:

- The very first level is safe and secure. Nothing must be deployed to the customer that is not safe and secure.
- At the next level is deployable. Something can technically be delivered to the customer if needed. That does not mean that it should be deployed.
- The addition needs to be useful. The new functionality is in itself something that can be utilised for some purpose.

- The implementation must be technically solid. The new feature has no technical problems.
- Technical quality is not yet sufficient; thus the next level is usable. The new functionality can be used easily and without risks for its purpose.
- Good user experience is often essential. While usable, the new feature needs to offer good experience for the user.
- At the highest level is desirability, which combines other factors. When something is really desired, then the new deployment is seen to offer high value and is received with gratitude.

These (or equivalent) are the levels that should be assessed and testing should have a role in that.

Priorities on those levels obviously depend on the type of product, see table below. Of course the priorities vary case by case, so this is just a visualisation that aims to show what kind of characteristics should be designed and assured by testing and other means.

Table 27. Priorities of quality levels for various system types (simplified). *** = critical, ** = important, * = relevant

Quality level	Consumer product	B2B information system	Infrastructure technology	Safety-critical system
Desirable	***	***	***	
Good user experience	***	**		
Usable	**	***	*	***
Useful	*	**	***	**
Technically solid	**	***	***	***
Deployable	***	***	***	***
Safe & secure	**	***	***	***

Does more speed mean more work?

Higher average speed obviously means that more features are produced per time unit, meaning more things to test. That could imply a need for more resources, but the speed could also be reached with current resources when things are done more effectively, with perhaps a leap in process capability. More rapid reaction to needs and external events does not mean that more work is done on average. Work is just done fast when it is needed. Fast pivoting or fast concept creation is more a matter of product development competence and utilisation of technologies that enable fast product creation. The main issue in all the types of speed is that everything must be controlled professionally. Otherwise there will be chaos and lots of unnecessary work.

Traditional use of testing to gain speed

A traditional misguided strategy has been to just skip testing or do less of it to gain speed. When the most testing has been left to the end of a project and there is a hurry to get the product out, testing was just skipped or done less.

Of course, nowadays most people understand that testing must not be left to the end of the project, but should be started in the beginning and done continuously with an aim to keep the product technically stable all the time – ready for deployment, stable to handle any changes without falling apart.

Healthy principles include:

- Test in parallel. Let testing proceed at the same time as development.
- Test less. That requires less implementation, a more focused product, which may be desirable. One development principle is to “develop for now”. That means that testing should focus on the most important issues in the next release – yet at the same time keeping the platform robust for any new developments. Good up-front work on user interfaces reduces the need for testing later (but does not remove it).
- Test faster. Use automation and competent testers.
- Test in advance. Use well-tested components.

A need for many layers of thinking

Testing and its ideal are seen to be part of the engineering paradigm. But so is the general product development culture! Only gradually are we learning that engineering is not product development – we need a broader view to the systems under development and to the needs of the customers. An eye-opener was at a time the understanding that paper machines need to be designed to be attractive. Before, the designs focused only on fluid dynamics and now the visual design language of the system. Making the machines more desirable in their looks boosted sales. Similar learnings will happen in many domains today. Of course, we need many layers in testing thinking and action, all of which fulfil their particular needs. The problem is when testing is too concentrated on some level, ignoring others. The table below visualises the thinking differences between engineering and product-oriented testing.

Table 28. Comparison between engineering-oriented and product-oriented testing (caricature)

Element	Engineering-oriented testing	Product-oriented testing
View to system	Internal view, structure, functions, architecture	External view, customer value, requirements
Metrics	Absolute	Relative to competition, past
Source of quality criteria	Standards, own views	Customer desires
Compromises	Coverage	Technical versus filling needs, including release speed
When testing is done	When closing criteria are met (such as coverage)	When it has found the information needed for decision making or activities
General testing basis	Specifications	Reality, customer's world
UI testing basis	Usability, standards	User experience, reaching goals, usability
Tester focus	Defined testing tasks and methods	Information provision using any means
Process approach	Testing lifecycle	Agile response to information needs
Role of test automation	Should test "everything"	Frees testers to use their mind for finding new information
World view	Testing creates control	The world is insecure

Still, it must be remembered that testing style and culture are always a reflection of development style and culture and need to have a suitable match. Development of testing requires assessing the whole product development activity.

Mental landscapes and competences

Doing things rapidly means that there is not much time to ponder and discuss things. Developers and testers must have clear understanding of critical issues and a risk-aware mind-set. In fast workflows we see in practice the principle in Toyota production system that everyone may stop the process at any time – for example, a manual tester must have the courage to say that this build is not good enough to be deployed to customer.

All modern testing requires understanding about the business and the customers' needs and this "domain of speed" emphasises it.

Strictness is needed to keep the processes rolling and that applies to testing too. Tests must never be skipped. One good feature of automation is that one can never think that “this is just a simple one-liner and it doesn’t need not be tested” – it will get at least some automated regression testing automatically. When manual testing is integrated into the workflows, it will be (or at least should be) signed off by a manual tester. Of course, this is how things should work, and not how they always work.

One needs to differentiate the constant throughput and the time spent on development. Which one of those is more important, needs to be analysed and things balanced.

More essential than the speed of a train is that you get to the next station without crashing…

One needs to remember that speed is not a value as such. Number of deployments per time unit is no value as such – or rather it is waste. Value for business is the goal.

Table 29. Some essential tester competences for two main types of speed (simplified)

Competence	High velocity	High reactivity
General competence	High generic tester key competencies	High generic tester key competencies Large personal “toolbox” with which to tackle fast any new testing task Ability to work fast and in agile manner Exploratory testing skills
Risk related	Regression awareness	Ability to analyse of how changes affect system Regression awareness Risk analysis skills
Understanding customer		Understanding how changed thing will be used – and how it should be tested
Tester identity	Strong tester identity as counter force for logistic thinking	Tester as a rapid actor who can control situations
Personal strength	Courage to stop deployment if quality is not sufficient	Strength to pinpoint critical issues and focus on them
Test system control	Configuration management skills	Configuration management skills
Programming skills	Varies. Scripting skills very useful.	Skill to do rapid test system tailoring

Essential principles for supporting raised speed with testing include these:

- Understanding what is done. Make testers understand the goal of development and the customer's needs so they can prioritise and focus their actions and make compromises elsewhere when speed is critical.
- Don't go for speed by compromising quality. Learn to do things properly first, then add manageable speed. That means that that only add speed, if your testing can handle it, and your testing can make the increased speed manageable.
- Have a solid testing approach to keep the platform in good shape so direction changes can be made rapidly.
- Use robust test automation that doesn't break workflows.
- Use intelligent manual testing, because automation never notices everything.
- Testing must be able to show differences with previous release to the customer and do comparisons.
- Delivery workflows must be able to be stopped if testing tells that quality is not sufficient.
- Have everything that done and tested under strict configuration control and version control.

Change-competence snippet 47	Fast product development
Change caused by -> enables	Rapid market entry, reactivity -> timing for actions, customer satisfaction
Competence implications (re: quality and testing)	Understanding the product development paradigm #U Quality advocacy #A Risk thinking #U Customer-centredness #O #U #A Process development #A (solid testing and rigour in doing it) UX testing for feature development #O #U #A Comparison testing #A Configuration management #A Rigour in collaborative keeping the platform robust and tolerating change #A Dependability #O #A Independent problem solving capability #A Changing company-level competence profile #O #U #A

Links with	<- Relation to change -> Innovation in product development -> Towards continuous delivery -> Lean -> Business understanding for all -> Experimentation culture <- Timing and rhythm

5.10.6 Modern risk management

In traditional processes, risk management has consisted of risk analysis in the beginning of a project and monitoring of risks during the project in everyday work and reporting reviewing the risk situation in process reviews.

Modern processes see some changes to that. Agile processes do not necessarily have a big up-front risk analysis, but risks are assessed along the way. The incremental way of development keeps the scope of the system manageable and thus risks are easier to identify informally and it is easier to make decisions for controlling them. Risk-based testing prioritizes features for testing partly based on their associated risk. Tracing allows for monitoring how the risks have been covered. There is a growing understanding that the customer's risks need to be assessed more. Thus, a risk analysis for the customer's business can be part of a project's activities. As testers are more involved in all of team's activities, they also have a more direct role in discussing risks.

Change-competence snippet 48	Modern risk management
Change caused by -> enables	Incremental development -> better understanding of risk by learning
Competence implications (re: quality and testing)	Business understanding #U Understanding the customer's business and needs #O #U Risk thinking #U Product risk analysis #A Customer's risk analysis #A Safety management #U #A (on safety-critical domains) Understanding information security risks #O #U Security assessment and testing #A Cost-benefit thinking in selecting quality practices #O #U #A Dependability #O #A

Links with	<- Agility and flexibility <- Business understanding for all <- Explosion of important quality attributes <- Information security and privacy <- Pervasive communication

5.10.7 The two-edged sword of craftsmanship

Craftsmanship is not a change as such, but as a recurring concept in discussions it deserves notice here.

Craftsmanship is a term that came into relevantly wide use in the first decade of this century. Traditionally in the research of work, craftsmanship had been seen as the original form of design, which was not seen as sufficient in the modern industrial age (Engeström, 2006). In the craftsmanship, tacit knowledge is the dominant knowledge type and the worker is also the designer. That was the case in ICT before the process era and creation of more specialist roles and practices. Craftsmen can produce great products as the result of their personal work. An industrial violin builder would assemble parts designed by others, check that seams are glued ok and that dimensions are within the tolerances and package the violin. A craftsman would design the parts herself, test the violin and adjust it until the sound is just right, with no regard for formal tolerances in dimensions or pitch. That is perfect, if the business idea is to build perfect violins.

However, the knowledge used is based on tradition and experience, and thus is not suited for innovation and even good collaboration. Craftsmen transfer their craft in tacit mode to apprentices, which is not often practicable. Craftsmen do not externalise their work in plans and reports, which was what the process era expected. But Agile was a reaction to that era and brought along it again the idea or craftsmanship and emphasis on good, clean, robust code that creates a solid backbone on top of which to develop and change the application.

That can lead to the dangers of craftsmanship, where code is developed for its sake and where focus in on the internals instead of customer value and delivery. In an industrial book criticising Extreme Programming (Stephens & Rosenberg, 2003) present a view how than can lead to endless refactoring of the codebase. From the point of view of quality, that is clearly a case of part-optimisation. Code is being optimised at the cost of the overall system and its development and delivery. Perfectness in one area is the enemy of overall quality. Working at a wrong abstraction level can destroy the others. This is something that the Finnish industry, often excellent in the engineering level, collectively needs to learn. The idea in business is to create works of engineering, but products.

Scrum as the development framework was an improvement, as it more than XP exposed the developers to the world of product management (in the form of product ownership) and customers, but there still was the code-centredness and lack of design and most importantly the heterogeneous teams. Such teams where not ideal, but in practice they were common, as there was a need to get all team members productive from day one and there was no need for designers (either architects or user interaction designers), as there was no such planning – just rapid implementations of the stories that found themselves at the top of the backlog.

Craftsmanship can be a very valuable mindset, when one is programming or designing tests, but as something that encompasses the whole identity or role, it is seriously lacking. There are other mindsets that are as important and finding a dynamism and balance between those is essential. The others, as seen by the author, include:

- Product engineer. The idea is to deliver the software and thus the developers and testers need to have a mindset that continuously focuses on that. Even if code would ideally need refactoring, the delivery can be more important. Quality needs to be balanced based on the big picture.
- Team player. Craftsmen are bad team players. They immerse themselves on their work, don't explain their ideas in other ways than in their produce. Modern team players need to be quite the opposite.
- Mediator. All development work is trying to understand the world of the customers and turn it into a new manifestation of ideas that expresses that world in new forms. That transformation requires mediation of the ideas to oneself and to the others. Developers can do that, and testers too.

Also, the craftsmanship can be expanded. As user experience is a very critical success factor, there is need for craftsmanship in that area (Lindell, 2014).

5.10.8 Changes and implied competences for this section visualised

A visualised summary of the change-competence snippets in this section are visualised in Figure 56.

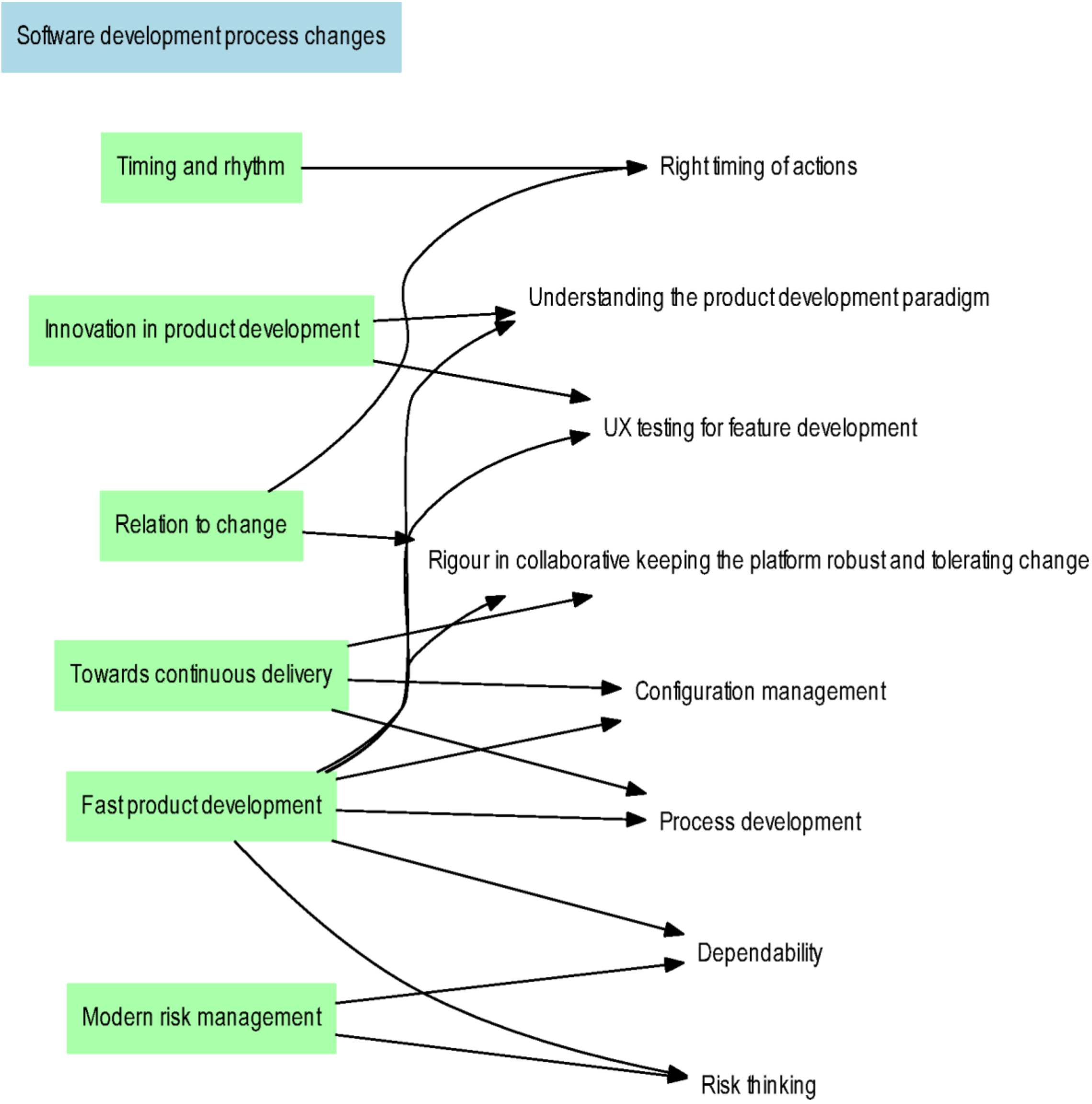

Software development process changes
Timing and rhythm
Innovation in product development
Relation to change
Towards continuous delivery
Fast product development
Modern risk management
Right timing of actions
Understanding the product development paradigm
UX testing for feature development
Rigour in collaborative keeping the platform robust and tolerating change
Configuration management
Process development
Dependability
Risk thinking

Figure 56. Visualised summary of the change-competence snippets in this section. To make the graph fit the page with legible font size, only the competences are included that are associated by at least two changes in this section.

5.11 Evolving lifecycle models

5.11.1 SEMAT as a framework of software development activity context

We shall look into some specific models of software development, but before that we need to provide a more generic framework for software development. A new development in that area is Software Engineering Method and Theory, SEMAT (Jacobson 2012). It is an approach to find the very essential elements that are present in every software development product and to specify a language that allows expressing every type of software process in a composition of such elements and their states. The whole SEMAT system is way too complex to explain here and we will not be utilising most of it, just some very essential basic blocks.

There are three most essential elements in SEMAT: 1) Areas of concern, 2) “Alphas”, the “things to work with”, and 3) Activity spaces. Contents of those are listed in Table 30. The basic elements are connected in many ways, for which we refer the reader to Jacobson (2012) for details.

Table 30. The basic elements of SEMAT kernel (Jacobson 2012).

	Customer concern area	Solution concern area	Endeavour concern area
“Alphas” – “things to work with	Stakeholders Opportunity	Requirements Software system	Team Work Way of working
Activity spaces – “things to do”	Explore possibilities Understand stakeholder needs Ensure stakeholder satisfaction Use the system	Understand the requirements Shape the system Implement the system Test the system Deploy the system Operate the system	Prepare to do the work Coordinate activity Support the team Track progress Stop the work

The most important elements in any such framework are the highest level ones that carry a distinct meaning. In this case, the Alphas are such elements and for that reason they may provide us generic guidance about the context of operation for testing in software development. For this work, we will more freely attach essential things to the elements, in order to make it more visible what the context is like and what relation the elements might have to testing – after all, SEMAT is only a neutral framework and it is up to its users to attach to it any other concepts that are essential for researching some topic. So, let’s take a look into the SEMAT Alphas, their relation to testing and quality and any competence issues they might have (any relation of text below to the activity spaces is informal).

- Stakeholders. They are “the people, groups, or organizations who affect or are affected by a software system”. All people, including testers need to understand who they are and how the system will affect those. Often the stakeholders can be divided into customer, users and other stakeholder. Testers need to understand those, especially the users and customers and what their goals are and what their business is like – what the processes and cultures are like. Today, testers may also collaborate directly with customers in specification and test planning and also in carrying out user tests at customer’s premises or otherwise. Of course, we aim at satisfying the stakeholders’ needs and verifying that requires testing – at many phases of the development.

- Opportunity. It is “the set of circumstances that makes it appropriate to develop or change a software system”. An opportunity allows the development team to understand why the system is developed and what is important to that. In this way, it relates to the concept phase of development where we try to have a hypothesis of the system-to-be-developed and to understand what its success factors are. There we need the contextual skills of understanding users and customers but also the understanding of various kinds of system concepts and what is essential for them. A risk analysis is definitely something that belongs here, and in this case its scope is the customer’s business.
- Requirements. They present “what the software system must do to address the opportunity and satisfy the stakeholders.” In traditional sense, requirements are expressed in, for example, functional requirements and non-functional (quality) requirements, but they may also be represented as use cases that the system must fulfil, or user stories that must be able to be carried out by a user with the system. Testers can participate in the traditional requirements specification, but testing of prototypes and preliminary system versions can also provide information of new requirements and customer preferences. The Lean Startup approach (Ries, 2011) is based on this. There every new system version is an experiment and thus the testers need to be involved in planning how to gather new information from every encounter between customers and systems. We will discuss this more later.
- Software system. It is “a system made up of software, hardware, and data that provides its primary value by the execution of_the software.” This is the physical thing under development and the testers must be capable of understanding it and then testing it, at various stages of development, for all requirements.
- Team. It is “the group of people actively engaged in the development, maintenance, delivery and support of a specific_software system”. Work is done in teams and that applies to testers too, meaning that the testers must have team work skills. Also, SEMAT emphasises the lifecycle of a team and a special challenge for the tester can sometimes be to manage the team dynamics and to find her/his role in the team.
- Work. It is “activity involving mental or physical effort done in order to achieve a result.” This is what people do. Work must be initiated, prepared, started, controlled, concluded and closed. That applies to testing-related tasks too.
- Way of working. This refers to “the tailored set of practices and tools used by a team to guide and support their work.” One particular question is traditionally the software development lifecycle (for example waterfall model or an agile process) and how testing is included in it. This is something that can vary greatly, depending on the industry and the type of system under development.

This far SEMAT doesn't show us much insight into the required competencies. However, perhaps it can help us identify some further "alphas"? Perhaps the SEMAT set of alphas is not quite sufficient? After all, it is very much experience-based and aims at an executable process definition, which requires certain properties from all alphas (having states is one essential property). If some essential "alpha-like" element does not have those properties, it is not included in the alpha list and may not find a place in the whole SEMAT architecture. The author thinks that two essential elements should have nearly the same role as the alphas, with the difference that while they are always present, they do not have states. Those are:

- Time. Time is always a very critical element in all work. Products have a window of opportunity that starts at a given time and closes at another time. Projects start at some time and end at another. Process models may have a given rhythm (based on iterations). This means that understanding time, having a sense of rhythm is a very essential skill to all professionals and more so to testers than some others, because tests need to be started at the right time, must be carried out promptly so the information can be utilised, testing must often be time-boxed, so work must be planned so that it fits a given amount of hours (or minutes). We will discuss the issues related to time more in another chapter.
- The cultural / organisational environment. Software development is not done in a vacuum, but always in some environment where there are cultural and organisational elements present. This is one issue that is analysed more in other parts of this dissertation.

Change-competence snippet 49	Designing new development lifecycles
Change caused by -> enables	Rethinking software development lifecycle models -> opportunity to design tailored methods for particular needs
Competence implications (re: quality and testing)	Process development #U #A (integrating testing into any new method)
Links with	-> Working in various development lifecycles -> The next steps in software development lifecycles -> Agile software development -> Lean

5.11.2 Working in various development lifecycles

The main defining characteristic of a software development project is its lifecycle. There are various basic types in use. Table 31 shows how Rothman (2007) sees them. Note that while the set of project priorities is the same for all, the order varies

Table 31. Basic software development project lifecycle types (Rothman, 2007)

Lifecycle type	Examples of this kind of lifecycle	Strengths and necessary conditions for success	Project priorities	Prognosis for success
Serial	Waterfall, phase-gate	Manages cost risk (if management uses the phase gates) Known and agreed-upon requirements Well-understood system architecture Requirements stable over the project Project team stable over the project	1. Features set 2. Low defects 3. Time to release	Successful with feedback
Iterative	Spiral, evolutionary prototyping	Manages technical risk Ever-evolving requirements	1. Features set 2. Low defects 3. Time to release	Successful assuming the finishing parts are planned and occur
Incremental	Design to schedule, staged delivery	Manages technical risk Can absorb small requirement changes but not enough changes that affect the architecture	1. Time to release 2. Low defects 3. Features set	Successful
Iterative / incremental	Agile (such as Scrum, XP)	Manages both schedule and technical risk Difficult to do well without a collocated integrated team	1. Time to release 2. Features set 3. Low defects	Successful
Ad hoc	Code and fix		1. Time to release 2. Features set 3. Low defects	Unsuccessful

All of these require practices that are tailored to the type and varying competences. In this thesis, we will look into the agile way of executing projects, as it is currently dominant and yet still the one least understood. In contrast, the linear project lifecycle is the one mostly described in textbooks.

Change-competence snippet 50	Working in various development lifecycles
Change caused by -> enables	Different situations require different lifecycles, job market dynamism -> adaptability, effectiveness, opportunities
Competence implications (re: quality and testing)	Understanding software engineering #U Understanding product/system development #O #U Understanding the product development paradigm #O #U Understanding of needs of development #U Competences usable in various process models and contexts #A Versatile method/practice toolbox #A Understanding the relevant lifecycles #U
Links with	-> Designing new development lifecycles -> The next steps in software development lifecycles -> Agile software development -> Lean

5.11.3 The next steps in software development lifecycles

While the agile development lifecycles are now common, we can learn from the history that the situation will not remain so. Lifecycle models come and go while the present ones are usually thought as nearly the ultimate ones, when a decade goes by they will be seen as relics and something to be avoided. Kennaley (2010) analysed the history of software engineering methods and showed how the current ones are a hybrid of old ones – with of course some new novel ideas – and there is no reason to think that this evolution of methods would ever stop. Figure 57 presents some elements of the evolution, based on Kennaley's work. It includes those lifecycle methods that are the most familiar and relevant in the Finnish context. The "founding" years are mostly from Kennaley.

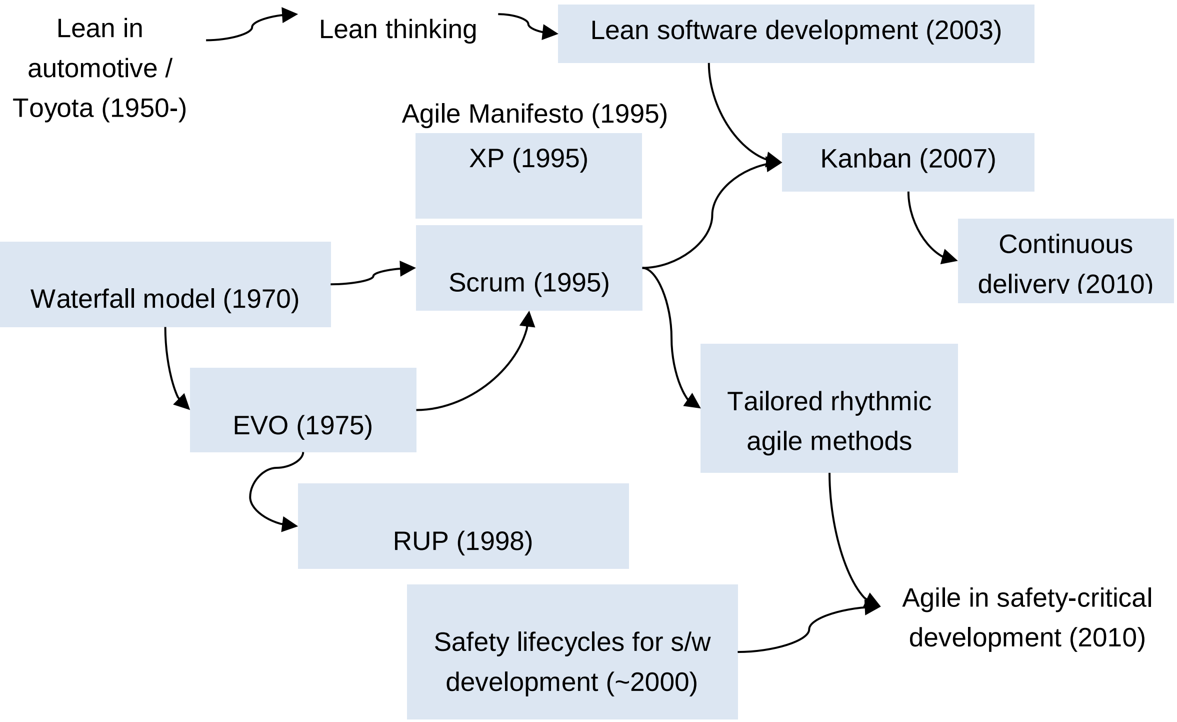

Lean in automotive / Toyota (1950-)
Lean thinking
Lean software development (2003)
Agile Manifesto (1995)
XP (1995)
Kanban (2007)
Continuous deliverv (2010)
Waterfall model (1970)
Scrum (1995)
EVO (1975)
Tailored rhythmic agile methods
RUP (1998)
Safety lifecycles for s/w development (~2000)
Agile in safety-critical development (2010)

Figure 57. The evolution of software development lifecycle methods based on Kennaley (2010). The safety critical development – important for Finnish industry – are added by the author.

It is very risky to state anything about the future of the lifecycle methods except that there will be such! What that means for the culture of testing is that we should not be attached to current methods, but “see through them”, add necessary things to them as needed, be ready to take (make!) the next steps in evolution.

In companies, one should always remember that practices should be built upon the current context and the actual needs and possibilities. Copied processes should never be expected to be the best ones for a company.

Change-competence snippet 51	The next steps in software development lifecycles
Change caused by -> enables	Lifecycles always evolve -> new practices fulfil perceived needs
Competence implications (re: quality and testing)	Understanding the product development paradigm #U Process development #U (needs for tailoring, addition, re-planning of processes) Competences usable in various process models and contexts #A Understanding overall product lifecycles #U
Links with	-> Designing new development lifecycles -> The next steps in software development lifecycles -> Agile software development -> Lean -> Fast product development

5.11.4 Agile software development

One very essential change in software development has been the adoption of agile development lifecycle models, to replace the previously more common waterfall or V models. Lately, agile development has even been used in safety-critical development; see Vuori (2011a) for an analysis how it fits in that context). There are many project models, but usually they are based on some cyclic process flow in which new versions of the system are created periodically or a linear model that only has certain items in process. Next we analyse many of the common characteristics of the agile culture and assess the competences required to be successful in those. Competences suggested by the agile principles are listed in Table 32.

Table 32. Analysis of agile values defined in Agile Manifesto (Beck et.al, 2001).

Value	Implied competences
[We value more] individuals and interactions [than] processes and tools	Social competence Domain competences
[We value more] working software [than] comprehensive documentation	Ability to understand technical systems Ability to tolerate uncertainty Ability to understand contexts and users – in order to understand how a system should work
[We value more] customer collaboration [than] contract negotiation	Social competence – people skills

Value	Implied competences
[We value more] responding to change [than] following a plan	Ability to change Ability to tolerate uncertainty – which change will bring Ability to react to changes rapidly Viewpoint to the future, not to the work already done

The implied competences are essential from the viewpoint of testing, because the ideals before the agile age favoured stability, plans and thus were negative towards change. Now, agile makes mental allowances for more freedom and independence in testing too.

What the values mean in practice is explained in the twelve Principles behind the Agile Manifesto (Beck et.al, 2001). The practices and agile values are also explained by Cockburn, 2007). They are analysed in Table 33. In this table, we also look at the implications for testing.

Table 33. Analysis of the twelve principles of agile development.

Principle	Implied competences related to quality and testing
Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.	Understanding what is valuable – and focusing on that Delivery orientation Ability to prioritise (what has more value than something else)
Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.	Ability to change, testing must not be opposed to change Ability to tolerate uncertainty – which change will bring Ability to react to changes rapidly; testing needs to adapt quickly Viewpoint to the future, not to the work already done
Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.	Delivery orientation – testing must orient to validating that the whole system is usable periodically, not just at the end of the project Timing capability
Business people and developers [and testers] must work together daily throughout the project.	Social competence People skills
Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.	Motivations Trustworthiness (testers are the cornerstone of trust)
The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.	Social competence People skills Oral communication – not just defect report

Principle	Implied competences related to quality and testing
Working software is the primary measure of progress.	Orientation to making software work
Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.	Ability to plan work so as to not get overworked, dividing the testing tasks along the development, not pushing them to the end of project or to the end of sprints Ability to plan operations so that they can have a sustainable pace
Continuous attention to technical excellence and good design enhances agility.	Orientation to good design, quality of work – testing is a key approach to that
Simplicity–the art of maximising the amount of work not done–is essential.	Orientation to simplicity Keep test systems and practices simple
The best architectures, requirements, and designs emerge from self-organising teams.	Social competence Team skills
At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behaviour accordingly.	Self-reflection skills

Looking briefly into the findings, it seems that agile values emphasise social competences, but also some orientation issues: orientation to production of value and working systems which require reflections and understanding the customer's and other parties' world, instead on the tester's mental models of things and how they should be.

Next, we shall see how the characteristics of the agile working look from the viewpoint of competences, see Table 34.

Table 34. Analysis of practices of agile development.

Practice	Implied competences relate to quality, testing
Teamwork – testers work in teams (mostly)	Social competence People skills Strong identity – needs to be a tester among other occupations
Positive attitude to test automation	Test automation skills People skills – helping others automate the right things
Exploratory testing	Intellectuality, curiosity Ability to analyse the behaviour of a systems
Rhythm	Timing skills
Close customer collaboration	Social competence People skills

Practice	Implied competences relate to quality, testing
Informal test management inside team	People skills
Some self-organisation and self-direction inside team	Team skills Persuasion skills – selling quality and testing
Visual process monitoring (burndown or burnup charts, Kanban)	(No special)
Small batch size, fast turnout	Agility in changing tasks
Tasks and stories as the units of development	Skills of developing use cases and test conditions from stories

One common idea in all agile approaches is the reduction of work in progress (WIP). This is a practice that was borrowed from Lean. Kanban (Kniberg, 2011) has got this further than Scrum, as a reflection of the long sprints in cyclic processes that make development less agile! Small amount of work in progress mean small batches of work at all phases of development and deployment, which can have a great positive effect of the effectiveness of development work, testing, quality and economy. Reinertsen (2009) presents the effects of small batch sizes for testing as in Table 35.

Table 35. Benefits of small batch sizes (Reinertsen, 2009).

Characteristic	Process benefit	Economics benefit
Smaller changes	Less debug complexity	Cheaper debug
	More efficient debug	Cheaper debug
Fewer open bugs	More uptime	Cheaper testing
	Higher validity	Cheaper testing
	Fewer status reports	Less non-value added
Faster cycle time	Less requirements change	
Early feedback	Faster learning	Better code
	Lower cost changes	Cheaper correction

The batch size reduction in every activity is one clear key to that and it directly helps developers and testers do a better job without other. Practical skills include:

- Keeping the amount of work in progress small and manageable.
- Following a scheduling heuristic (such as first in – first out) once items are on the WIP task list and supporting the development heuristics – perhaps providing a rhythm or cadence (such as an integration rhythm or a rhythm for given special

tests) or supporting the developer with the task that needs testing attention the most.

- If there is a limit for defects in correction, the testers need to have management for found defects until there is a “pull” in the correction process. There are many strategies for this.
- If a queue seems to be reaching its limit, everyone must be willing and able to help keep the queue within limits. This is about process bottlenecks that should be removed (of course, when one bottleneck is removed, something else will be the bottleneck!).
- In order to keep the flow manageable, work should not expand. That means that any work items are sized and planned so that they do not block the process. That includes any testing tasks. Obviously, good preparation and agile readiness for tasks help here.

Agile development has received critique, see Kruchten (2011). Critique towards lacking architecture design and user-centred design are seen commonly and those are critical issues from the viewpoint of quality. Teams and professional need to learn to do the quality-related actions that are needed, no matter what a new method’s manual says about it. And lately there have been advancements on these areas and for example architecture design in agile has gained a dissertation (Eloranta, 2015), as well as user experience design, see Kuusinen (2015) and Gothelf & Seiden (2013).

This is a trend of moving away from the programmer-centric nature of Agile, which was very problematic in the early days of agile development. Testing was focused on functional testing and only on the low levels of unit and integration testing. What’s more, integration testing was often in nature just execution of unit tests in an integration environment. There whole thinking was code-centred, which obviously has its benefits too. Good, clean, robust code creates a solid backbone on top of which to develop and change the application. But that too can lead to the dangers of craftsmanship, where code is developed for its sake and where focus in on the internals instead of customer value and delivery. In an industrial book criticising Extreme Programming (Stephens & Rosenberg, 2003) present a view how than can lead to endless refactoring of the codebase. From the point of view of quality, that is clearly a case of part-optimisation. Code is being optimised at the cost of the overall system and its development and delivery. Perfectness in one area is the enemy of overall quality.

Lately, system level testing and testing of all quality factors have gained focus. In some ways, this is a normalisation of testing thinking towards what was understood to be needed before the “agile invasion”. The end result is that organisations have learned to apply all relevant test types in the development in a rich way:

- Emphasising the very first sprint to develop architectures and user interfaces and to assess them and the overall system concept, and not just letting the critical things “emerge”.
- Using exploratory testing in testing of new features at the user interface level and not just relying on low-level test automation.
- Understanding that not all increments need to produce a release to the customer, and that the ones that do, may need special quality assurance activities.
- Understanding that for example security issues are such that they need to be tackled in a different mindset than functional features.
- Developing the concept of “definition of done” to consider testing of new features in the context of the overall product.
- Integration of safety-critical development practices in the rhythmic development process.

That trend can be expected to continue. They will bring along reconsiderations of the overall development process, which may lead to either evolving the common agile methodologies or forming new methodologies. After all, all methodologies have only a limited lifespan until they are seen as old-fashioned and replaced with something new.

Yet, as there is clearly a need for “filling” the development frameworks with unique practices, strong skills are required from testing practitioners. First, the ability to understand the essential practices and to implement them in the context so that the whole is optimal, e.g. creating a good synthesis of manual and automated testing. Second, there is a need for communicating and negotiating skills and personal strengths to make the process implementations happen.

This emphasises the importance of good generic education that produces competences that can be utilised in various contexts.

Change-competence snippet 52	Agile software development
Change caused by -> enables	Deplorability, changeability, learning -> less risks
Competence implications (re: quality and testing)	Understanding the product development paradigm #U Exploratory testing for feature development #A UX testing for feature development #A Right timing of actions #A Sense of rhythm #U Communication skills #U #A Team skills #A Active, self-steered working for quality #A Configuration management #A Short work-in-progress lists #O #U #A Rigour in collaborative keeping the platform robust and tolerating change #A Role finding #A
Links with	-> Designing new development lifecycles -> The next steps in software development lifecycles -> Lean <- Agility and flexibility <- Timing and rhythm

5.11.5 Lean

Lean is an approach that is often associated with the context of agile development. Lean was originally called “Lean production”, but today “Lean” represents only a vision of good efficient ways of action. It has been applied mostly in manufacturing industries, but recently it has been coming into the software development world (see, for example, Poppendieck 2007). Lean originates from Toyota’s car manufacturing, where some new principles were identified that enabled Toyota to be fast and flexible in its car production. Lean was also identified to be inexpensive, due to low capital and quality costs, efficient (fault-free, optimised production and logistics, no learning curve in production) and clearly it would produce more faultless products that was customary at the time.

In its purest form, Lean is documented as “The Toyota Way” (The Toyota Way, Wikipedia article) and its most concise expression are its 14 Principles. Those are electronically available in Wikipedia (2016). The principles are elaborated more in, for example, a book by the same name (Liker, 2004), available in many languages, including Finnish.

But culturally, Lean can mean various things. Modig & Åhlström (2013) note that Lean can refer to these:

- Way of working.
- Philosophy.
- Approach to improvement.
- Systems thinking.
- Culture.
- Quality system.
- Way of life.
- Method.
- Production system.
- Strategy.
- Elimination of waste.
- System of understanding.
- Mindset.
- Values.
- Management system.
- Toolbox.

When Lean was introduced into software development, the elimination of waste got a lot of understanding. That was a counter-reaction to the traditions of development where lots of documents were produced with no apparent effect on getting software done. Mary and Tom Poppendieck have been the most active in transforming the principles into the software development world. They have formed the following guiding principles (condensed from Poppendieck 2007):

- Eliminate waste. Understand what your customers value. Create nothing but value. Write less code.

- Create knowledge. Create design-build teams. Maintain a culture of constant improvement. Teach problem-solving methods.

- Build quality In. Synchronise tasks from the start. Automate so that tasks become routines. Do it in a way where people can improve the process. Make it possible to change anything. Refactor and eliminate code duplication to zero.

- Defer commitment. Schedule irreversible decisions at the last responsible moment. Break dependencies between components. Develop alternative solutions.

- Optimise the whole. Focus on the entire value stream and the whole organisation, not on single processes. Deliver a complete product.

- Deliver fast. Work in small batches and short release cycles. Limit work to capacity. Put in your task queue small tasks that cannot clog the process for a long time.

- Respect people. Train team leaders/supervisors. Move responsibility and decision-making to the lowest possible level. Foster pride.

The principles are more agile than the original Toyota's principles and can be used to develop agile practices inside an agile project model. Still, Lean is not the same as agile. They have very deep differences, which is condensed in Coplien (2010) into the list in Table 36.

Table 36. Contrast between Lean and Agile (Coplien, 2010).

Lean	Agile
Thinking and doing	Doing
Inspect-plan-do	Do-inspect-plan
Feed-forward and feedback (design for change and respond to change)	Feedback (react to change)
High throughput	Low latency
Planning and responding	Reacting
Focus on process	Focus on people
Teams (working as unit)	Individuals (and interactions)
Complicated systems	Complex systems
Embrace standards	Inspect and adapt
Rework in design adds value, in making it is waste	Minimise up-front work of any kind and rework code to get quality
Bring decisions forward (Decision Structure Matrices)[27]	Defer decisions (to the last responsible moment)

The first item on the table captures a lot of the spirit of differences between agile and Lean: Lean is built on thinking, whereas the core of agile is in doing. As the "thinking" part has been seen to be lacking in many agile development projects and practices, Lean can bring the missing parts into the process.

Of the many principles of Lean, the following should be especially noted in any critical context (these are freely adapted from the principles):

[27] Note: in his presentation at Tampere University of Technology, in 2009 (Coplien, 2009), Coplien explained Lean's approach to the timing of decisions as: "Letting a decision go beyond the point where it affects other decisions causes rework, so bring decisions forward to a point where their results don't propagate".

- Leadership and management support for the critical quality (such as security or safety).
- All-encompassing quality culture.
- Respect for expert knowledge and skills on the critical areas.
- Use of analysis and analytical tools.
- Aiming at efficient standardised processes.
- Continuous process improvement (on top of the solid standard processes).
- Flexible tools that can respond to change and can be taken fast into use in a project or its phase (a new increment that needs adjustments to the tools' configuration or other preparation).
- Full understanding of what customers need – full understanding about product's use and its expected safety features.
- Welcoming defects in the development phase – they were not left for the users to find.
- Deep problem analysis and changing of activity to prevent problems in the future.
- Helping of subcontractors reach the same level and same thinking as us.
- Continuous long-term development of all areas of activities (Kaizen), not just development or verification process improvements.

All in all, Lean is not a process, but a way of thinking, shared by every member of the company and its subcontractors.

Analysis of some important Lean practices

It should be noted that implementing Lean fully requires a holistic approach. It does not contain a "silver bullet" of company transformation. There are many instances where companies have failed in Lean when they have misunderstood the approach and have tried to implement Lean in a mechanistic way, leaving out important ingredients.

Still, Lean has brought into agile culture some important practices that can be implemented individually, to produce benefits. We will look into the most important ones shortly.

a) Management of production flow. Lean uses Kanban to produce a pull process and management of amount of work in progress.

b) Visual process monitoring. Again. the Kanban boards and flags in dashboards represent this.

c) Short, readable reports that focus on critical issues. As Lean aims at sharing information between all parties effectively, it supports short and concise reports that can fit on a large printout or one computer display. This is a principle that is valuable for communicating with management – "management summary", but obviously, technical reports, specifications or risk analysis reports contain valuable information that simply requires a certain number of documentation pages.

d) Decisions on the production "floor".

Because production needs to be efficient, problems must not be passed forward to the next project phase, but instead, the process can be halted and problems solved before continuing. In a lean factory, everyone can stop production when such a need arises. Of course, as production is done in teams, not everything needs to be stopped – that would be stupid – but just a flow through a certain process. This is a needed element in modern software development. The team needs to assess the situation every day and react to problems. Regardless of the managing principle, a discussing approach is emphasised in Lean and in Agile, because it is understood that without power and respect, the teams are not efficient.

In practice, the stopping of the process can happen in various ways:

- If integration tests are noticed not to work, the situation needs to be corrected immediately and the developers should help in correcting the problem before continuing development.

- The development team and the test team should have a vote on not releasing a product, even if it meets all the mandatory requirements, if they feel that it is not (yet) fit for use. Agile processes provide good environment for this kind of discussion at the end of all increments, and good controlled way of making the necessary changes in the next increment.

- One way of stopping the process flow in agile development is to dedicate an increment to correcting errors and problems and doing maintenance tasks. If the unresolved software defects start piling up, the situation needs to be corrected and an agile process with its increments provides a good environment for that.

e) Analysis and problem solving in teams

When there are problems, such as a defect found in a very late testing phase or a new kind of defect, it needs to be analysed:

- Why did the defect get this far?

- Could it have been found in earlier testing tasks?
- What was the cause of it? Specification, lack of communication, errors in implementation or something else?
- What could be done about it? How can we change our way of acting so that the reoccurrences of this kind of defect can be minimised?

This kind of analysis should be done in every kind of development, but especially when problems are found in the validation phase. When problems are found at that phase, correcting of the problems will need repeating some of the preceding process and that takes time and resources and costs money. What is special about Lean is that instead of a single expert doing the analysis, the team does it together, which will make process improvements much easier. Tools are used in the analysis, such as cause-consequence diagrams and this tool/method-based approach should be part of the culture in every development organisation. For more analysis of Lean's approach to improvement and minimising defects, see Vuori (2010a; in Finnish).

f) Learning org**a**nisation

A lean organisation should be a learning organisation. Lean, on the other hand, is based on the idea that the whole organisation should learn from what one team has experienced. The increment-based agile process used provides a good rhythmic basis for that, very different to traditional projects, which publish their lessons learned and reusable components, etc. perhaps only after a project has been ended. In contrast, non-lean agile processes usually contain a self-reflection task at the end an increment, but that is mainly for the team to be able to adjust its behaviour in the next increments and there is no process to transfer the learned things to the rest of the organisation (not that there is anything to prevent it). So, the learning is mostly internal, adaptive and brings small, continuous improvements. That can produce good results if applied properly.

It is possible to build around an agile process functions that help teams spread their new technologies, new process improvements, new development and testing tools, etc. to the rest of the organisation. That was the process part of the equation. Just as important is Lean's understanding that excellent people are the most important asset. The best product needs the best product developers. The best processes need the best people to plan them. The best collaboration helps everyone get the best out of themselves and others.

g) Approach to risk taking in development

Agile, lean and safety-critical cultures differ in their approach to taking risks. By risks we mean issues of:

- Dealing with uncertainty.
- Accepting the failure of a development decision.
- Accepting the insufficiency of a product iteration. This includes safety and security.

The relation to risks of these three cultures is compared in Table 37. In the table, safety-critical development has been added as a comparison culture, as it is specialised to handle risks.

Table 37. Comparison of risk-taking approaches of Agile, Lean and safety-critical cultures.

Issue	Agile	Lean	Safety-critical development
Experimenting	Just try it – it is ok to fail; you can do it again	In product development, new concepts are very welcome	Find something already proven, then assess it and test it
Alternatives	Alternatives are not sought.	Create redundant alternatives	Alternatives are needed for diversity, exhausting use of alternatives at concept level
Design failures	Failing is ok – it helps us learn – just adjust at next increment	Not accepted. Carefully validate designs before implementing.	Failing is absolutely not ok, as someone might get hurt
Phase of risk taking	All development phases	Concept design. Risk taking stops at the design phase.	None
Co-operation and risks	The social process produces compromise – medium innovation risks, medium safety risks	Do the design well	Sense of responsibility is high, making people cautious. Risks are assessed in collaboration.

Overall, Lean has important values for testing and quality. First of all, it brought along the methods that allows for monitoring the development of individual features, thus enabling focus and visibility to their quality. On the broader level, Lean emphasises up-front planning and design, the idea being that the execution can be effective, when it is understood what is being done. Central to Lean is also the idea of collective learning and commitment to the team and the company.

Change-competence snippet 53	Lean
Change caused by -> enables	Need for practices in agile development, flow control -> value flow, focus
Competence implications (re: quality and testing)	Understanding the product development paradigm #U Process development #U #A (integrating testing into the flow) Short work-in-progress lists #O #U #A Helping developers in development queue #A Deployment and automation skills #A Configuration management #A
Links with	<- Agility and flexibility <- Agile software development -> Designing new development lifecycles -> The next steps in software development lifecycles

5.11.6 Changes and implied competences for this section visualised

A visualised summary of the change-competence snippets in this section are visualised in Figure 58.

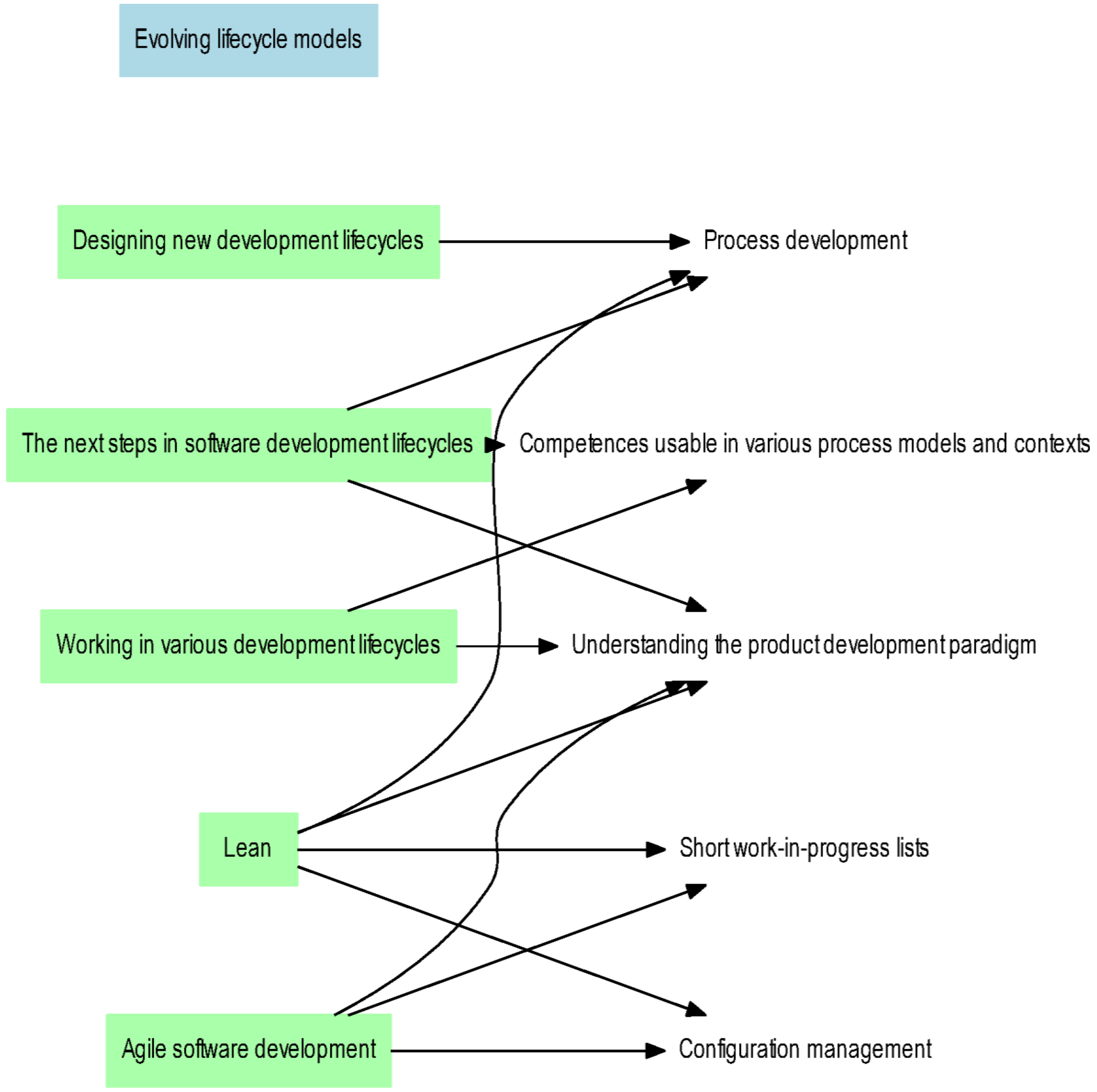

Evolving lifecycle models
Designing new development lifecycles
Process development
The next steps in software development lifecycles
Competences usable in various process models and contexts
Working in various development lifecycles
Understanding the product development paradigm
Lean
Short work-in-progress lists
Agile software development
Configuration management

Figure 58. Visualised summary of the change-competence snippets in this section. To make the graph fit the page with legible font size, only the competences are included that are associated by at least two changes in this section.

5.12 Changes in testing thinking

5.12.1 Rethinking of the goals of testing and quality assurance

If we see the world as changing, it is at this point necessary to rethink the goals of testing and quality assurance, so that we can also more freely rethink the actions that we use to reach those goals. It is essential for the rethinking to have simplicity and a suitably practical abstraction level.

On a general level, the purposes of testing have evolved during decades from finding defects to creating information about quality. They are sometimes too abstract. If we just create “information”, it leaves in real-life contexts open the question about the nature of the information and how it is used. Based on what we know about the modern companies, we could formulate that the purpose of testing is to create, with empirical means, quality-related information that supports the development of a system or any business activities and decisions around it. That means that in testing, not only information is created, but each piece of it has a purpose, it is targeted to some use, some roles in the activity system, to an operation that happens in some time span.

Recently there has been more emphasis on identifying a goal for all testing performed, so that its execution and reporting can be optimised and the resources are used optimally. This is a good thing as traditionally testing has sometimes been done just because the processes say so. The emphasis of goals was also necessary at the introduction of agile development process models, which required rethinking. For example, Scrum (Scrum Guides, 2015) does not prescribe any testing activities, but clearly they need to be added. So the question is: what should be added, when the aim is to keep the whole as lean as possible.

One view that has gained plenty of support recently and requires mentioning here is the “agile testing quadrants” by Brian Marick and thoroughly documented by Crispin & Gregory (2009). It is presented in Figure 59.

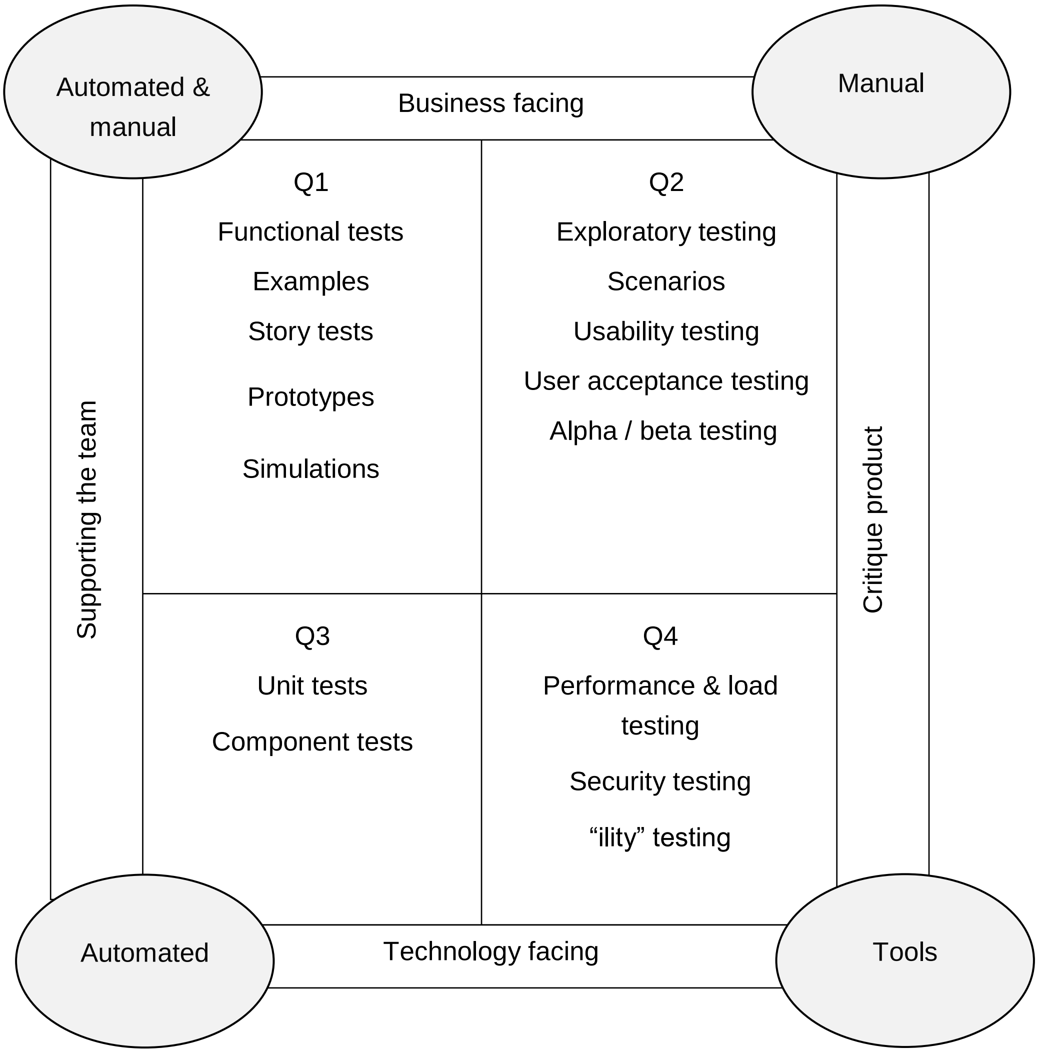


Figure 59. The (agile) testing quadrants by Brian Marick, described in Crispin & Gregory (2009).

That model is valuable in that it presents important principles:

- The development team needs support from testing.
- Critiquing the product is essential. Testing has always been seen as a balancing force in a system where other activities and roles press for a product to be released, in any quality and maturity.
- Business and technology are key elements in the activity system.
- All testing types *must* be able to be positioned in regard to their relation to business and technology.

One clear value of the quadrant model is also that the model fits on one page, making it possible to explain to various people what purpose some testing type has and what benefits it might provide. Yet, it leaves open the question why and how something would support business? Should that question perhaps be understood better or should the project teams and testers be always ready to support others in any way possible. That is not possible if the actions – the test types – are defined in advance.

Ideally, we should understand the goals of business and derive any quality related actions from those. That way we can better plan the actual ways of action so that the best support those goals. Some actions might be experimental testing, but some others might be something else – for example inspections, reviews, simulations, the analysis of information from customers and previous projects and so on. That is why we need to step up the abstraction level and try to see the needs and thus goals at different levels of activity. Then we can use that "development business understanding" to search for good ways to provide the expected value and use our competences in the best possible ways – and to develop our competences for those purposes. Table 38 shows a concise analysis of that issue.

Table 38. Levels of quality assurance in projects including quality outcomes and inputs (for systems business) – condensed visualisation.

Level	Quality outcome	Information input
Business	Successful business	Sales
	Satisfied customers	Feedback from customers
		Analysis of customer reports of defects, problems.
		Proper acceptance testing
	Satisfied users	Feedback from users
		Analysis of user reports of defects, problems.
System / product concept	Best concept selected	Prototype tests of alternative concepts provide information
	Concept implemented well	Testing provides information
Requirements	Most important requirements found	Collaboration in requirements gathering (specification)
		Prototype tests
		Use of experience
	Requirements are met	Testing provides information
	Essential risks identified	Collaboration in risk analysis
	Risks are assessed	Testing provides information
		Use of experience
Design	Design done well	Design analysis
		Architecture analysis
Implementation	Robust, flawless implementation	Testing provide information

Level	Quality outcome	Information input
	Easy to change and maintain	Assessments provide information
	Good technologies and techniques used	Participation in the selection of technologies
		Proof of concept tests

We can assess the needs of the elements of the activity system in a similar fashion in Table 39.

Table 39. Quality assurance regarding some elements of activity system including quality outcomes and inputs (for systems business) – condensed visualisation.

Elements	Quality outcome	Input
System under development	Everyone knows status	Testing produces information
	Status and trends are shared and recorded in objective way	Quality reports, dashboards / radiators make status visible and shared
	Decisions are based on reliable data	Quality metrics
	Product flaws are known and can be assessed	Maintain error database
	Information flows to and from participants	Collaboration and communication
Development project	Everyone knows status of the project	Testing provides information
	Project can be estimated	Project metrics
	Information about problems and potential problems reaches those who can act on the information	Collaboration and communication
	Risks can be managed	Monitoring of risks
	Participants learn from their actions, successes and problems	Reflection

Note that we have avoided mentioning any practical system characteristics that should be developed and “assured”, as any definition of those is context depended. There are general-purpose quality model standards, such as ISO/IEC 25010 (ISO/IEC 25010, 2011), but they are proposed to be used mostly as a checklist.

In the product development a usual concept is "problem-solution fit". Clearly, all the tasks and goals above require some competences for their realisation, some specific and some more general. The main idea here is that we should not even think that there are any general competences that are of value as such. Instead we need to think of the problem-competence fit. When there is a "testing problem", what are the competences that best fit that, bringing the most value?

Change-competence snippet 54	Rethinking the goals of testing and quality assurance
Change caused by -> enables	Better effectiveness and support for business -> better business
Competence implications (re: quality and testing)	Test planning for purpose #U #A Business understanding #O #U Understanding product/system development #O #U Understanding of needs of development #U Understanding software engineering #U Versatile method/practice toolbox #A Collaboration skills #U #A Communication skills #U #A Right timing of actions #A
Links with	-> Business understanding for all -> Designing new development lifecycles

5.12.2 Need for personal understanding of quality

The ideal used to be that product development was based on requirements that would capture the quality attributes and their desired statuses. However, it was already understood during the phase when that ideal was believed in that requirements are never sufficient. Later, the agile values bought lesser emphasis on documenting the requirements. What was documented was mainly user stories and similar, but they always leave open the quality of the product when the cases are being executed. It is now more an open-ended situation where the testers themselves need to understand quality, for example:

- Given a product concept, what makes it good? What are the things that we should look for or look at when testing? What are the things that must not fail? What are the things that should be optimised?
- What are the things that differentiate this particular product and thus need to be implemented in an excellent way?

- What do the customers value? What features and characteristic bring them the most benefit and satisfaction?
- How do the users really use the products? What things are essential for them and what are secondary?

Getting answers to those kinds of questions requires two kinds of things. First, the testers need to have an understanding about products and systems and their usage. This is largely general knowledge of “things” around us and how people in various cultures use them.

Secondly, the testers need instruments for getting the answers in any particular situation. They need to be able to do or have someone else do sufficient studies to find out about the issues, including customer surveys, user research and so on.

Those are approaches that were previously left to usability experts and product planners, but now, and in the future, everyone must have at least a small personal toolset for those.

Change-competence snippet 55	Need for personal understanding of quality
Change caused by -> enables	Less reliance of formal requirements -> opportunity to find essential characteristics
Competence implications (re: quality and testing)	Open-minded quality thinking #O #U Understanding of product, product culture, businesses and their needs #U Understanding technical systems #U Using social media and web in getting information and sharing information #O #U #A Personal competence development #O #U #A
Links with	<- Explosion of important quality attributes <- Changing working life <- Agility and flexibility <- Business understanding for all <- New thinking on defect costs during application lifecycle

5.12.3 New thinking on defect costs during application lifecycle

Commonly it has been thought that defects are most inexpensive to repair if they are found as early in the product’s development lifecycle as possible. Defects that are found during use, by customers, have been seen as very costly, due to the effort of delivering updates to the users. Recently, the updating mechanisms have improved greatly – think of over the air updates to phones and continuous deployment for web systems – and the updates are not such an obstacle that they were during the pre-Internet age, from which many cost models originate.

Embedded software is very costly to update, and it is often safety-critical software or exposing it to public networks can cause security risks, but many desktop and mobile applications are easy to update from Internet. This does not imply that the corrected versions should not have proper, and sometimes costly, testing. Still, updates can be very costly for systems that have a very large number of users – for example, updating a widely used operating system might cost millions of dollars.

This means that in many cases, serious defects can be very important to find before the initial software delivery. The main lesson here is that each situation needs to be assessed to find the real cost of defect correction, so that the testing process can be optimised, considering cost, risks and customer satisfaction.

Change-competence snippet 56	New thinking on defect costs during application lifecycle
Change caused by -> enables	Update / deployment mechanisms -> opportunities to rethink and prioritise processes
Competence implications (re: quality and testing)	Understanding overall product lifecycles #U Understanding customers #U Business understanding #O #U Understanding software engineering #U Understanding deployment #U Deployment and automation skills #A
Links with	<- Fast product development <- Towards continuous delivery <- Small inexpensive apps -> Business understanding for all

5.12.4 Changes and implied competences for this section visualised

A visualised summary of the change-competence snippets in this section are visualised in Figure 60.

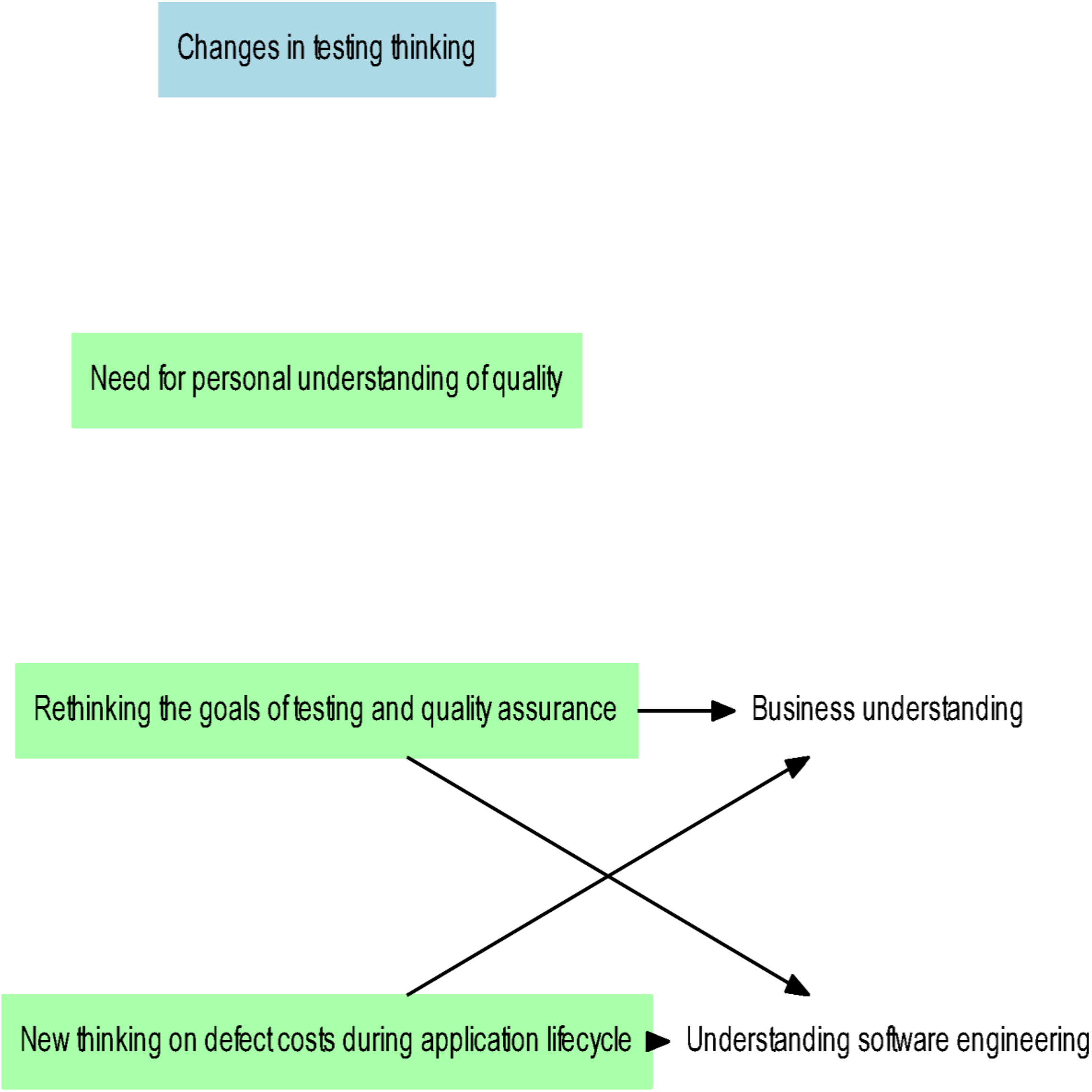

Changes in testing thinking
Need for personal understanding of quality
Rethinking the goals of testing and quality assurance
Business understanding
New thinking on defect costs during application lifecycle
Understanding software engineering

Figure 60. Visualised summary of the change-competence snippets in this section. To make the graph fit the page with legible font size, only the competences are included that are associated by at least two changes in this section.

5.13 Testing arrangements

5.13.1 Testers changing context more often

Some of our expectations about workplaces have traditionally been that a person can join a new company and a new team and gradually settle in and at some point be as beneficial member as the rest. Sometimes that can take months though. During that time, the person can get training about the company's ways, processes, practices and tools, but mostly the new worker must rely on observing how the others work and try to do similar, when participating in a task. Traditionally the process is one-way – the newcomer in introduced to the system, minimising any disturbances. At the end of the process:

- The newcomer has found an operational role in the organisation.
- She has also found a psychological role in the organisation, especially her own team.
- She may have introduced some changes in the team that soon ripple out.
- She has learned the ways of the team and expectations for her performance.
- She has learned the culture of the organisation, its assumptions, the ways of talking and doing things.
- She has noted some deficiencies in the organisation, and may or may not start pointing them out.
- Depending on her role, such discussions will or will not lead to anything.
- The most expected outcome is that she is "dissolved" into the culture.

Can we today afford that? How can companies develop if they just neutralise every newcomer to their current situation? Of course, the idea is not that. The newcomer should be taught the frames of behaviour, but in practice, the status quo, assumptions and bad patterns are the first to stick. Companies can naturally evolve, and do so, with continuous learning and by improving practices, but in lean organisations there often are not many opportunities for that.

Furthermore, companies are now seen as transition – growth, changing their products, businesses and domains and so on. Each newcomer should bring an extra ingredient that helps in the transition. So, the transformation should go both ways. The newcomer should definitely learn the logic of the context she is stepping in, but also understand where it is going to. She should understand that any organisation is lacking in some ways and she should be the part that removes some of the lack. So, the incubation process should end in a state where:

- She has noted some opportunities in the organisation to improve and in her role has started the improvements.

All in all, in small and agile companies the newcomer must by necessity be active in the process and take personal responsibility, because the time and personnel resources dictate so. The competences needed for that are various:

She should have mental orientation to the context changes and understanding that different contexts need different approaches. She should have a generic mental toolbox about various contexts to be able to do personal re-orientation. That is, she should understand sufficiently about businesses, quality factors for different kinds of products, quality assurance requirements and good practices in various contexts, product and project risks and last but not least about the typical cultures in the domain.

Taking a role in a team requires not only ability to read the work contract, but ability to read the team dynamics and existing roles. Sensemaking has already been discussed as a relevant skill when the domain of product's usage is unknown territory. In the same way, a new working context may be unknown territory and one needs to make sense of it before taking any serious action.

If and when she possesses clear added value for the team, that needs selling: making others understand what she has to offer and what benefits she might bring. Example of this are the ex-Nokians who have during a couple of years entered other companies with the implicit promise of understanding serious international software development, and good development processes and practices. Similarly, test automation experts need to be able to produce a value proposition broader than just knowing a tool.

Practical work requires the personal creation of a quality model for the product (and business) as there rarely are good descriptions or even a shared understanding about it. If there is an understanding it, may be invalid and based on situations some years ago. This depends on design / product thinking present in team; not necessarily in leadership positions, but somewhere, in someone's mind, that could form some consensus. This is where the national level issues show up in team level: a team might be geared up to do testing that supports engineering and logistic, neglecting other areas.

Simplified, the introduction of new ideas can be twofold:

- Formal, selling ideas, negotiating new practices. Formerly, this was a part of process development, implemented in projects. Today, there is mostly room for that when new projects are planned. That takes skills of presenting ideas and the benefits and other rationale behind that.
- Knowledge transfer during daily work by externalizing ideas by communicating them in discussions and showing how to do them. Pairing and for example team sessions analyzing the product are traditionally seen as valuable in this.

This is a two-way process, meaning that competence transfer and development of practices require willingness of the organisation and suitable skills to be able to learn from the newcomers. That is once again where the cultural issues come to discussion. We will not go deeper into the organisational learning of change leadership here, though.

Change-competence snippet 57	Testers changing context more often
Change caused by -> enables	Business and national dynamics -> competence transfer, new ideas
Competence implications (re: quality and testing)	Understanding overall contexts #U Understanding domains, contexts and situations #O #U Business understanding #O #U Active, self-steered working for quality #A Broad flexible competence #O #U #A Domain-agnostic competences #A Versatile method/practice toolbox #A Multi-skilledness #O #U #A Role finding #A Reflection on working styles #U #A Cultural adaptation #A Social skills #A Collaboration skills #U #A Communication skills #U #A Using social media and web in getting information and sharing information #O #U #A
Links with	<- Changing working life <- Smaller companies <- The startup phenomenon -> Quest for multi-skilledness -> Changing engineering education -> Need for personal understanding of quality

5.13.2 Better workplaces

Testing is mostly seen as a process activity, while doing it very much defines some people's job profile. Those people are the ones whose occupation is "tester" and those who in other occupation do plenty of testing. For others, testing has a smaller, more reflective role along the activities and tasks that form their mental model of their jobs. But because there is a need for people who do testing a lot and whose job profiles are defined by the testing they do, there is a need to consider, how well the testing practices meet the positive and negative characteristics of modern work?

Work research and conscious work design seem to be somewhat neglected areas in software engineering. The reasons for that may be traditions – the work is seen as office work and that has been seen as basically good ergonomically compared with work in industrial occupations and environments. Now that the work profiles are changing we need to take at least a short reflective look into the essential elements of work design, the characteristics it proposes for work profiles. All that provides yet another element of the context where testing is done and form the opportunities to use any competences. Besides that, if we wish that competent, intelligent people wish to do testing, the work design must match the expectations of modern professionals.

A common list of characteristics of good work that suits this dissertation nicely is based on the ergonomics of humans' work with machines. Testing is quite clearly that: working on a workstation, using and monitoring the system under test and using a collection of test and other tools. The standard SFS-EN 614-2 (2009) includes in its informative annex a list of "Characteristics of well-designed jobs and implications for design". It is reflected against testing in Table 40.

Table 40. Characteristics of well-designed jobs and implications for design (SFS-EN 614-2, 2009).

Characteristic	Reflections on testing
Experience and capabilities of the operator	Testing tasks and tools needs to match the skills of the testers. For example, while advanced testing techniques might provide value, they may be too difficult for the organisation's testers. Hard task must be combined with easier ones so that the whole is not too consuming. For example, days test modelling can be combined with exploratory testing and there may be manual testing of various types – some more straightforward than the others. But testing should not be considered as a "break" from for example programming.
Meaningful whole	The job must be "complete". It should not be a collection of fragmented small tests, but more an overall assessment of something – testing of a feature or a use case instead of a component, for example.
Contribution to the total work output	Testing needs to be visible so that the contributions of testing tasks are seen clearly as a part of the whole contributions. An example of this is the explicit presentation of testing in the criteria of "done" and not just invisibly including testing in the implementation of software.
Variation	There needs to be variation in the work in the cognitive demands, styles and working and so on. Mixing different types of testing and analyses is one way to achieve that.
Autonomy	Testers need freedom in choosing how to test. Pre-determined test cases do not provide much freedom, but for example exploratory testing does. Letting testers do test planning and design helps here.
Learning opportunities	There should be learning opportunities. Doing test design provides that; there should be allowances to try new techniques. Participation in different types of projects is very effective for learning.
Feedback	All workers need feedback. That is one reason testing should be made visible and also measurable. Yet, one must remember the hazards of metrics.

Characteristic	Reflections on testing
Over- and underload	In old project models testers were sometime underloaded for days when they waited for something to test and then overloaded as the new release should be tested as fast as possible. Doing testing continuously, in small batches, removes those problems.
Repetitiveness	Especially scripted regression testing can be too repetitive, but test automation and exploratory testing can efficiently remove excessive repetition from testing. Test automation should be used for that, allowing the testers to do diverse testing.
Opportunities for contact	There is still a possibility of lacking human contact for example when a tester works remotely or separated from the developers. Modern teamwork builds on collaboration. If some people are left outside the circle of collaboration, the situation needs to be corrected fast.

Looking at the principles we can see that modern diverse testing that uses both test automation and exploratory testing, and other modern habits of testing can have a positive ergonomic influence. There really are good reasons for developing competences, work profiles and tasks. They can build a positive circle of improvement that boosts work satisfaction, health and business:

- The more skills a tester has, the more there are possibilities for an ergonomically solid work profile.
- The better the work profile is ergonomically, the more there are opportunities for learning in the job.
- Ergonomically better work profiles are likely to produce better testing and better product development, which produces even more opportunities for improvements due to better financial resources and a peace of mind, which is a positive factor in all development. Improvements in workplaces are hard to design and implement in haste and fear.

Change-competence snippet 58	Better workplaces
Change caused by -> enables	Modern worked need well designed work and workplaces -> well-being, effectiveness, quality
Competence implications (re: quality and testing)	Work, process and practice design #A
Links with	<- Changing working life <- Need for new types of workers -> Gamification for engagement

5.13.3 Gamification for engagement

Recently, gamification has received plenty of interest as a way to make work more suitable for modern people. After all, if games provide great engagement for them, would not the same principles applied in "serious tasks" cause similar engagement, making users more active, thus work more in productive and creative ways and making work more interesting and even desirable for modern professional who are not always that interested in traditional office systems and professional practices. The concept is indeed new Detering et al. (2011) note that the first uses of the term were in 2008 and it was widespread in 2010.

For the purposes of this thesis, the key is that someone has to develop the workplaces and work processes for the future and the tools used. Those people have a great significance for the future. Just think for example how the designers of Scrum changed the practices in a huge number of companies and how the users of widely used testing tools need to suffer from their deficiencies.

Of course, tool designers try to apply usability principles and even user experience design principles, but perhaps some gamification principles could be useful?

Does gamification work? Hamari, Koivisto & Sarsa (2014) made a literature review of empirical studies of gamification. They note that "Most of the reviewed papers reported positive results for some of the motivational affordances of the gamification implementations studied. Only two studies found all of the tests positive" and "most of the quantitative studies concluded positive effects to exist only in part of the considered relationships between the gamification elements and studied outcomes". So, it does work and may work in the context of testing too. There seems to be no studies about gamification of testing, but Singer & Schneider (2012) write about gamification of version control. Mostly their experiment used new approaches from social media, but a leaderboard was one example of gamification. Leaderboards, that show how many commits people have made, clearly emphasise the need for frequent commits, but cause competition that is not always healthy. For example, if we were to show metrics of reported bugs, the results could be negative for the very reason that people would start "gaming" the metrics: Start reporting plenty of small bugs and not find their root cause, focus on finding bugs and not preventing them. It would also be unjust, as different testers work on different areas of the system with varying defect densities. So, this already shows some elements of bad, harmful gamification.

Kumar (2013) looked into gamification of business software and presented the elements of gamification as points, badges, leaderboards, relationships, challenges, constraints, journey, narrative, emotion, game economy. Leaderboards we already tackled. Badges are given based on points. Badges are used in discussion forum and there they are based on the growing help people give to others. That is valuable in testing. Relationships are similar. If we want people to connect outside teams and organisational borders, badges could help in that, but again might be unjust for the introverts who have a critical role in their context that does not show in trivial collaboration metrics. Challenge is something to motivate developers and testers are often encouraged to present a bug as a challenge: can the developer figure out why something happens and solve it? Here the challenge is embedded in communication and not a formal mechanism. Constraints are related to time. Development is usually in some form time boxed and making it stricter is not necessarily sensible though the time of issues being open is something to monitor. Journeys and narratives already exist some form in user scenarios and when exploratory testing is done in a role of some stereotypical user. That is clearly a type of gamification and there is a need to develop more skills for that. Emotional design is something that could be used in bug communication.

Hamari, Koivisto & Sarsa (2014) identified these elements in their survey: points, leaderboards, achievements/badges, levels, story/theme, clear goals, feedback, rewards, progress and challenge. Levels are interesting. Testing is traditionally based on levels (unit, integration, system and acceptance), which is a nice way of raising abstraction level and changing focus. But in agile development there is constant movement up and down in the levels and some people will only work on one level. Clear goals are essential. Every test session needs a goal and when that is reached, one should get similar satisfaction as in a game. Splitting the focus to features instead of builds is good for this and makes a tester feel completion and get feedback. Similarly, the work division practices in agile development help in seeing progress. The traditional ways of measuring progress with coverage and test case execution rates etc. obviously do this, but have known problems.

Games are often about exploration. That is why exploratory testing is pure gamification and needs no added-on elements (there have been attempts of those like the “tours” by Whittaker (2010)). The author sees those as external elements that just distracts from the testing mission and have no place in professional testing. Prototypes, models and simulation are great tools for gamification as one can freely “play” with them, explore all their characteristics to gain invaluable understanding. Exploratory testing where the tester simulates (in an error-making way) a stereotypic user, or personas, utilises role-play together with exploration. Personas are already a common tool in user-centred development and the author has for years recommended similar tools for functional testing.

Games are often social. Bug hunts where all project personnel gather around to find bugs and perhaps get small rewards used to be used in companies to supplement too-strict test processes and nowadays to supplement lacking testing! They can have value in getting everyone together in a playful mood to find bugs and assess the product, but they are dangerous, if used to replace proper testing.

Clearly there are many elements of games that could be used in testing processes and tools. Many of them are already used, but the gaming perspective could allow emphasising them in new ways. And yes, there are pitfalls. Nicholson (2012) presents criticism of gamification he has found in literature:

> "By putting the term "game" first, it implies that the entire activity will become an engaging experience, when, in reality, gamification typically uses only the least interesting part of a game – the scoring system. (...) "once gamification is used to provide external motivation, the user's internal motivation decreases"

In testing, the internal motivation is essential. The process and tools are just interfaces for turning that motivation into action. Further, Nicholson notes:

> "Another threat to meaningful gamification is mechanism-centered design. A trap that game designers and companies can fall into is seeing a new or interesting game mechanism and deciding to build that into the gamification. "

Nicholson reminds that a key to successful gamification is to put the user's (here: tester's, worker's) needs ahead of the needs of the organisation and to do the design of the gamified system using user-centred principles.

Gamification is sometimes criticised for being a means for making people do unethical task. Therefore, any gamification planning should be made by people with solid ethical competences.

As a summary, the relevant competences here are for the process and tool developers to know the gaming elements and to apply them in ways that work with "serious" principles of task design and that do not cause unhealthy "gaming".

Change-competence snippet 59	Gamification for engagement
Change caused by -> enables	Characteristics of games improve work -> well-being, effectiveness, quality
Competence implications (re: quality and testing)	Testing tool UX design #A Work, process and practice design #O #U #A Understanding about ethics #O #U Doing ethical assessment #A
Links with	<- Better workplaces <- Changing working life <- Need for new types of workers

5.13.4 Subcontractor competences

When a company uses a subcontractor, there are explicit or implicit requirements for them. Large companies have traditionally had some formal subcontractor criteria, which are assessed when subcontractors are selected and audited during the co-operation.

There are three levels of competence requirements:

- Company level requirements, which include quality assurance systems, company size (to ensure service availability and scaling), security arrangements and so on.
- Testing related requirements, such as testing certificates or other proof of professionalism, maturity level assessments, ability to provide some set of test types and so on.
- Personal requirements, such as CVs as proof or competence, including experience of a product development method and tools, history on the domain, and other proof of being a good choice to a team.

All that is a low level view to the issues. The author made some years ago a synthesis of what it is like to be a good subcontractor (Vuori, 2009), based on experiences as quality manager and product manager and interviews of customers. The synthesis, presented as a training slide set, pointed higher abstraction level issues, which we present here along with some reflections on competence requirements.

Table 41. Issues in subcontracting and reflections on competence.

Issue in subcontracting	Reflection on competence
Precondition: Healthy profitable business. Big enough size. Wide clientele. Cost control.	These issues are related to business risks. There should not be a risk of the subcontractor going under. Therefore, competences in the business and a suitably mature business are essential.
Subcontracting is partnership. Long-range relationship. Understanding the customer's business is important. Compatibility with the customer.	These point to a close relationship, where both parties learn together. Especially new business and technology areas need rapid learning and that is something to accept and to support. The testers are expected to learn about the business and technology and become gradually more competent and proactive. The relationship benefits from cultural similarity.
Flawless services. Keeping deadlines, Perfect reliability and confidence. Efficient, quick and productive work. Quality management system. Customer satisfaction monitoring. Customer service attitude. Continuous improvement	These are traditional criteria for services.

Issue in subcontracting	Reflection on competence
Versatile cooperation.	In general, cooperation needs to work well, but it also must be versatile and adaptive to changing situations
Fluent communication with customer.	Co-operation is all about communication, either in meetings, emails or bug reports – in everything. That requires the ability to communicate in a multicultural environment.
Master-class competence is expected. The subcontractor as an ideal, something to desire.	On demanding domains, it must be expected that the subcontractor is simply the best in business. If the customer is a market leader, why would it get work from somebody who is less proficient? The subcontractor is seen as an ideal, something to desire. The subcontractor, as best, is something that boosts the customer by showing how things are really getting done in the best possible manner.
Vision and will.	The subcontractor should have a vision of its services and competences and a will to develop them.
Comprehensive service ability.	Subcontracting is a service and every tester must be able to serve the customer well. That does not mean doing it blindly. The testers must notice when the customer is about to shoot itself in the foot and must make sure that the customer understands what could follow from for example some decisions regarding testing.
Constructive criticism to customer.	Critiquing the customer is possible when relations are good. The customer must not be allowed to “shoot itself in the foot”.
Openness and honesty.	This is about being open with the subcontractor’s problems – before they become a problem to the customer.
Information security.	Absolutely essential in product development.
Competent and motivated staff.	In the agile world, staff is everything.
Resource flexibility Know-how flexibility – learning ability.	Flexibility is needed in two ways: availability of people and how the people adapt for example to new technology, how fast they can learn to work with it.
Ability to solve problems independently.	This is important with testing technology. There will be problems and they need to be fixed or worked around, even if a contract says nothing about it.

For this dissertation, the obvious question is, how are the issues changing? The changes relate to all elements of the co-operation and may include a very wide range of issues already noted in this work.

Change-competence snippet 60	Subcontractor competences
Change caused by -> enables	More business-critical collaboration -> competences promote effectiveness, efficiency, mutual development
Competence implications (re: quality and testing)	Service design #A Customer-centredness #O #U #A Understanding the customer's business and needs #U Entrepreneurial competencies #A Reputation management #A Organisational competence development #U #A Competence development focused on business needs #U #A Competences usable in various process models and contexts #A Understanding of possibilities and alternatives in testing #U Platform-agnostic skills #A Collaboration skills #U #A Communication skills #U #A Dependability #A
Links with	-> Business understanding for all -> Fast product development -> Networked communication -> Testing service competences

5.13.5 Testing service competences

At the turn of the century there was a big trend of offering testing as service. Besides offshoring “traditional” testing types, many specialised services become available. Many practitioners offered services in a way that mirrored the internal testing processes found in the clients’ operations. The competences emphasised were core testing competences. Service competences were and are as important though. Recently, context-related understanding has risen in importance, bringing a richer palette of competences to be required and offered. Moreover, while there are no public statistics for this, companies do consider “reshoring” offshored services in search for better collaboration and quality – amplified by the lower cost differences in the old offshoring countries. And this dissertation is helping in that when it analyses the competences needed for collaboration.

Of course, outsourcing and thus services offered by testing houses and consultants are important in the areas where companies are not always expected to have personnel capable of professional testing, although perhaps having good understanding of the issues. Those areas include security, performance and UX testing.

Change-competence snippet 61	Testing service competences
Change caused by -> enables	Outsourcing essential -> reshoring more probable when service offerings develop
Competence implications (re: quality and testing)	Understanding the customer's business and needs #O #U #A Service skills at all levels of the organisation #A Platform-agnostic skills #A Entrepreneurial competencies #A Understanding overall product lifecycles #U Customer-centredness #O #U #A Independent problem solving capability #A Special testing services #A (security, performance, UX) Versatile method/practice toolbox #A Understanding of possibilities and alternatives in testing #U Competences usable in various process models and contexts #A Competence development focused on business needs #U #A Discipline #O #A
Links with	<- Subcontractor competences <- Explosion of important quality attributes -> Business understanding for all -> Fast product development -> Networked communication

5.13.6 Crowd testing – or testing in the human cloud

A very different phenomenon is the idea that what companies need is not more automation, but more exploratory testing. Good exploratory testing requires good testers, whom the companies may not wish to employ due to because they are perceived to be needed only for a short time. One way to solve this has traditionally been to use testing houses to do testing or to hire an in-house tester from a testing house. That has been noted to be difficult for small projects, and using freelancers is a solution that is used on many domains in similar situations. Perhaps, in the age of the internet, there could be a global network of freelancers, from which expert testers could be found. For example, Applause (2015), more known by its former name uTest, is an organization designed for that. It provides many kinds of testing using a global network of testers. Essential characteristics of the system include:

- Any tester can join the network.
- There is a growth path, starting with small assignments in a team, and once experience and good feedback are gained, a tester can lead assignments and a team of other testers.
- The testers buy and manage their personal tools themselves (computers, mobile devices, testing tools), but the network provides only the collaboration infrastructure.

So this is clearly a new type of freelancer entrepreneurship. It will provide testing services in geographical areas where traditional testing houses are not available or where their services are too expensive. Because of that, it provides new opportunities in the same areas, but also gives market opportunities to any tester that cannot find employment in her local area. The competences related to this include:

- Entrepreneurial competencies. Personal budget management and management of the tester's own infrastructure.
- Responsibility of the tester's own reputation in the global community.
- Responsibility of the tester's own competence development to meet any new needs of the clientele and the personal career path in the community (from tester to test manager and project manager).

Any crowdsourcing has ample opportunities for unethical business. It is common to see for example design-related crowdsourcing, where many designers participate in designing for example a logo, but only the winner gets paid for her work. Similarly, it is possible to provide testing tasks where many testers participate in the testing. but only some of them get paid for their work. Crowdsourcing can also move some costs that traditional arrangements would have the employer pay, to the employees, such as computers, communications cost, purchase of testing tools, working spaces and so on. To manage temptation for exploitation, the planners of any crowdsourcing activity need to have solid ethics and the ability to assess the business model from the perspective of ethics.

Change-competence snippet 62	Crowd testing
Change caused by -> enables	Opportunities for employment, micro-services -> flexibility for some cases
Competence implications (re: quality and testing)	Entrepreneurial competencies #A Reputation management #A Personal competence development #A Understanding about ethics #O #U Doing ethical assessment #A
Links with	<- Subcontractor competences <- Testing service competences -> Fast product development -> Networked communication

5.13.7 Changes and implied competences for this section visualised

A visualised summary of the change-competence snippets in this section are visualised in Figure 61.

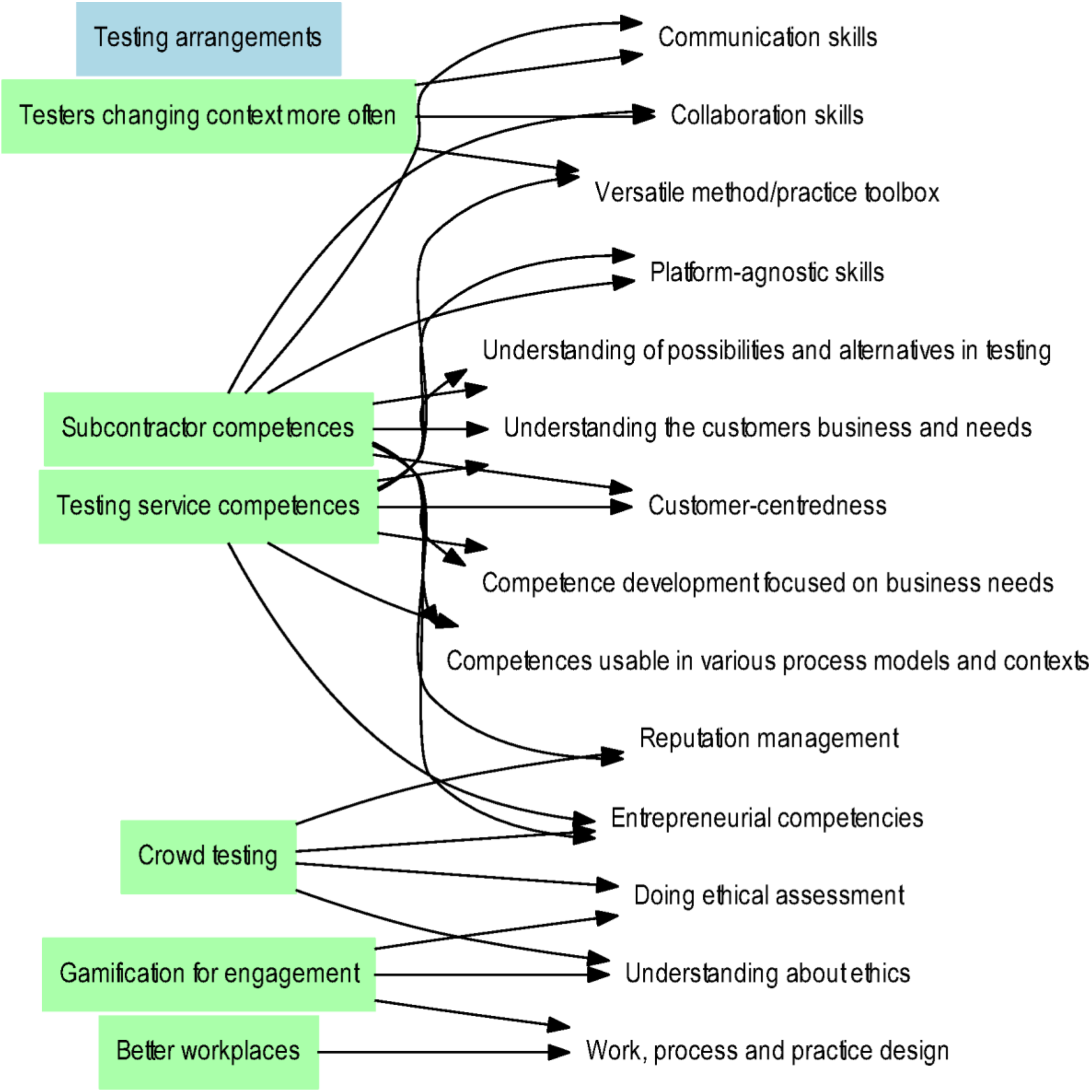

Testing arrangements
Testers changing context more often
Subcontractor competences
Testing service competences
Crowd testing
Gamification for engagement
Better workplaces
Communication skills
Collaboration skills
Versatile method/practice toolbox
Platform-agnostic skills
Understanding of possibilities and alternatives in testing
Understanding the customers business and needs
Customer-centredness
Competence development focused on business needs
Competences usable in various process models and contexts
Reputation management
Entrepreneurial competencies
Doing ethical assessment
Understanding about ethics
Work, process and practice design

Figure 61. Visualised summary of the change-competence snippets in this section. To make the graph fit the page with legible font size, only the competences are included that are associated by at least two changes in this section.

5.14 Is it all about changes?

People are often so focused on changes that they fail to see that most things remain the same. We are blind to the things that are cultural, so common that they need not be even mentioned. But new people entering the cultures and contexts need the default skills. For example, Finnish industry noted some years ago that basic training for factory foremen had been neglected and it was hard to find competent people for those tasks. Similarly, some years ago programming was thought to be something that “someone else” does and mental focus was on other issues. There forces will get a counter force, however, but such fluctuation is not the best way of doing things nationally.

There are many reasons for that. First, the new phenomena are so tempting that they mask everything else. Second, proponents of new things tend to rename old practices cutting the link to the personal and collective knowledge about things. Third, there is ignorance.

So, we need to remember that these kinds of things remain pretty much the same:

- Systems and products are still produced in projects and people need to understand project work.
- How things are tested and otherwise validated, needs planning.
- Even though teamwork and multi-skilledness is emphasised hard expertise is needed more than ever.
- In spite of new organisational practices, companies will mostly be like before.
- Testing requires the hard core skills of test case design even if exploratory testers would often not use them formally. Good test cases are the essence of test automation. With bad test cases, it is useless.
- There still are test levels and processes that look into the product at various abstraction levels and states.
- Testing by humans is dearly needed, because test automation is not able to see anything besides what it has been programmed to check.
- Even though the bulk of testing is done inside a team, bug reporting and storing and sharing bug information is valuable.
- While user experience is emphasised, the old-fashioned ergonomics are still critical in product design.

Seeing the invisible and seeing the stable behind the changes are skills themselves.

5.15 Reflection on ranking the changes

Of course, the changes that were analysed are not all as important. Some must be more important than other. But some of them have a direct impact on humans, organisations and the nation whereas some have an indirect effect. That means that simple ranking of them could do harm to our understanding – while providing tables with nice numbers that look good. All in all, we live in a system of systems where there are plenty on known and unknown interaction.

The effect any of the changes have is also dependent on the choices we make. We may “go with the flow” with some changes or utilise some of them fully. For example, there are signs that robotics and industrial Internet are areas that Finland could develop a leading position in, and such changes change the situation immediately – more in some domains than some others.

Furthermore, there does not seem to be any reliable research on this in Finland, so it is all guesswork. But for reflection we can take a look into the global situation. The World Economic Forum was already mentioned in Chapter 2 for its report on human capital. They have also published a report, where they report results of a global survey about the future of jobs (World Economic Forum, 2015b). The survey was participated by human resources officers and other senior talent and strategy executives of "leading global employers, representing more than 13 million employees across 9 broad industry sectors in 15 major developed and emerging economies and regional economic areas". 371 individual companies responded to the survey. Finland was not mentioned in the report[28]. Indeed, Finland does not have leading global employers! This dissertation sees Finland as a special case especially in the work related to high technology, so this survey will reflect more the situation of our customers in the global economic system, which should have direct impact of our product and system development – and also global co-operation.

Overall, the survey found the demographic and socio-economic drivers to rank as shown in the list below:

- Changing nature of work, flexible work: 44% rated as top trend
- Middle class in emerging markets: 23 %
- Climate change, natural resources: 23 %
- Geopolitical volatility: 21 %
- Consumer ethics, privacy issues: 16 %
- Longevity, ageing societies: 14 %
- Young demographics in emerging markets: 13 %
- Women's economic power, aspirations: 12 %
- Rapid urbanization: 8 %

And the technological changes for all respondents:

- Mobile internet, cloud technology: 34 % rated as top trend
- Processing power, Big Data: 26 %
- New energy supplies and technologies: 22 %
- Internet of Things: 14 %
- Sharing economy, crowdsourcing: 12 %
- Robotics, autonomous transport: 9 %

[28] The regions in the survey were ASEAN, Australia, China, India, Japan, France, Germany, Italy, Turkey, United Kingdom, Gulf Cooperation Council. South Africa, Brazil, Mexico, United States.

- Artificial intelligence: 7 %
- Adv. manufacturing, 3D printing: 6 %
- Adv. materials, biotechnology: 6 %

We are of course interested in how the ICT industry sees things. The ranking for that industry is as follows:

- Mobile internet, cloud technology: 69 % of respondents rated as top trend
- Processing power, Big Data: 44 %
- Changing nature of work, flexible work: 36 %
- Internet of Things: 33 %
- Consumer ethics, privacy issues: 31 %
- New energy supplies and technologies: 17 %
- Sharing economy, crowdsourcing: 11 %
- Middle class in emerging markets: 8 %
- Climate change, natural resources: 8 %
- Geopolitical volatility: 8 %
- Longevity, ageing societies: 8 %
- Rapid urbanization: 6 %
- Adv. manufacturing, 3D printing: 6 %
- Artificial intelligence: 6 %
- Young demographics in emerging markets: 3 %
- Women's economic power, aspirations: 3 %
- Robotics, autonomous transport: 0 %
- Adv. materials, biotechnology: 0 %

The global view is certainly interesting and makes one reflect on the areas much hyped in the Finnish media. For example, currently the media is full of stories of robotisation and how it will change everything. It certainly may do that, if we make it so, as a strategic choice. For our benefit, it is good to see more clearly the changes that really matter. But of course, there is also the possibility of making wrong decisions based of scenarios that will not happen. That is the nature of future.

One aspect that links many issues together is the “Fourth Industrial Revolution”. The World Economic Forum report summarises it:

> “Today, we are at the beginning of a Fourth Industrial Revolution. Developments in genetics, artificial intelligence, robotics, nanotechnology, 3D printing and biotechnology, to name just a few, are all building on and amplifying one another. This will lay the foundation for a revolution more comprehensive and all-encompassing than anything we have ever seen. Smart systems—homes, factories, farms, grids or cities—will help tackle problems ranging from supply chain

> management to climate change. The rise of the sharing economy will allow people to monetize everything from their empty house to their car.

We have analysed some concrete aspects of the revolution, but revolutions will bring with them new phenomena that cannot yet be seen. We just need to deal with what we know and be prepared for others. Parts of the preparation are taking care of high competence that can easily be adapted for new needs. And for the easy adaptation, system elements are needed that are adaptable, flexible, modular, and capable of being implemented and executed fast.

Another view to the ranking would be practical. In which way do the changes relate to the goals of business, such as innovativeness, effectiveness, agility and speed? If a systemic view is assumed, just about all the changes are related to all or those in varying level. The top level changes affect the infrastructure that enable all those qualities and even the changes that seem to affect for example mainly effectiveness, by doing that make more room for innovativeness, if applied correctly. And the correct application of things is the most critical issue. And then there are the absolutely critical changes, like the rise of importance of information security that need a strong response, but even those are just one element in the overall picture.

5.16 Summary at this point

5.16.1 Most important changes

Now it is time to take a concrete look into the findings. First, in Table 42 shows the changes that have the most number of associated competences. That means that they are very competence-intensive from the viewpoint of quality and testing, and the competences they require should be paid attention to. Another attribute is the number of links to other changes a change has. That describes how a change is embedded in a network or changes and either drives or is driven by other changes. That a change drives other changes is essential here and therefore we only count those relations. The importance results from those, independent factors and because we have no idea of their relative importance, we just add the counts of competences and "forward" links together to obtain an importance value for a change.

Table 42. Changes, their number of competences and forward links, sorted by importance (sum of competences and forward links).

ID	Change	Competences	Forw. links	Importance
1	Digitalisation	26	13	39
9	Need for new types of workers	13	16	29
17	The rise of the game industry	17	8	25
16	The startup phenomenon	13	11	24
43	Innovation in product development	18	6	24
47	Fast product development	12	12	24
24	Experimentation culture	13	7	20
33	Industrial Internet	11	9	20
8	Changing working life	12	7	19
18	Finnish style challenged	10	9	19
29	Business understanding for all	5	14	19
42	Testing of intelligent systems	14	5	19
52	Agile software development	12	7	19
5	Information security and privacy	9	9	18
15	Platform economy and API economy	11	7	18
57	Testers changing context more often	15	3	18
60	Subcontractor competences	13	4	17
61	Testing service competences	13	4	17
2	Responding to change	9	7	16
6	Emphasis on real competence	7	9	16
10	Changing engineering education	8	8	16
14	From products to services	10	6	16
25	Agility and flexibility	7	9	16
37	Multi-device systems with new interaction styles	14	2	16
48	Modern risk management	10	6	16
19	Competences focused on business type	11	4	15
20	Machine industry turning into software industry	10	5	15
53	Lean	6	9	15
54	Rethinking the goals of testing and quality assurance	9	6	15
12	Smaller companies	8	6	14
41	New technology products	10	4	14
26	Faster decision making	11	2	13
13	New external operating environment	6	6	12
21	Effective work in small, smart companies	8	4	12
23	Networked communication	5	7	12
34	Big Data	8	4	12
39	The changing requirements of technical software systems	9	3	12
3	Living with contradictions	7	4	11
28	Quest for multi-skilledness	3	8	11
50	Working in various development lifecycles	7	4	11
51	The next steps in software development lifecycles	4	7	11

ID	Change	Competences	Forw. links	Importance
7	Changing Finland	2	8	10
40	Small inexpensive apps	5	5	10
44	Relation to change	6	4	10
55	Need for personal understanding of quality	5	5	10
11	Cultural competences emphasised	3	6	9
35	Cloud testing	5	4	9
46	Towards continuous delivery	6	2	8
56	New thinking on defect costs during application lifecycle	6	2	8
4	Pervasive communication	5	2	7
27	Flexibility over maturity	5	2	7
30	Scaling of competences	7	0	7
38	Explosion of important quality attributes	5	2	7
45	Timing and rhythm	2	5	7
49	Designing new development lifecycles	1	6	7
62	Crowd testing	5	2	7
32	Integrated QA	5	1	6
31	Testing in every process	4	1	5
36	Virtualisation	3	2	5
59	Gamification for engagement	4	1	5
22	Testers in development teams	4	0	4
58	Better workplaces	1	3	4
	Average	8	6	14

The list (and similar later) should be read as only illustrative due to the inaccuracy of the analysis. It will give information about the relative relevance of the changes, but the details in the order of the list and the differences in the numbers should be neglected. Indeed, it might even make sense to present the list in a word cloud type of presentation, see Figure 62.

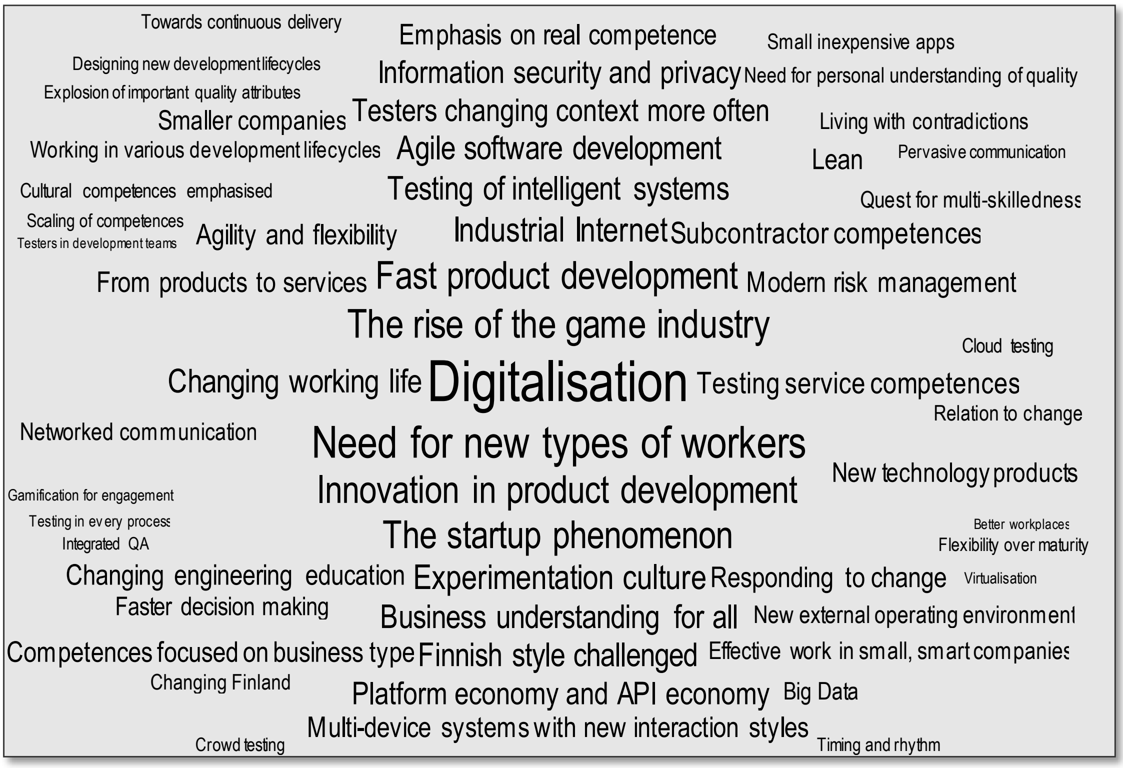

Towards continuous delivery
Emphasis on real competence
Small inexpensive apps
Designing new development lifecycles
Information security and privacy
Need for personal understanding of quality
Explosion of important quality attributes
Smaller companies
Testers changing context more often
Living with contradictions
Working in various development lifecycles
Agile software development
Lean
Pervasive communication
Cultural competences emphasised
Testing of intelligent systems
Quest for multi-skilledness
Scaling of competences
Testers in development teams
Agility and flexibility
Industrial Internet
Subcontractor competences
From products to services
Fast product development
Modern risk management
The rise of the game industry
Cloud testing
Changing working life
Digitalisation
Testing service competences
Relation to change
Networked communication
Need for new types of workers
New technology products
Gamification for engagement
Innovation in product development
Testing in every process
Better workplaces
Integrated QA
The startup phenomenon
Flexibility over maturity
Changing engineering education
Experimentation culture
Responding to change
Virtualisation
Faster decision making
Business understanding for all
New external operating environment
Competences focused on business type
Finnish style challenged
Effective work in small, smart companies
Changing Finland
Platform economy and API economy
Big Data
Multi-device systems with new interaction styles
Crowd testing
Timing and rhythm

Figure 62. Changes in a word cloud type of presentation.

Digitalisation is a generic driver in many areas. Game industry has lots of economic value, but seems to also have associated quality competences that should be addressed. Mostly that industry is addressed only from the business and creativity viewpoints.

Changing working life and the general need for new types of workers are generic issue, but reminds that the ideas that span all occupations of work also apply in testing – it is not an island. The dynamism of the environment is related to also the changing of contexts.

Innovation and experimentation are presented on the list as expected, as are issues of new technology and old domains making transformations.

Overall, we see at the top of the list an interesting mix of issues.

The changes are ranked related to their effect on practical product development performance factors in Appendix 5.

As the changes have been identified and presented under some layer of activities, contained in this document under respective headings, it is interesting to see how the most important changes are divided under those. That is shown in Table 43

Table 43. Rank of changes under layers of activity (sum of ranks of changes).

Sections	No. of competences	Importance
Changes in the structure of the economy	41	109
Global environment	45	107
Software development process changes	45	89
Working style in companies	40	84
Changing national working life	21	83
Testing arrangements	36	68
Evolving lifecycle models	22	63
Changes in product technology	28	62
Changes in product requirements	34	62
Changes in some businesses	26	49
Relations to competence in companies	18	48
Changes in testing thinking	18	33
Average	31	71

The dissertation is based on the idea that the higher levels of activity in the society drive the lower ones and the table demonstrates that. But changes in software development processes are also high on the list, as obviously, testing and all quality-related activities need to be integrated into the overall processes. The changes in product technologies have a lesser effect.

5.16.2 Most important competences

Now that there is a view on what changes are more relevant than the others, it is time to see what the more important competences are. In that we use the information on what competences the changes require. That relation provides a weight for the competences. We have now a real network of data which to assess. That is illustrated in Figure 63.

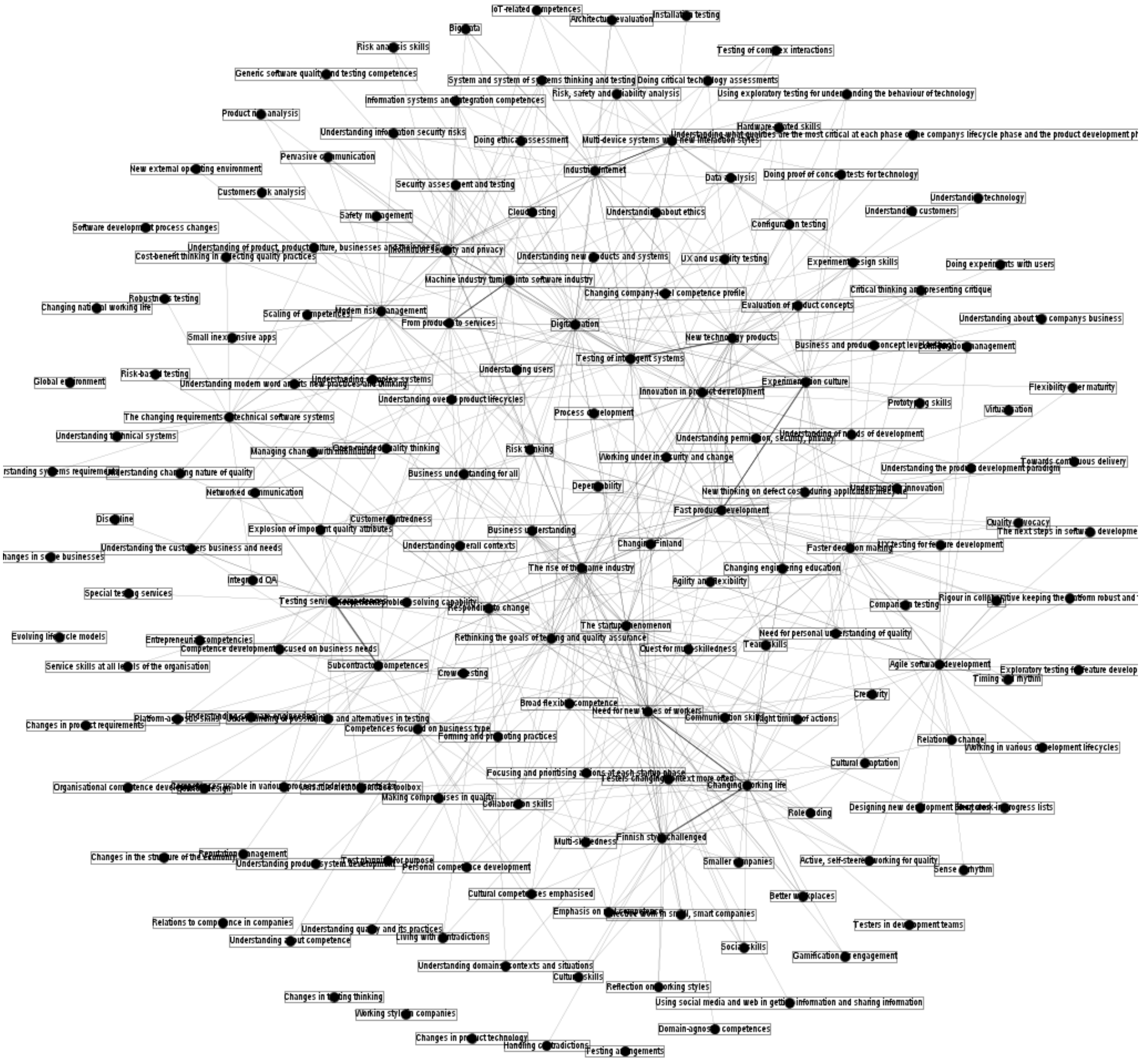

IoT-related competences
Installation testing
Architecture evaluation
Big data
Risk analysis skills
Testing of complex interactions
Generic software quality and testing competences
System and system of systems thinking and testing
Doing critical technology assessments
Risk, safety and reliability analysis
Using exploratory testing for understanding the behaviour of technology
Information systems and integration competences
Product risk analysis
Hardware-related skills
Understanding information security risks
Doing ethical assessment
Multi-device systems with new interaction styles
Pervasive communication
New external operating environment
Industrial internet
Data analysis
Doing proof of concept tests for technology
Customers risk analysis
Security assessment and testing
Understanding technology
Safety management
Cloud testing
Understanding about ethics
Understanding customers
Software development process changes
Configuration testing
Cost-benefit thinking in affecting quality practices
Understanding new products and systems
UX and usability testing
Experimentation design skills
Doing experiments with users
Machine industry turning into software industry
Critical thinking and presenting critique
Robustness testing
Changing company-level competence profile
Changing national working life
Evaluation of product concepts
Scaling of competences
Modern risk management
From products to services
Digitalisation
Understanding about the companys business
Small inexpensive apps
New technology products
Business and product concept level validation
Risk-based testing
Testing of intelligent systems
Global environment
Understanding users
Understanding modern world and its new practices and thinking
Understanding complex systems
Experimentation culture
Flexibility over maturity
Innovation in product development
Understanding overall product lifecycles
Prototyping skills
Virtualisation
The changing requirements of technical software systems
Process development
Understanding technical systems
Understanding permission, security, privacy
Understanding of needs of development
Managing change with information
Open-minded quality thinking
Risk thinking
Working under insecurity and change
Understanding the product development paradigm
Towards continuous delivery
Understanding systems requirements
Understanding changing nature of quality
Business understanding for all
Dependability
New thinking on defect costs during application lifecycle
Understanding innovation
Networked communication
Fast product development
Discipline
Customer-centredness
Quality advocacy
The next steps in software development
Explosion of important quality attributes
Business understanding
Understanding the customers business and needs
Understanding overall contexts
Changing Finland
Faster decision making
UX testing for feature development
Changes in single businesses
The rise of the game industry
Changing engineering education
Integrated QA
Agility and flexibility
Special testing services
Testing services
Problem solving capability
Responding to change
Companion testing
Rigour in collaborative keeping the platform robust and
The startup phenomenon
Evolving lifecycle models
Entrepreneurial competencies
Competence development focused on business needs
Rethinking the goals of testing and quality assurance
Need for personal understanding of quality
Quest for multiskilledness
Team skills
Service skills at all levels of the organisation
Subcontractor competences
Crowdtesting
Agile software development
Exploratory testing for feature development
Timing and rhythm
Creativity
Broad flexible competence
Need for new types of workers
Communication skills
Right timing of actions
Changes in product requirements
Platform-aware skills
Competences focused on business type
Forming and promoting practices
Relationship change
Working in various development lifecycles
Cultural adaptation
Focusing and prioritising actions at each startup phase
Testers changing to next more often
Changing working life
Organisational competence development
Making compromises in quality
Collaboration skills
Designing new development
Role-trading
Work-in-progress lists
Multi-skilledness
Finnish style challenged
Changes in the structure of the economy
Reputation management
Test planning for purpose
Understanding product system development
Personal competence development
Smaller companies
Active, self-steered working for quality
Sense of rhythm
Cultural competences emphasised
Better workplaces
Emphasis on soft competences
Effective work in small, smart companies
Relations to competence in companies
Understanding quality and its practices
Testers in development teams
Understanding about competence
Living with contradictions
Social skills
Gamification and engagement
Understanding domains, contexts and situations
Cultural skills
Reflection on working styles
Changes in testing thinking
Using social media and web in getting information and sharing information
Working styles in companies
Domain-agnostic competences
Changes in product technology
Handling contradictions
Testing arrangements

Figure 63. The network of changes and competences, for just illustrating the data, not meant to be legible. Graph was created using Gephi application.

The competences were ranked based on what changes they are associated with. A rank of a competence is a weighted sum of the ranks of the changes it is associated with. There are two weights:

- The importance value of the change, which reflects the importance of the changes the competence supports or allows us to manage.
- The weights of the competence levels. They have been assigned as 1 of #O, 3 for #U and 9 for #A, reflecting that obviously being able to do things is more important than just understanding the issues. The weights used here are the same as traditionally used in Quality Function Deployment methodology, QFD (QFD Institute, 2016), where they reflect "weak", "moderate" and "strong" influence of customer needs towards the goal.

The choice of weights is obviously just illustrative. Similar weights are often used in this kind of situations and they are based on the "hunch" of the analyser. In the following tables, the importance values have been normalised to a range of 0-100 to emphasise their relativeness.

Importance values of competences are listed in Table 44 .

Table 44. Competences, their number of mentions in the change-competence snippets, sorted by their rank (mentions weighted by the importance values of changes).

Competence	#O	#U	#A	No. of changes	Importance value
Broad flexible competence	13	13	13	13	100
Multi-skilledness	12	12	14	14	100
Business understanding	18	20	0	20	74
Collaboration skills	0	10	10	10	62
System and system of systems thinking and testing	0	9	8	9	52
Communication skills	0	10	10	10	50
Process development	2	9	10	11	48
Security assessment and testing	0	0	10	10	43
Team skills	0	1	9	10	43
Customer-centredness	7	7	7	7	39
UX and usability testing	1	1	7	7	33
Role finding	0	0	8	8	32
Information systems and integration competences	5	5	6	6	30
Experiment design skills	0	0	7	7	28
Personal competence development	3	3	7	7	26
Risk thinking	0	12	0	12	24
Cultural skills	7	7	7	7	23
Data analysis	0	5	5	5	22
Active, self-steered working for quality	0	0	7	7	22
Working under insecurity and change	6	1	3	6	21
Right timing of actions	1	1	7	7	20
Understanding overall contexts	0	10	0	10	18
Creativity	0	0	7	7	18
Evaluation of product concepts	0	0	5	5	17
Prototyping skills	0	0	5	5	17
Architecture evaluation	0	1	5	5	17
Social skills	0	0	6	6	17
Quality advocacy	1	1	7	7	17
Critical thinking and presenting critique	0	0	5	5	16
Doing ethical assessment	0	0	5	5	16
Hardware-related skills	0	5	5	5	16
Understanding information security risks	8	8	0	8	15
UX testing for feature development	3	3	4	4	15

Competence	#O	#U	#A	No. of changes	Importance value
Competence development focused on business needs	0	5	5	5	14
Competences usable in various process models and contexts	0	0	6	6	13
Doing proof of concept tests for technology	0	0	4	4	12
Risk, safety and reliability analysis	0	0	5	5	11
Configuration management	0	0	5	5	11
Business and product concept level testing	0	0	4	4	10
Making compromises in quality	3	3	3	3	9
Dependability	3	0	4	4	9
Doing critical technology assessments	0	0	3	3	8
Comparison testing	0	0	4	4	8
Understanding permission, security, privacy	5	5	0	5	7
Managing change with information	0	0	3	3	7
Understanding about ethics	5	5	0	5	7
Cultural adaptation	0	0	4	4	7
Adaptability and flexibility	0	2	4	4	7
Changing company-level competence profile	3	3	3	3	7
IoT-related competences	2	2	3	3	6
Reflection on working styles	0	3	3	3	6
Understanding the product development paradigm	1	6	0	6	6
Rigour in collaborative keeping the platform robust and tolerating change	0	0	4	4	6
Versatile method/practice toolbox	0	0	4	4	6
Independent problem solving capability	0	0	3	3	5
Platform-agnostic skills	0	0	4	4	5
Open-minded quality thinking	5	5	0	5	5
Using exploratory testing for understanding the behaviour of technology	0	0	3	3	5
Short work-in-progress lists	2	2	3	3	5
Understanding complex systems	0	4	0	4	4
Risk analysis skills	0	0	2	2	4
Understanding domains, contexts and situations	4	5	0	5	4
Using social media and web in getting information and sharing information	3	3	3	3	4
Understanding of possibilities and alternatives in testing	0	5	0	5	4
Deployment and automation skills	0	0	4	4	4
Understanding overall product lifecycles	0	5	0	5	4
Entrepreneurial competencies	0	0	3	3	4
Understanding new products and systems	2	3	0	3	3
Understanding changing nature of quality	4	4	0	4	3
Reputation management	0	0	3	3	3
Understanding innovation	0	4	0	4	3
Networking skills	0	0	3	3	3
Configuration testing	0	0	2	2	3

Competence	#O	#U	#A	No. of changes	Importance value
Safety management	0	2	2	2	3
Understanding the customer's business and needs	2	3	1	3	3
Product risk analysis	0	0	2	2	2
Customer's risk analysis	0	0	2	2	2
National competence infrastructure development	0	0	2	2	2
Forming and promoting practices	0	0	2	2	2
Focusing and prioritising actions at each startup phase	0	1	1	1	2
Understanding users	0	3	0	3	2
Domain-agnostic competences	0	0	2	2	2
Understanding customers	0	3	0	3	2
Testing of complex interactions	0	0	2	2	2
Understanding of product, product culture, businesses and their needs	0	4	0	4	2
Cost-benefit thinking in selecting quality practices	2	2	1	3	2
Understanding of needs of development	1	3	0	3	2
Understanding software engineering	0	4	0	4	2
Discipline	2	0	2	2	2
Understanding modern word and its new practices and thinking	2	0	0	2	1
Handling contradictions	0	2	0	2	1
Understanding about competence	0	3	0	3	1
Work and process design	0	0	1	1	1
Improving education for quality and testing	0	1	1	1	1
Assessment and testing of innovations and product concepts	0	0	1	1	1
Understanding quality and its practices	0	2	0	2	1
Generic software quality and testing competences	1	1	1	1	1
Understanding of domains and cultures	0	2	0	2	1
Instrumenting of systems	0	0	1	1	1
Efficient and secure data collection	0	0	1	1	1
Using analysis and reporting tools	0	0	1	1	1
Installation testing	0	0	1	1	1
Understanding technical systems	0	2	0	2	1
Risk-based testing	0	0	1	1	1
Robustness testing	0	0	1	1	1
Doing experiments with users	0	0	1	1	1
Sense of rhythm	0	2	0	2	1
Understanding product/system development	2	2	0	2	1
Exploratory testing for feature development	0	0	1	1	1
Helping developers in development queue	0	0	1	1	1
Test planning for purpose	0	1	1	1	1
Work, process and practice design	1	1	2	2	1
Service design	0	0	1	1	1

Competence	#O	#U	#A	No. of changes	Importance value
Organisational competence development	0	1	1	1	1
Service skills at all levels of the organisation	0	0	1	1	1
Special testing services	0	0	1	1	1
Scaling personal toolbox	0	0	1	1	0
Scaling resource management practices	0	0	1	1	0
Understanding cloud systems, their possibilities and problems	0	1	0	1	0
Managing of test environments in the cloud	0	0	1	1	0
Learning new testing tools	0	0	1	1	0
Understanding virtualisation	0	1	0	1	0
Virtual environment design and implementation	0	0	1	1	0
Virtual environment deployment skills	0	0	1	1	0
Understanding systems' requirements	0	1	0	1	0
Understanding technology	0	1	0	1	0
Understanding what qualities are the most critical at each phase of the company's lifecycle phase and the product development phase	0	1	0	1	0
Understanding about the company's business	0	1	0	1	0
Supporting deployment decisions with assessment and test information	0	0	1	1	0
Understanding the relevant lifecycles	0	1	0	1	0
Understanding deployment	0	1	0	1	0
Testing tool UX design	0	0	1	1	0
N = 132	142	281	360		
Average				4	11

Let's again show a word cloud type presentation of the ranked competences in Figure 64.

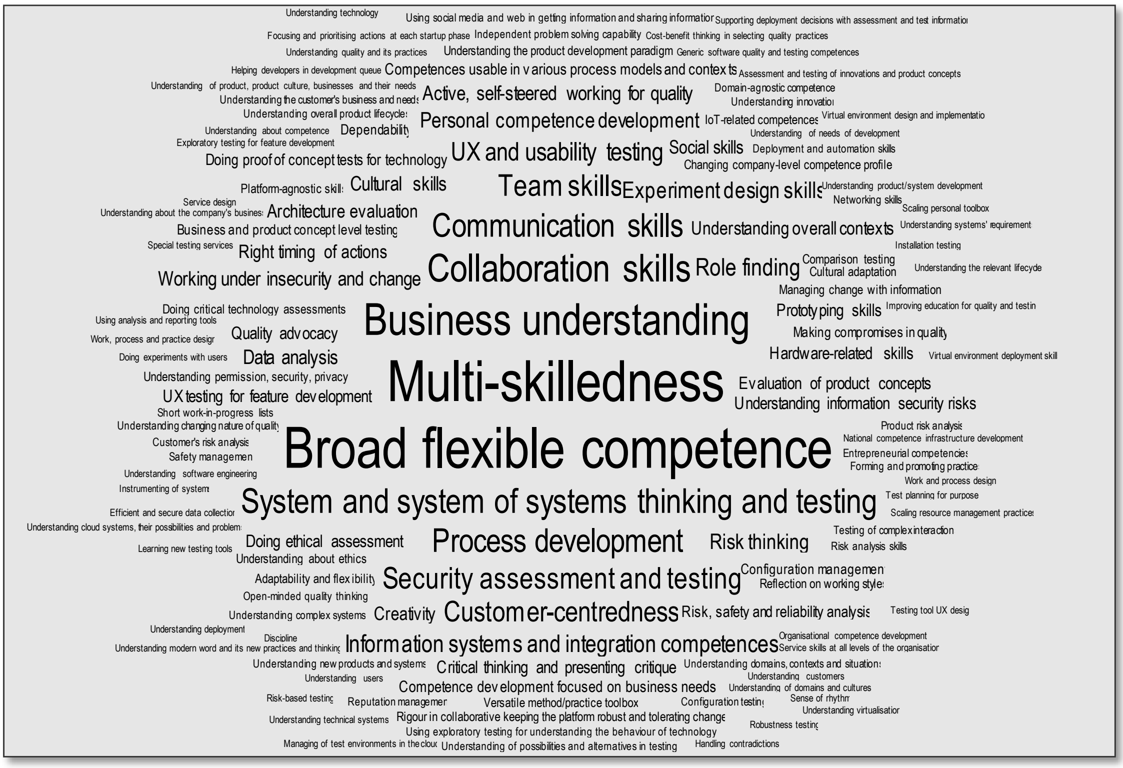


Figure 64. Ranked competences in a word cloud type of presentation.

Multi-skilledness and broad, flexible competence have the highest importance value, which reflects well the general views seen in media. They is followed by business understanding It is clearly a time for all testing to reflect on the business goals and work within the business processes. At the same time, many of the changes have risks both in technology and in business and thinking of risks is highly important. Collaboration issues are strongly present in the list. UX is perhaps surprisingly low in the list considering its importance in innovation and the generic business value of products.

5.16.3 Top competences of types #O, #U, and #A

The competence model divides the competences into three types, or levels – orientation #O, understanding #U and ability to actually do the related tasks #A. The highest ranked competences of the types are listed in Table 45, Table 46, and Table 47. Note that the importance column is calculated only for the primary type (#O, #U, or #A).

Table 45. Top 15 competences of type #O ranked by their influence on that type only.

Top 15 #O competences	#O	#U	#A	No. of changes	O importance
Business understanding	18	20	0	20	100
Broad flexible competence	13	13	13	13	70
Multi-skilledness	12	12	14	14	64
Working under insecurity and change	6	1	3	6	45
Understanding information security risks	8	8	0	8	45
Customer-centredness	7	7	7	7	44
Understanding permission, security, privacy	5	5	0	5	35
Information systems and integration competences	5	5	6	6	34
Cultural skills	7	7	7	7	31
Understanding about ethics	5	5	0	5	30
Open-minded quality thinking	5	5	0	5	23
UX testing for feature development	3	3	4	4	23
Making compromises in quality	3	3	3	3	19
Changing company-level competence profile	3	3	3	3	19
Understanding new products and systems	2	3	0	3	18

The top #O competences show a clear picture of areas that most everyone needs to be oriented towards, to have awareness. That orientation promotes sensitivity for changing situation and collaboration with the experts who properly understand the issues and can do the necessary actions. One may immediately think that directors and managers are people who should possess the orientations. That is true, but in the current team-based organisations everyone needs to have the broad views to different factors. In education and training, the competences promote a rich mental framework that surrounds the more focused areas of knowledge and skill development.

Table 46. Top 15 competences of type #U ranked by their influence on that type only.

Top 15 #U competences	#O	#U	#A	No. of changes	U importance
Business understanding	18	20	0	20	100
Broad flexible competence	13	13	13	13	67
Risk thinking	0	12	0	12	66
Multi-skilledness	12	12	14	14	61
Collaboration skills	0	10	10	10	57
Understanding overall contexts	0	10	0	10	56
System and system of systems thinking and testing	0	9	8	9	49
Communication skills	0	10	10	10	48
Understanding information security risks	8	8	0	8	43
Customer-centredness	7	7	7	7	42

Top 15 #U competences	#O	#U	#A	No. of changes	U importance
Understanding permission, security, privacy	5	5	0	5	33
Understanding the product development paradigm	1	6	0	6	32
Data analysis	0	5	5	5	32
Information systems and integration competences	5	5	6	6	32
Process development	2	9	10	11	31

The #U competences are somewhat similar, but already have some more concrete areas. It is for example not always sufficient to realise that there is a business behind every product or systems development project and that its needs need to be served by testing, but one should also understand the very logic of the business in the current domain. Similarly, understanding of information security risks needs to be at the level of understanding attack types, protection measures and what activities should be carried out in development and in testing.

Table 47. Top 15 competences of type #A ranked by their influence on that type only.

Top 15 #A competences	#O	#U	#A	No. of changes	A importance
Multi-skilledness	12	12	14	14	100
Broad flexible competence	13	13	13	13	98
Collaboration skills	0	10	10	10	83
Security assessment and testing	0	0	10	10	78
Team skills	0	1	9	10	72
Communication skills	0	10	10	10	71
UX and usability testing	1	1	7	7	70
Role finding	0	0	8	8	65
System and system of systems thinking and testing	0	9	8	9	64
Experiment design skills	0	0	7	7	63
Customer-centredness	7	7	7	7	62
Process development	2	9	10	11	55
Information systems and integration competences	5	5	6	6	54
Creativity	0	0	7	7	53
Active, self-steered working for quality	0	0	7	7	52

The last type, #A presents the areas of actual skills to do the tasks needed. It is interesting that the breath of skills is emphasised at the top of the list. It is a sign of the times and really implies that we need to make sure that people have multiple skills and promote that in all types of competence development. The times seem to be over, when a tester would present herself as competent when showing only test type and test tool -related competences in her CV.

In the middle of the list we see experiment design, UX and usability testing and also process development. Competences for those are not required for everyone. They are expert areas and for best results there should be orientation and understanding of those as such.

Note also that the list is about responding to the changes in the operational environment. That is why the "core skills" of testing are not mentioned often. It should be assumed that people who do testing in professional settings have a sufficient grasp on those.

5.16.4 Competences grouped by traditional levels

To understand the set of competences in practice, it needs some structure that relates it to organisational activities. For that, the competences were divided into various areas that represent layers of development. That was done in a bottom-up way, based on what kind of groups emerge from the list that are easy to understand, see Table 48

Table 48. Competences divided into areas representing traditional elements of product development.

Competence	No. of changes	#O	#U	#A	Importance
Contextual competences	(Comps: 18)				**170**
Business understanding	20	18	20	0	74
System and system of systems thinking and testing	9	0	9	8	52
Understanding overall contexts	10	0	10	0	18
Understanding of possibilities and alternatives in testing	5	0	5	0	4
Understanding domains, contexts and situations	5	4	5	0	4
Reputation management	3	0	0	3	3
Understanding the customer's business and needs	3	2	3	1	3
Understanding of product, product culture, businesses and their needs	4	0	4	0	2
Understanding customers	3	0	3	0	2
Understanding users	3	0	3	0	2
Focusing and prioritising actions at each startup phase	1	0	1	1	2
Understanding modern word and its new practices and thinking	2	2	0	0	1
Special testing services	1	0	0	1	1
Handling contradictions	2	0	2	0	1
Understanding of domains and cultures	2	0	2	0	1
Understanding about the company's business	1	0	1	0	0
Understanding systems' requirements	1	0	1	0	0
Understanding what qualities are the most critical at each phase of the company's lifecycle phase and the product development phase	1	0	1	0	0

Competence	No. of changes	#O	#U	#A	Importance
General elements of competences	(Comps: 22)				**361**
Multi-skilledness	14	12	12	14	100
Broad flexible competence	13	13	13	13	100
Customer-centredness	7	7	7	7	39
Active, self-steered working for quality	7	0	0	7	22
Creativity	7	0	0	7	18
Critical thinking and presenting critique	5	0	0	5	16
Competences usable in various process models and contexts	6	0	0	6	13
Making compromises in quality	3	3	3	3	9
Adaptability and flexibility	4	0	2	4	7
Reflection on working styles	3	0	3	3	6
Versatile method/practice toolbox	4	0	0	4	6
Platform-agnostic skills	4	0	0	4	5
Open-minded quality thinking	5	5	5	0	5
Independent problem solving capability	3	0	0	3	5
Understanding changing nature of quality	4	4	4	0	3
Discipline	2	2	0	2	2
Domain-agnostic competences	2	0	0	2	2
Generic software quality and testing competences	1	1	1	1	1
Understanding quality and its practices	2	0	2	0	1
Using analysis and reporting tools	1	0	0	1	1
Scaling personal toolbox	1	0	0	1	0
Understanding technology	1	0	1	0	0
Organisational skills	(Comps: 15)				**296**
Collaboration skills	10	0	10	10	62
Communication skills	10	0	10	10	50
Team skills	10	0	1	9	43
Role finding	8	0	0	8	32
Cultural skills	7	7	7	7	23
Working under insecurity and change	6	6	1	3	21
Social skills	6	0	0	6	17
Quality advocacy	7	1	1	7	17
Dependability	4	3	0	4	9
Cultural adaptation	4	0	0	4	7
Managing change with information	3	0	0	3	7
Entrepreneurial competencies	3	0	0	3	4
Networking skills	3	0	0	3	3
Service skills at all levels of the organisation	1	0	0	1	1
Scaling resource management practices	1	0	0	1	0
Product concept development, experimentation	(Comps: 12)				**123**
Experiment design skills	7	0	0	7	28
Evaluation of product concepts	5	0	0	5	17
Prototyping skills	5	0	0	5	17
Doing ethical assessment	5	0	0	5	16

Competence	No. of changes	#O	#U	#A	Importance
Doing proof of concept tests for technology	4	0	0	4	12
Business and product concept level testing	4	0	0	4	10
Doing critical technology assessments	3	0	0	3	8
Understanding about ethics	5	5	5	0	7
Understanding innovation	4	0	4	0	3
Understanding new products and systems	3	2	3	0	3
Doing experiments with users	1	0	0	1	1
Assessment and testing of innovations and product concepts	1	0	0	1	1
Product engineering	(Comps: 23)				**133**
UX and usability testing	7	1	1	7	33
Data analysis	5	0	5	5	22
Architecture evaluation	5	0	1	5	17
UX testing for feature development	4	3	3	4	15
Configuration management	5	0	0	5	11
Comparison testing	4	0	0	4	8
Rigour in collaborative keeping the platform robust and tolerating change	4	0	0	4	6
Using exploratory testing for understanding the behaviour of technology	3	0	0	3	5
Short work-in-progress lists	3	2	2	3	5
Configuration testing	2	0	0	2	3
Testing of complex interactions	2	0	0	2	2
Exploratory testing for feature development	1	0	0	1	1
Helping developers in development queue	1	0	0	1	1
Efficient and secure data collection	1	0	0	1	1
Installation testing	1	0	0	1	1
Robustness testing	1	0	0	1	1
Instrumenting of systems	1	0	0	1	1
Managing of test environments in the cloud	1	0	0	1	0
Testing tool UX design	1	0	0	1	0
Learning new testing tools	1	0	0	1	0
Understanding virtualisation	1	0	1	0	0
Virtual environment deployment skills	1	0	0	1	0
Virtual environment design and implementation	1	0	0	1	0
Products, technology	(Comps: 7)				**64**
Information systems and integration competences	6	5	5	6	30
Hardware-related skills	5	0	5	5	16
Understanding permission, security, privacy	5	5	5	0	7
IoT-related competences	3	2	2	3	6
Understanding complex systems	4	0	4	0	4
Understanding technical systems	2	0	2	0	1
Understanding cloud systems, their possibilities and problems	1	0	1	0	0
Risk related	(Comps: 9)				**105**
Security assessment and testing	10	0	0	10	43

Competence	No. of changes	#O	#U	#A	Importance
Risk thinking	12	0	12	0	24
Understanding information security risks	8	8	8	0	15
Risk, safety and reliability analysis	5	0	0	5	11
Risk analysis skills	2	0	0	2	4
Safety management	2	0	2	2	3
Product risk analysis	2	0	0	2	2
Customer's risk analysis	2	0	0	2	2
Risk-based testing	1	0	0	1	1
Process related	(Comps: 18)				**96**
Process development	11	2	9	10	48
Right timing of actions	7	1	1	7	20
Understanding the product development paradigm	6	1	6	0	6
Understanding overall product lifecycles	5	0	5	0	4
Deployment and automation skills	4	0	0	4	4
Understanding software engineering	4	0	4	0	2
Understanding of needs of development	3	1	3	0	2
Cost-benefit thinking in selecting quality practices	3	2	2	1	2
Forming and promoting practices	2	0	0	2	2
Sense of rhythm	2	0	2	0	1
Work and process design	1	0	0	1	1
Service design	1	0	0	1	1
Work, process and practice design	2	1	1	2	1
Understanding product/system development	2	2	2	0	1
Test planning for purpose	1	0	1	1	1
Understanding deployment	1	0	1	0	0
Supporting deployment decisions with assessment and test information	1	0	0	1	0
Understanding the relevant lifecycles	1	0	1	0	0
Competence development	(Comps: 8)				**56**
Personal competence development	7	3	3	7	26
Competence development focused on business needs	5	0	5	5	14
Changing company-level competence profile	3	3	3	3	7
Using social media and web in getting information and sharing information	3	3	3	3	4
National competence infrastructure development	2	0	0	2	2
Understanding about competence	3	0	3	0	1
Improving education for quality and testing	1	0	1	1	1
Organisational competence development	1	0	1	1	1

Quite many of the listed competences are generic competences for ICT professionals and one may wonder whether they should be included. The only way to show which generic competences are important or relevant is to analyse specific contexts, as in this case testing and quality assurance.

The competences listed here provide raw material for making conclusions of the research and we will look into them again later from various viewpoints.

5.16.5 Competences related to activity system elements

One working hypothesis of the dissertation was that some extra insight could be gained by using the triangle model of action research, presenting the activity system. A modified version of it was presented containing the following elements:

- Self.
- System under test.
- Tools and methods.
- Development goals.
- Organisation.
- Teamwork.
- Processes.

The competences were grouped by the nodes of the version of the activity system triangle used. This should do two things. First, it should provide added insight. Second, it should provide some triangulation [sic.] of the previous findings. Note that we will return to the triangle again later. The grouping is shown in Table 49.

Table 49. Competences divided into nodes of the activity system.

Competence	No. of changes	#O	#U	#A	Importance
Self	(Comps: 15)				**436**
Multi-skilledness	14	12	12	14	100
Broad flexible competence	13	13	13	13	100
Business understanding	20	18	20	0	74
System and system of systems thinking and testing	9	0	9	8	52
Personal competence development	7	3	3	7	26
Active, self-steered working for quality	7	0	0	7	22
Creativity	7	0	0	7	18
Critical thinking and presenting critique	5	0	0	5	16
Dependability	4	3	0	4	9
Adaptability and flexibility	4	0	2	4	7
Independent problem solving capability	3	0	0	3	5
Reputation management	3	0	0	3	3
Discipline	2	2	0	2	2
Understanding about competence	3	0	3	0	1
Handling contradictions	2	0	2	0	1
System under test	(Comps: 14)				**65**
Understanding overall contexts	10	0	10	0	18

Competence	No. of changes	#O	#U	#A	Importance
Hardware-related skills	5	0	5	5	16
Understanding permission, security, privacy	5	5	5	0	7
IoT-related competences	3	2	2	3	6
Platform-agnostic skills	4	0	0	4	5
Understanding complex systems	4	0	4	0	4
Understanding new products and systems	3	2	3	0	3
Understanding customers	3	0	3	0	2
Understanding users	3	0	3	0	2
Understanding modern word and its new practices and thinking	2	2	0	0	1
Understanding technical systems	2	0	2	0	1
Understanding cloud systems, their possibilities and problems	1	0	1	0	0
Understanding systems' requirements	1	0	1	0	0
Understanding technology	1	0	1	0	0
Development goals	(Comps: 11)				**108**
Customer-centredness	7	7	7	7	39
Risk thinking	12	0	12	0	24
Understanding information security risks	8	8	8	0	15
Making compromises in quality	3	3	3	3	9
Understanding about ethics	5	5	5	0	7
Open-minded quality thinking	5	5	5	0	5
Understanding changing nature of quality	4	4	4	0	3
Understanding the customer's business and needs	3	2	3	1	3
Understanding of product, product culture, businesses and their needs	4	0	4	0	2
Understanding quality and its practices	2	0	2	0	1
Understanding what qualities are the most critical at each phase of the company's lifecycle phase and the product development phase	1	0	1	0	0
Organisation	(Comps: 18)				**154**
Communication skills	10	0	10	10	50
Cultural skills	7	7	7	7	23
Social skills	6	0	0	6	17
Quality advocacy	7	1	1	7	17
Competence development focused on business needs	5	0	5	5	14
Cultural adaptation	4	0	0	4	7
Changing company-level competence profile	3	3	3	3	7
Entrepreneurial competencies	3	0	0	3	4
Understanding domains, contexts and situations	5	4	5	0	4
Networking skills	3	0	0	3	3
Domain-agnostic competences	2	0	0	2	2
National competence infrastructure development	2	0	0	2	2
Improving education for quality and testing	1	0	1	1	1
Organisational competence development	1	0	1	1	1

Competence	No. of changes	#O	#U	#A	Importance
Service skills at all levels of the organisation	1	0	0	1	1
Understanding of domains and cultures	2	0	2	0	1
Scaling resource management practices	1	0	0	1	0
Understanding about the company's business	1	0	1	0	0
Teamwork	(Comps: 4)				**143**
Collaboration skills	10	0	10	10	62
Team skills	10	0	1	9	43
Role finding	8	0	0	8	32
Reflection on working styles	3	0	3	3	6
Processes	(Comps: 23)				**136**
Process development	11	2	9	10	48
Working under insecurity and change	6	6	1	3	21
Right timing of actions	7	1	1	7	20
Competences usable in various process models and contexts	6	0	0	6	13
Understanding the product development paradigm	6	1	6	0	6
Understanding overall product lifecycles	5	0	5	0	4
Deployment and automation skills	4	0	0	4	4
Understanding innovation	4	0	4	0	3
Understanding software engineering	4	0	4	0	2
Understanding of needs of development	3	1	3	0	2
Cost-benefit thinking in selecting quality practices	3	2	2	1	2
Focusing and prioritising actions at each startup phase	1	0	1	1	2
Forming and promoting practices	2	0	0	2	2
Sense of rhythm	2	0	2	0	1
Work and process design	1	0	0	1	1
Service design	1	0	0	1	1
Special testing services	1	0	0	1	1
Work, process and practice design	2	1	1	2	1
Understanding product/system development	2	2	2	0	1
Test planning for purpose	1	0	1	1	1
Understanding deployment	1	0	1	0	0
Supporting deployment decisions with assessment and test information	1	0	0	1	0
Understanding the relevant lifecycles	1	0	1	0	0
Tools and methods	(Comps: 47)				**362**
Security assessment and testing	10	0	0	10	43
UX and usability testing	7	1	1	7	33
Information systems and integration competences	6	5	5	6	30
Experiment design skills	7	0	0	7	28
Data analysis	5	0	5	5	22
Evaluation of product concepts	5	0	0	5	17
Prototyping skills	5	0	0	5	17
Architecture evaluation	5	0	1	5	17
Doing ethical assessment	5	0	0	5	16

Competence	No. of changes	#O	#U	#A	Importance
UX testing for feature development	4	3	3	4	15
Doing proof of concept tests for technology	4	0	0	4	12
Configuration management	5	0	0	5	11
Risk, safety and reliability analysis	5	0	0	5	11
Business and product concept level testing	4	0	0	4	10
Doing critical technology assessments	3	0	0	3	8
Comparison testing	4	0	0	4	8
Managing change with information	3	0	0	3	7
Rigour in collaborative keeping the platform robust and tolerating change	4	0	0	4	6
Versatile method/practice toolbox	4	0	0	4	6
Using exploratory testing for understanding the behaviour of technology	3	0	0	3	5
Short work-in-progress lists	3	2	2	3	5
Understanding of possibilities and alternatives in testing	5	0	5	0	4
Using social media and web in getting information and sharing information	3	3	3	3	4
Risk analysis skills	2	0	0	2	4
Configuration testing	2	0	0	2	3
Safety management	2	0	2	2	3
Testing of complex interactions	2	0	0	2	2
Product risk analysis	2	0	0	2	2
Customer's risk analysis	2	0	0	2	2
Doing experiments with users	1	0	0	1	1
Assessment and testing of innovations and product concepts	1	0	0	1	1
Exploratory testing for feature development	1	0	0	1	1
Helping developers in development queue	1	0	0	1	1
Generic software quality and testing competences	1	1	1	1	1
Efficient and secure data collection	1	0	0	1	1
Using analysis and reporting tools	1	0	0	1	1
Installation testing	1	0	0	1	1
Risk-based testing	1	0	0	1	1
Robustness testing	1	0	0	1	1
Instrumenting of systems	1	0	0	1	1
Managing of test environments in the cloud	1	0	0	1	0
Scaling personal toolbox	1	0	0	1	0
Testing tool UX design	1	0	0	1	0
Learning new testing tools	1	0	0	1	0
Understanding virtualisation	1	0	1	0	0
Virtual environment deployment skills	1	0	0	1	0
Virtual environment design and implementation	1	0	0	1	0

The node for tools and methods has the highest importance value. That is actually as expected, because the competences listed are concrete and specific to testing.

As second comes the area for self. In a dynamic and demanding environment the personal qualities really matter. The competences in that area are not specific to testing, but generic competences for participants of product development.

Next up are, quite close to each other, organisational and process competences. Those are the environments where people work and thus need good understanding of. Product development is after all, collaboration inside an organisation, structured by some mutually understood process.

Here we need to remember that the analysis of the changes did not look into issues from this viewpoint. It is natural that for example discussion of technological changes focuses to competences that are reasonably directly related to the elements of technology. Thus, another view that is more focused to the activity system will potentially utilise the model more effectively. This is why the previous table should not be taken too literally and the reader is advised to make conclusions only later.

6 Reflective survey to Finnish testing professionals

6.1 Introduction and goals

To find out how the members of the Finnish testing community – the professionals in the field – see as the central testing competences in the future, a survey was made for the members of, TestausOSY (www.testausosy.fi) during spring 2014.

The research tried hard to avoid the problems of using large volume surveys, because that will turn the results into a description of the mediocrity and because people usually see the opportunities of the future as answers to problems they had a couple of years ago. That is because they only understand properly the past, and can see resolutions only in what they understand.

The survey was used to in deductive mode to gain insight of testing experts' thinking and to deductively construct small models of their views. This was only to see if there is any new "weak signals", not to create any real valid basis as such.

The goals of the survey were:

- Get weak signals from the field (and be ready to follow-up on those if seen useful).
- Get a sample of what the active actors in the tester community think about the future for author's and readers' reflection and enrichment of information.
- Deductive forming of mini models or the presented thinking to visualise the ideas usually presented in very fragmented form.
- Possibility to find people whose interviewing could be useful.

The goals did not include these:

- Any validation of information or knowledge or theory.
- Any validation of the syntheses about future needs presented earlier.
- Forming of any valid information basis about the future.

The reader is adviced to not get any false impressions about the survey from it lengthy presentation here.

6.2 Challenges of this kind of survey

A survey about the future is challenging and it was expected that the number of respondents would be small and that the results might now have restricted value. Already in 2010 when the author published his future-oriented slide set “Trajectories of testing” (later version is Vuori, Matti. 2014d), it was already generally understood that even professionals respond to changes in their local context, but not to changes in the external context of product or system development. They also define their world view by their daily activity and their school of testing. That is why there would probably be only a small number of responds and they might be very “siloed”, that is, thinking about testing based on their representative testing school or based on the needs and practices on the domain.

As the idea was to get fresh ideas and perhaps even weak signal of what is happening, all questions were open questions, further restricting the number of people who would be willing to respond. It should also be noted that 2014 was not a psychologically good time for testers. During that time there were lots of ideas testing testing into automated testing done by developers, or even stopping testing altogether. The ICT industry was in bad shape. Companies were not doing well and many testers were unemployed. That kind of situations is not good for thinking about the future.

However, the numbers are not essential here, but getting the views of the ones who have thought about the upcoming challenges.

6.3 Respondents and value of the survey

The number of respondents turned out to be 13, which corresponded with the expectations.

As would be expected, the respondents were very experienced, thus having must likely the ideas about the future and the confidence to express them (in an anonymous survey). The experience in ICT was in average 16 years (range 3...30) and in testing or quality 13 years (range 3 to 24). Gender or location of the respondents were not asked as there is no theory that they would have any relevance here.

As for how competent the respondents were? That is not something one can assess about oneself. Even experience is not a reliable measure of it. Expertise can usually be assessed by the answers and, furthermore, the author knew many of them either in social media or in "real life". Based on that, the average competence of the respondents was very high.

This is all good. But at the same time it was seen that the expectations that the respondents represent only their domain and not the synthesis of the field that is both converging in technological challenges and diverging in other aspects. That is why the 13 respondents were a suitable sample size. More of the same would not produce valid ideas about the future even for the contexts of the respondents, because they are seeing the various changes.

6.4 Should this be earlier? Or included at all?

Actually, the survey produced conservative ideas that reflect the current mainstream thinking about testing. It was also done before the analysis for the future got into speed, so it couldn’t be used for validating any of the presented hypotheses for the future.

So why is it presented here and not earlier, perhaps used here and there enrichening other analyses? That decision was reader-centred, with a goal of giving more rhythm to the text and some variation sandwiched between the analysis of changes and the following synthesising chapters.

There was even an idea that this part would not be included at all, as its role is easy to understand wrongly, but the benefits of inclusion clearly outweighed the exclusion. Without the inclusion, the first thing a reader would have asked would have been: “ok, but what do the practitioners think about all this?” Now we can tell something about that sufficiently, but have not used too much time on that.

6.5 Survey process

The channels for the web survey were the community’s e-mail list (980 addresses at the time when the survey started) and the LinkedIn group of the community (820 members at the time). Majority of both populations are the same people, but there is no exact information about the overlap available. Survey was opened 12.3.214 and closed 15.4.2013.

The invitations were sent only once as it is not considered polite to repeat mailings on email lists. Also, there was no need to gain any more responses, as pointed out in previous chapter.

The framing of the questions was simple:

- “In your opinion, what would a good tester be in Finland in 2025 like? What are all the things that she does? What things is she good at? What are the special things that she can do? What does she concentrate in?”
- “What are the most important differences with the current situation? What are the main differences in tasks and ways of working? What kind of (new?) competences are emphasised in the near future?”
- “To justify the answer, tell a little about views about the tester’s future environment? (For example the ways the organisation works, the tools in use etc.). Limit as necessary – tell for example in what domain you see in your mind the tester you described.”
- “For background, tell a little about yourself. How many years of experience from this area (testing, quality, product development) are your views based on?”

The survey used only open-ended questions, because the purpose was not to get selections from a ready-made list of possible answers, but give room for new thoughts and new questions. The results were first presented in May 2014 in a seminar of Ohjelmistotestaus ry and in June in Tampereen Testauspäivä (Tampere Testing Day) (Vuori, 2014b). The raw answers are in Appendix 2.

6.6 Analysis of the answers

Each respondent has a slightly different story. They all told their own hypothesis, views to the possibilities, what might be coming. Influencing this are the respondents' domains. That produces contextual understanding about what things are those that specifically competent people are needed for. For example, in the making of embedded systems different things are emphasised than in the making of tailored information systems.

The answers were analysed from bottom-up fashion:

1) Statements presented in answers were combined together on their main themes.

2) Statements were linked together to form flows from neutral statements about competence to ones that describes ability in some situation and statements that describe the value of that.

3) Those flows were drawn as graphs to visualise the connections.

4) The emerged graphs were then reflected upon by the researcher: what is the story that shows in them? How does it relate to what other sources tell about the issue?

5) Finally, the issues were combined under higher level sections to bring more structure to the presentation in the dissertation.

This is a form of theory building in the fashion of grounded theory, but here we use the survey responses only as a very partial source of information.

The raw answers are included as an appendix as in this kind of research, hearing the people’s voice directly, without coding, is invaluable.

6.7 Answers and reflections

The next pages will present diagrams drawn directly from what the people said. The diagrams include many answers synthetized into a graph. The views of various people are combined by themes under respective headers. Between the diagrams there are reflections about the answer. Diagrams are partially colour coded, see Figure 65.

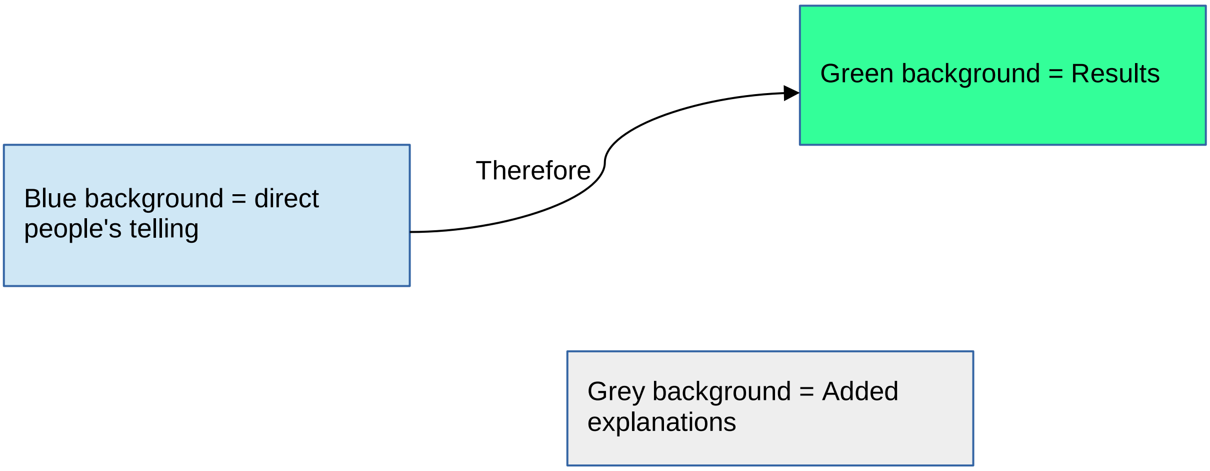

Green background = Results
Therefore
Blue background = direct people's telling
Grey background = Added explanations

Figure 65. Colour codes of the graphs drawn from the responses.

6.8 Work profile

6.8.1 Tester role and title

Answers related to tester role are presented in Figure 66.

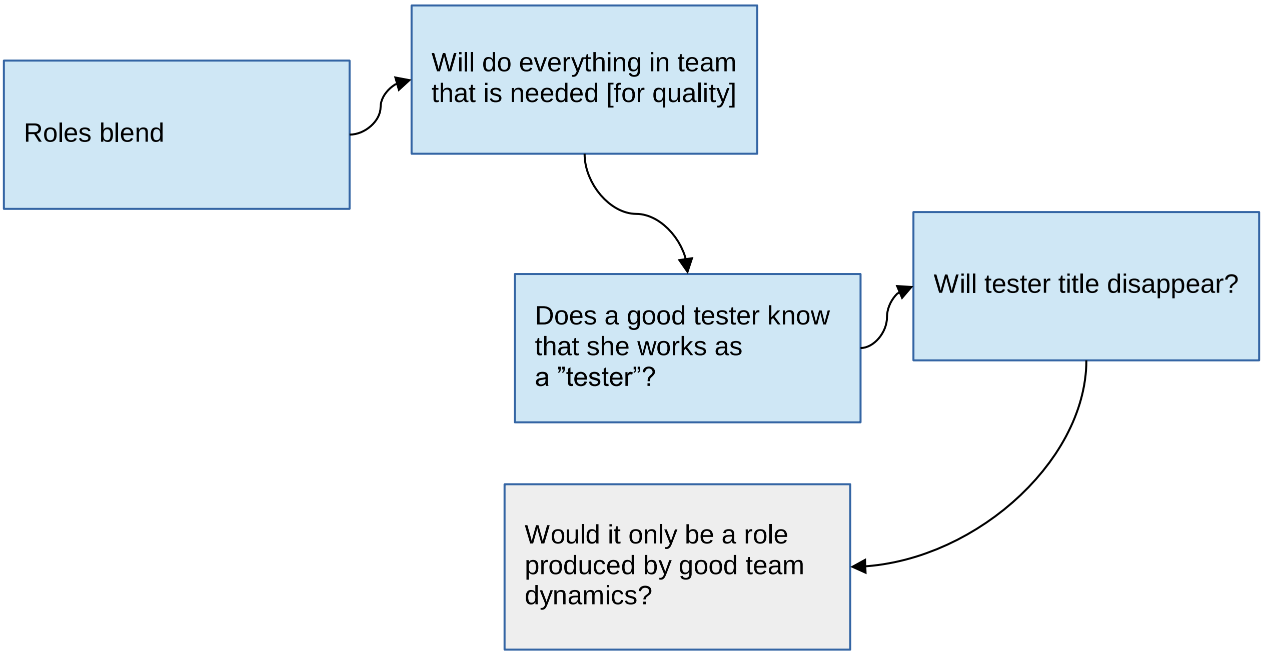

Roles blend
Will do everything in team that is needed [for quality]
Does a good tester know that she works as a ”tester”?
Will tester title disappear?
Would it only be a role produced by good team dynamics?

Figure 66. Specificity of the role and the title.

The respondents noted the trend of roles blending and the need for effective teamwork. Perhaps even the title “tester” will disappear in some contexts. Indeed, a “tester” is born by naming someone as such or by giving her such tasks. When collective competence is mature, there will emerge testing tasks from the team's activity and if the team is very dynamic, the tasks can get divided for people in an optimal way. But the collective competence is not always mature and not even developing into maturity. Thus, a team's self-guiding characteristic can be weak. As a result, there will be self-misguiding. Team dynamics can lead to a situation where the people's competence and roles do not meet. To get the dynamism working requires time and iteration, during which the team is supposed to get a couple of projects done. Still, a special “tester” can be thought to be needed for these reasons:

- The blindness of people about their work can be reduced if there is someone who has been assigned to monitor that blindness.
- A counter force to business pressures – so as not to shoot oneself in the foot.
- Speeding up of team dynamics.
- Explication of special skills.
- When there are specific QA processes and tasks require specialist skills.

6.8.2 Attitude

The first diagram presents answers linked to attitude and character Figure 67. The numbers in parenthesis show how many respondents raised an idea in their answers.

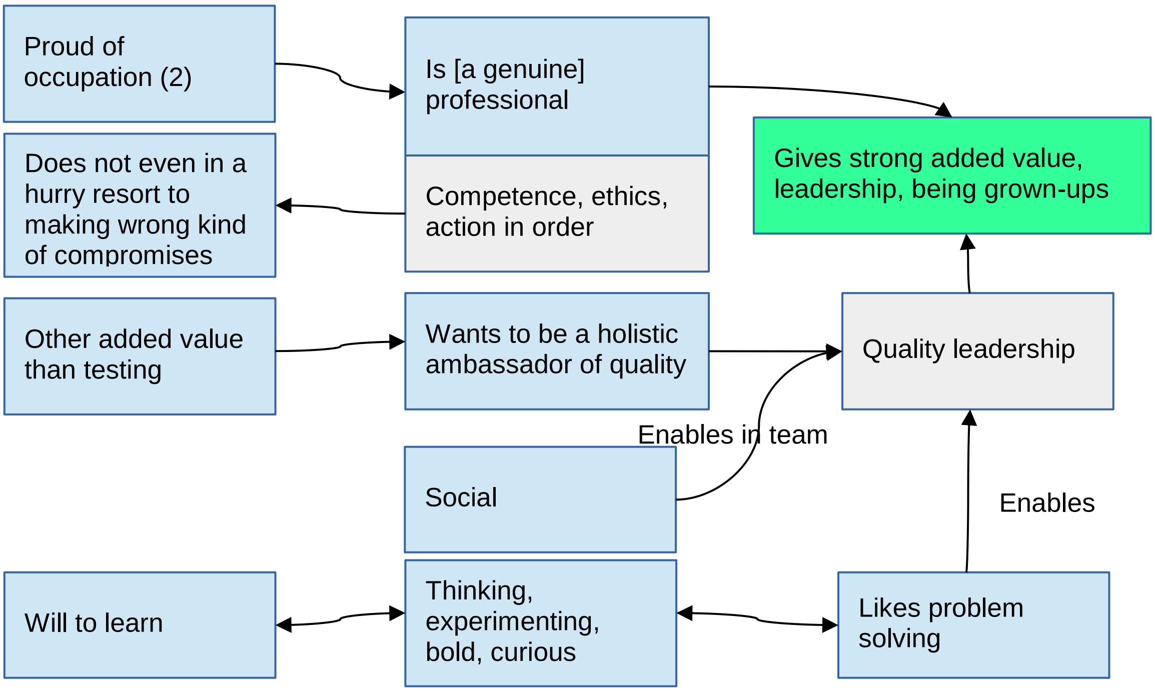

Proud of occupation (2)
Is [a genuine] professional
Competence, ethics, action in order
Does not even in a hurry resort to making wrong kind of compromises
Gives strong added value, leadership, being grown-ups
Other added value than testing
Wants to be a holistic ambassador of quality
Quality leadership
Enables in team
Social
Enables
Will to learn
Thinking, experimenting, bold, curious
Likes problem solving

Figure 67. Attitude and character.

Feeling of tester's own professional identity is an essential part of professionalism and the added value produced through it. This no matter what the professional special areas and practices would be. This is emphasised because in the fragmentation and chaos of the world, clarity at personal level is needed – as it may not always exist in in processes or in culture. Identity produces understanding, clarity, potential for passion towards work and that in turn gives many essential benefits – a will to do excellent work, to develop practices, to take responsibility. It is expected that many of the respondents of a survey like this tend to respect professionalism and quality more than average and that is a good thing because if they don't, who does? Note that one critical premise of the dissertation is that quality matter and that there is a need for people who work on that more than others, because of the pressures for too many quality compromises.

A central part of identity is ethics, see for example Vuori (2010d). That is also helped by passion. About that see Vuori (2010c).

Identity does not mean limitations to the work profile (”I don't do that”), but the opposite: when the goals of tester’s own role are understood (in a context of shared goals), one can find new means. Still: one needs to protect her own focus when it is in danger to get lost.

Quality leadership is often mentioned – here and elsewhere. But what is it really? First, a couple of words about leadership in general. An anecdote, supposedly from management guru Peter Drucker: "Management is doing things right; leadership is doing the right things". This is obviously critical for quality. In the previous decades, quality leadership would have been psychological leadership by top management: the company makes quality, it is important, don't do compromises, think about the customer. A leader gives vision, shows example, and maintains morale. Sometimes a company might have had a quality manager, but her job is more measurement, reporting, process improvement – not leadership.

Quality leadership is needed in teams (product teams, all startups) as a viewpoint that maintains good practices, gets people to concentrate on the right things, helps people identify quality problems, helps improve things, reminds all about factors that are important for success, maintains memory of past pitfalls, maintains dialogue about quality, and helps people learn about quality and how to produce it. This requires orientation, motivation, vision, inner flame, competence and so on. From which roles / jobs could such be found? Asking this question is more important than ever and its importance continues to increase.

There are no general answers to it, however. Anyone can become the carrier of the torch. The dynamics in the organisation should be such that all teams will be organised themselves so that this role is also filled. There are generally two ways for the formation:

1. A manager decides the group’s composition. She can obviously take into consideration this, among other factors related to each member’s roles.
2. The group self-forms and self-organises itself, filling the formal and psychological roles by themselves and through team dynamics.

Ultimately, whether the leadership role will be filled, depends on the quality culture of the organisation.

6.8.3 What does the tester concentrate on?

Concentration is essential in any expert task. Figure 68 presents respondents answers related to that.

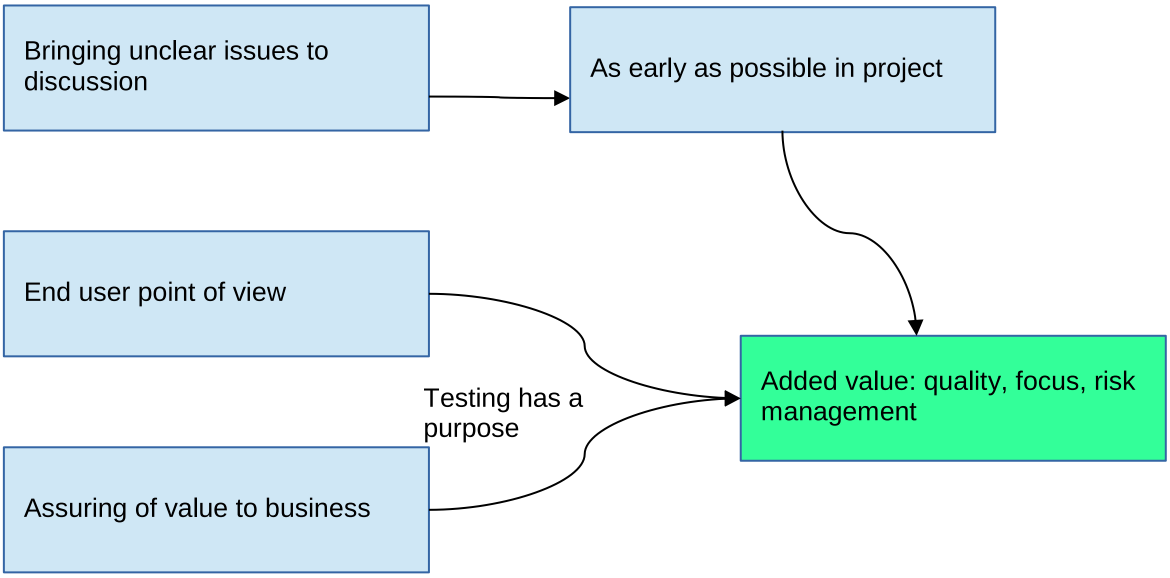

Bringing unclear issues to discussion
As early as possible in project
End user point of view
Testing has a purpose
Assuring of value to business
Added value: quality, focus, risk management

Figure 68. What does the tester concentrate on?

Raising issues to discussion is important. The issues are not only defects, but any things that should be tackled. Sometimes those are issues that others seem reluctant to face or are ignoring. If doing that becomes, team dynamically, a task for someone, it is hard to think that she would be someone other than a tester. It is often noted that a tester's role is to be an observer and analyser. Unlike others, a tester has tools for that. Therefore, the issues can be raised in a fruitful way: there are facts, experiences, the ability to analyse the information. Many core ideas of testing culminate in this task:

- Responsibility about quality.
- Producing information – making sure that people have the facts they need for addressing the issues.
- Communication in a way that supports common goals.
- Collecting of facts to support decision making.

Note again, that tester in this discussion does not mean occupation. In fact, the observer role reminds that one can become a "tester" by group dynamics. There will often be an observer and analyser and that can lead to more focused testing.

Other than that, the idea that a tester should often represent the user is a common one and a solid one.

6.8.4 Breadth of work profile and learning

Answers related to the breath of the work profile of tester are mapped in Figure 69.

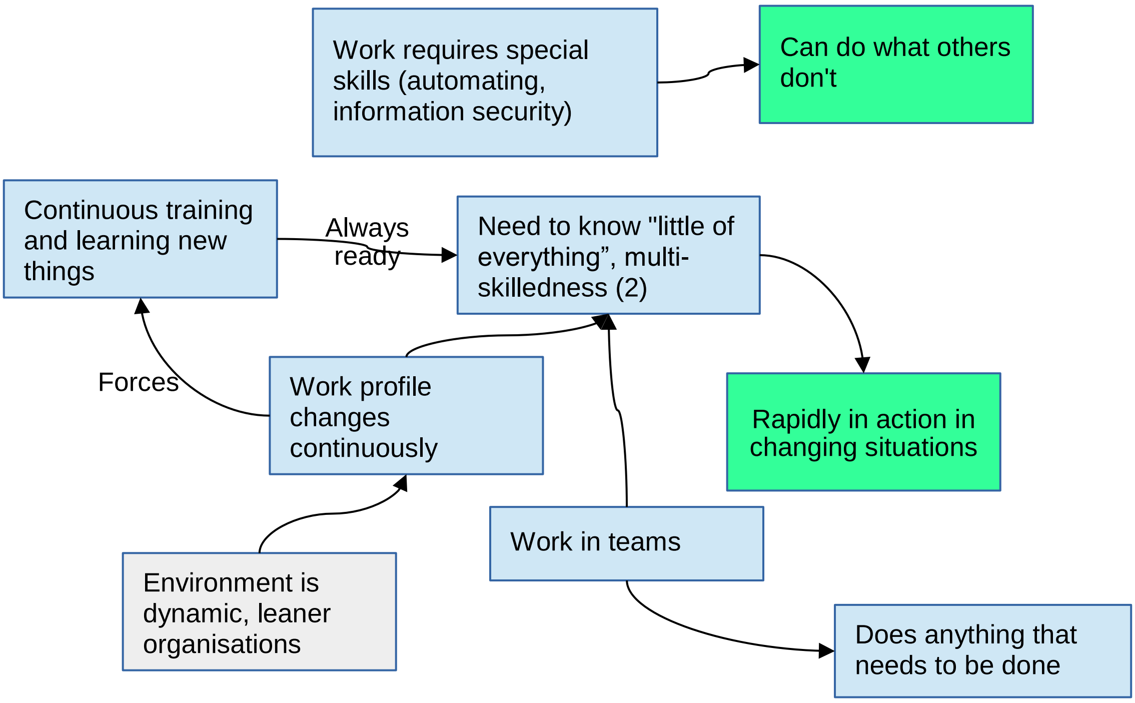

Work requires special skills (automating, information security)
Can do what others don't
Continuous training and learning new things
Always ready
Need to know "little of everything", multi-skilledness (2)
Forces
Work profile changes continuously
Rapidly in action in changing situations
Work in teams
Environment is dynamic, leaner organisations
Does anything that needs to be done

Figure 69. Breadth of work profile and learning.

The respondents noted that continuous learning and multi-skilledness are relevant due to environmental dynamics and teamwork. But if one must learn all the time, how does it happen in the everyday of the working life? Note that one should learn for the next job, to be ready for it! The current job will change to another when this one has been learned properly. There are many questions that lack good answers:

- How to support learning from colleagues? As we aim for the performance of the team and the organisation, how to support the learning of others? How to bring to the activities more the elements of shared learning? That would mean for example reflecting, externalising and experimenting.
- Is there any value in a proactive collecting of certificates?
- How to apply mentoring? How to bring the national network of colleagues / the community for support?
- How to bring elements of coaching to management?

6.8.5 Changing jobs – robotics and polarisation

Answers related to changing jobs due to new technologies are mapped in Figure 70.

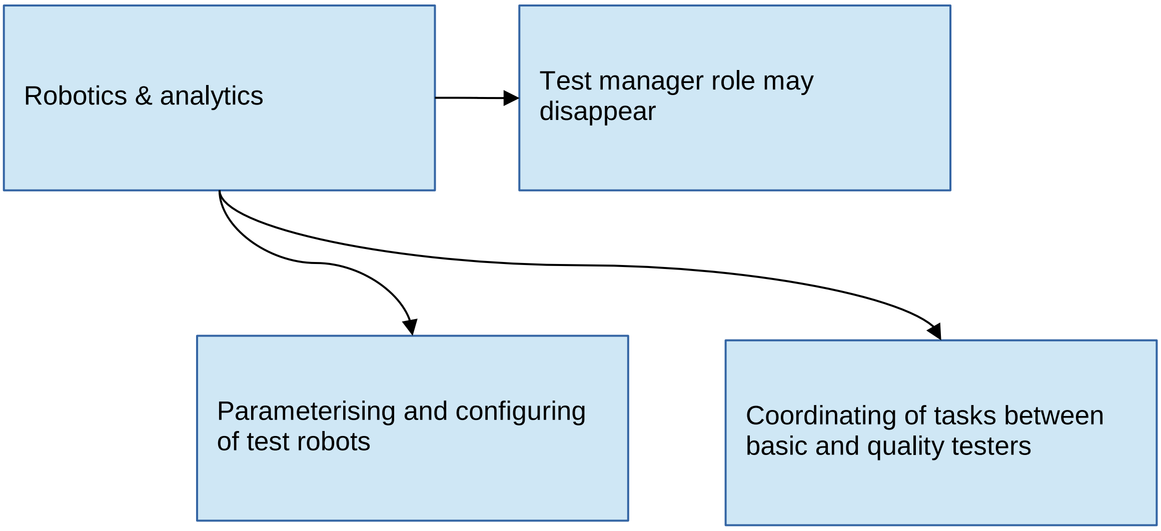

Robotics & analytics
Test manager role may disappear
Parameterising and configuring of test robots
Coordinating of tasks between basic and quality testers

Figure 70. Changing of jobs.

Robotics was a new issue in media at the time of the survey. At that time robotics were used in testing in the form of physical robots that would use a user interface with a robot finger (or multiple fingers) in the same way as a human would. They would be used in testing response characteristics of devices and non-intrusive functional testing, which that would require no instrumentation in the product (Vuori, 2013). After that software robotics and artificial intelligence have been discussed plentifully in media and one hypothesis seems to be that intelligent software agents could do much of what humans currently do and humans would just configure and manage such robots. There is still no sign of such intelligence in the field of testing. It is still mostly simple automation defined and programmed by humans and test actions are not started by intelligence, but just actions in the version control system or similar. There are areas in test management that could be supported by “software robotics”, but there is no need to call it that. The traditional term “process automation” describes it more clearly.

But this is an area that should be monitored.

This is very much related also to a possible polarisation of jobs are summarised in Figure 71.

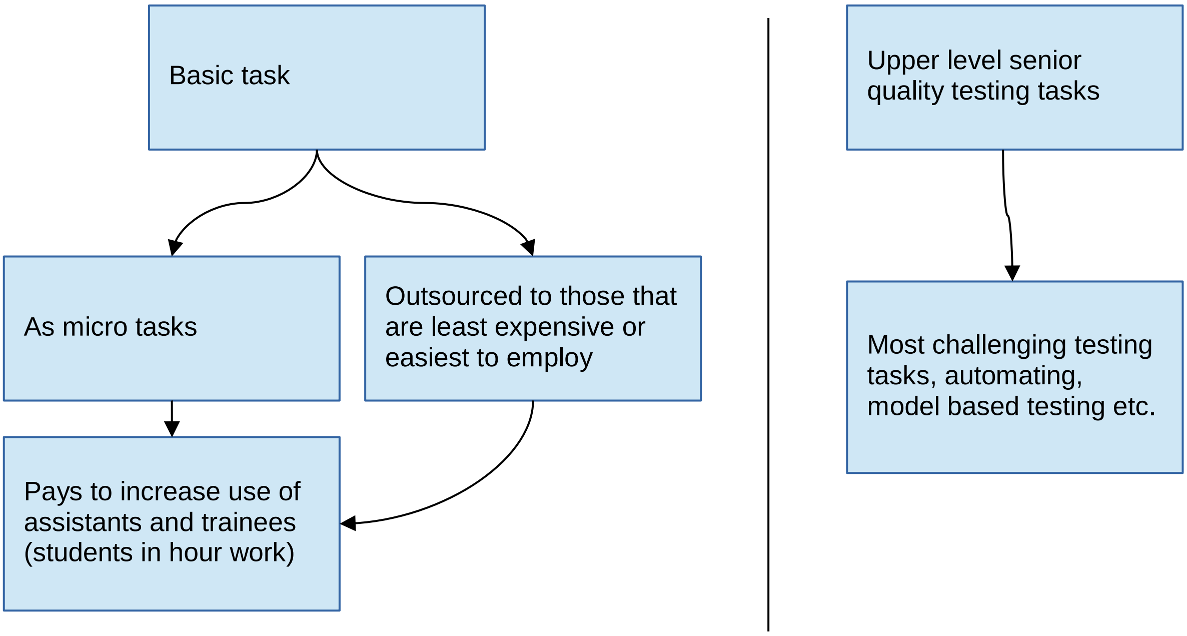

Basic task
As micro tasks
Outsourced to those that are least expensive or easiest to employ
Pays to increase use of assistants and trainees (students in hour work)
Upper level senior quality testing tasks
Most challenging testing tasks, automating, model based testing etc.

Figure 71. Polarisation of jobs.

Besides the positioning of competence, for example in the dimension of testing technology vs. testing for business, also the level or requirements for testing varies and perhaps polarises. A historical problem has been that people think that they can manage with one tester profile – either one – without thinking of the whole.

There seems to be several ideas linked to this. Yes, the basic tasks could be outsourced, but at the same time there is a tendency to automate the basic tasks (here one thinks of simple manual testing). Will there be simple automation tasks that can be outsources in a simple way? If the code base is public, as in open source or dual license development, yes. Perhaps the simple task will be given to intelligent software robots, which is different form of automation than what people have been used to in software development.

6.9 Competences

6.9.1 Nature of testing competences

The mapping of answers in Figure 72 is about the nature of testing competences.

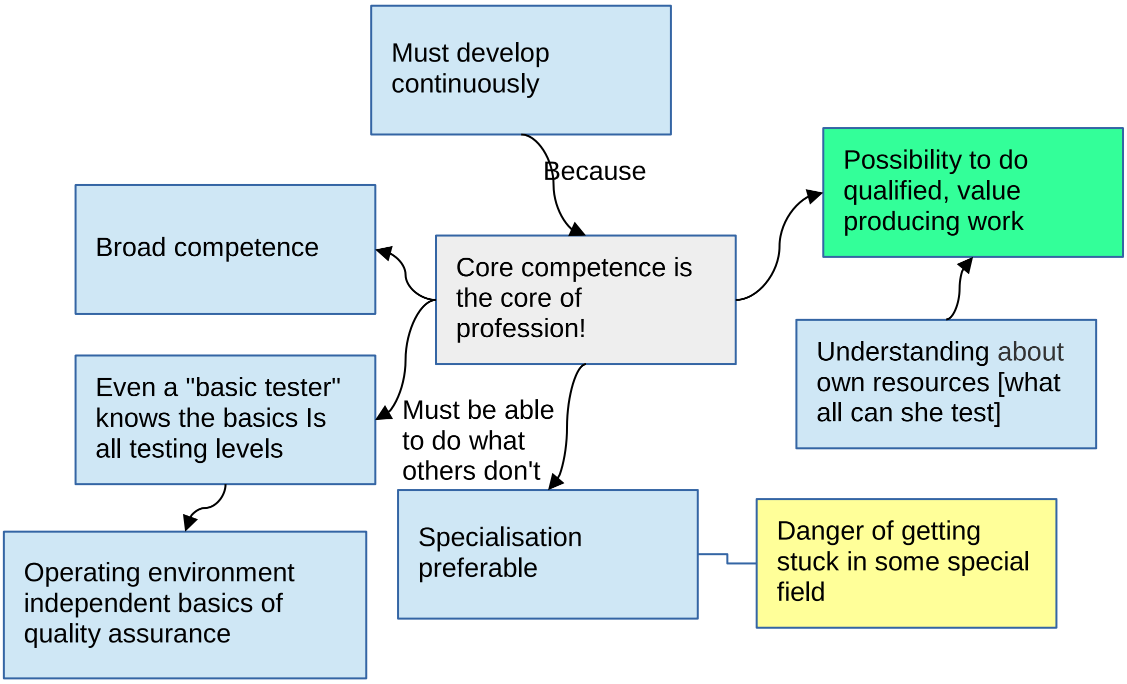

Must develop continuously
Because
Possibility to do qualified, value producing work
Broad competence
Core competence is the core of profession!
Even a "basic tester" knows the basics Is all testing levels
Must be able to do what others don't
Understanding about own resources [what all can she test]
Operating environment independent basics of quality assurance
Specialisation preferable
Danger of getting stuck in some special field

Figure 72. Nature of testing competence.

This graph of the responses again raises the issues of continuous learning, but from the viewpoint of having competence that others don’t have. Everyone in a team should possess something unique that others do not.

Also, the dangers of specialisation were raised. There is a danger of getting stuck in some focus area: One finds her own speciality in a narrow area (such as testing with a framework) and closes her eyes for everything else. A tester needs critical thinking about oneself and continuous renewing. A part of professionalism is recognition of what one really is proficient in and what all things one can do. Sometimes for example usability testing is considered trivial, but it is far from trivial and real competence is required for that. How about information security testing? One must activate the company, team, to fill the holes in competence internally and by subcontracting.

6.9.2 Competence palette

Answers related to tester’s competence palette are mapped in Figure 73

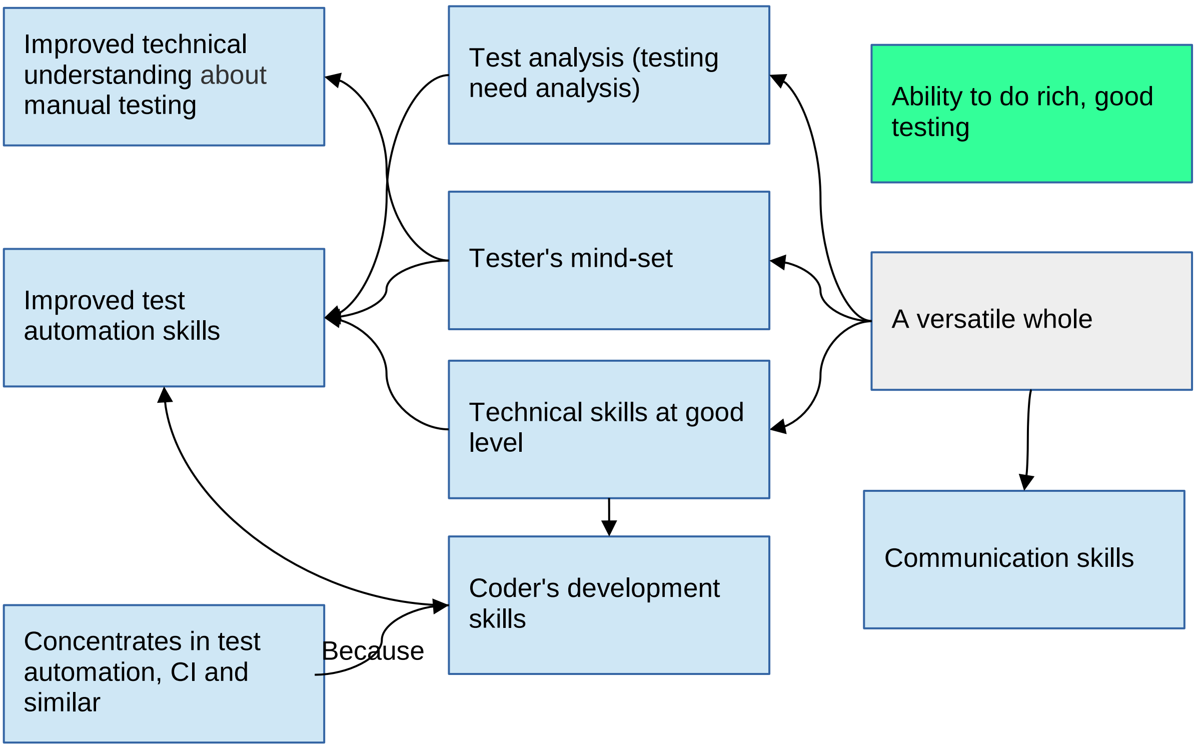

Improved technical understanding about manual testing
Test analysis (testing need analysis)
Ability to do rich, good testing
Tester's mind-set
Improved test automation skills
A versatile whole
Technical skills at good level
Communication skills
Coder's development skills
Concentrates in test automation, CI and similar
Because

Figure 73. Competence palette.

Many of the respondents raised the common issues of test automation as an important element of the near future, but in a context of technical skills. Good test automation needs good test analysis and generic technical skills. Knowing “coding” is not sufficient. This is still a challenge for testers that have a background in other disciplines – and we need those people too. We also need to remember the need for testing in other processes other than integrated in software implementation. The respondents clearly did not consider much the business- and concept level testing, but the tasks that a traditional “tester” does in software projects.

There is no basis for thinking that everyone should be able to program or work with test automation. The formula of expectations can be impossible to solve, if a large generic competence and basic readiness to everything is needed, but at the same time special skills are expected. A single person cannot be everything. Several different tester's concepts, and work profiles are needed, including a business oriented tester type and product technology-oriented type. Both can be mixed with other roles that produce suitable synergy, such as roles in user centred design and technical development.

At the core of all those are at least the tester's mindset, attitude and basic skills. Those are sometimes so obvious, that they are in danger of being forgotten. Still, they are what differentiate testers from others. Even those need expansion. Too often in the discussion about the profiles people think as a basis a "generic functionality tester". Yet, there have for a long time been special experts on usability testing, performance testing and information security testing. One challenge in testing is to get rid of the strong stereotype of testing being functional testing and remembering that there already are many kinds of testing and testers. New arrangements should perhaps be sought that all this expertise can be provided for companies in flexible ways.

6.9.3 Computer-aided testing

Answers related to “computer-aided testing” are shown in Figure 74.

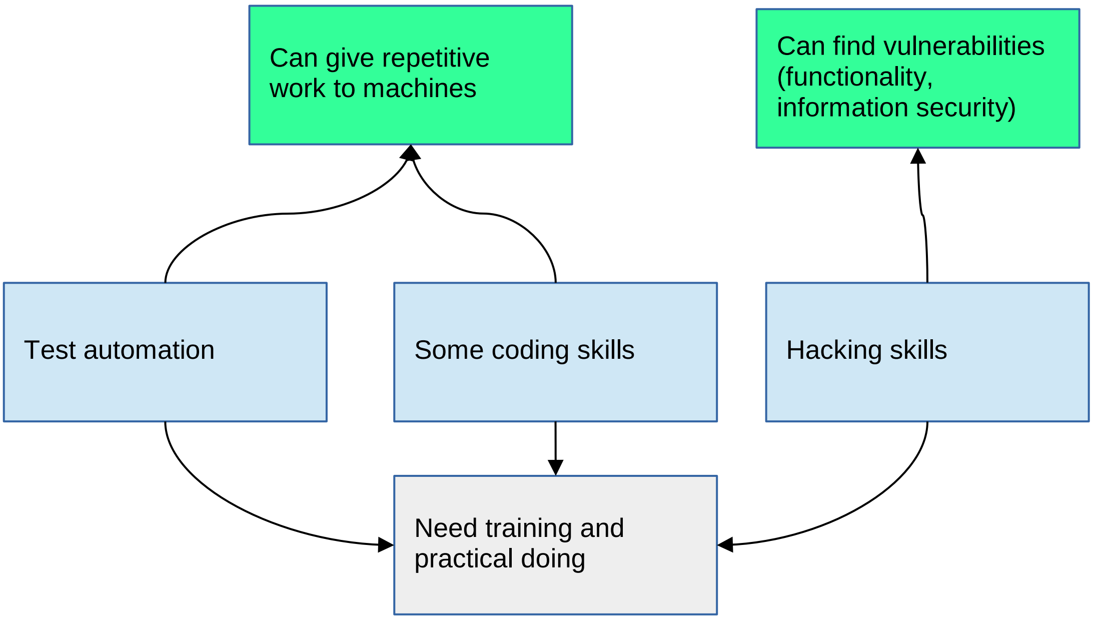

Can give repetitive work to machines
Can find vulnerabilities (functionality, information security)
Test automation
Some coding skills
Hacking skills
Need training and practical doing

Figure 74. Computer aided testing.

When we talk about test automation, we often only mention testers' “coding skills”. Coding only helps in making small and simple scripts. A more challenging testing benefits from hacker's skills, using those one can really see how the target behaves and what it tolerated. That would be a healthy breaking mentality taken into the order of two. However, it is pointless to expect that of everybody.

6.9.4 General competences

Figure 75 combines answers related to goals of work.

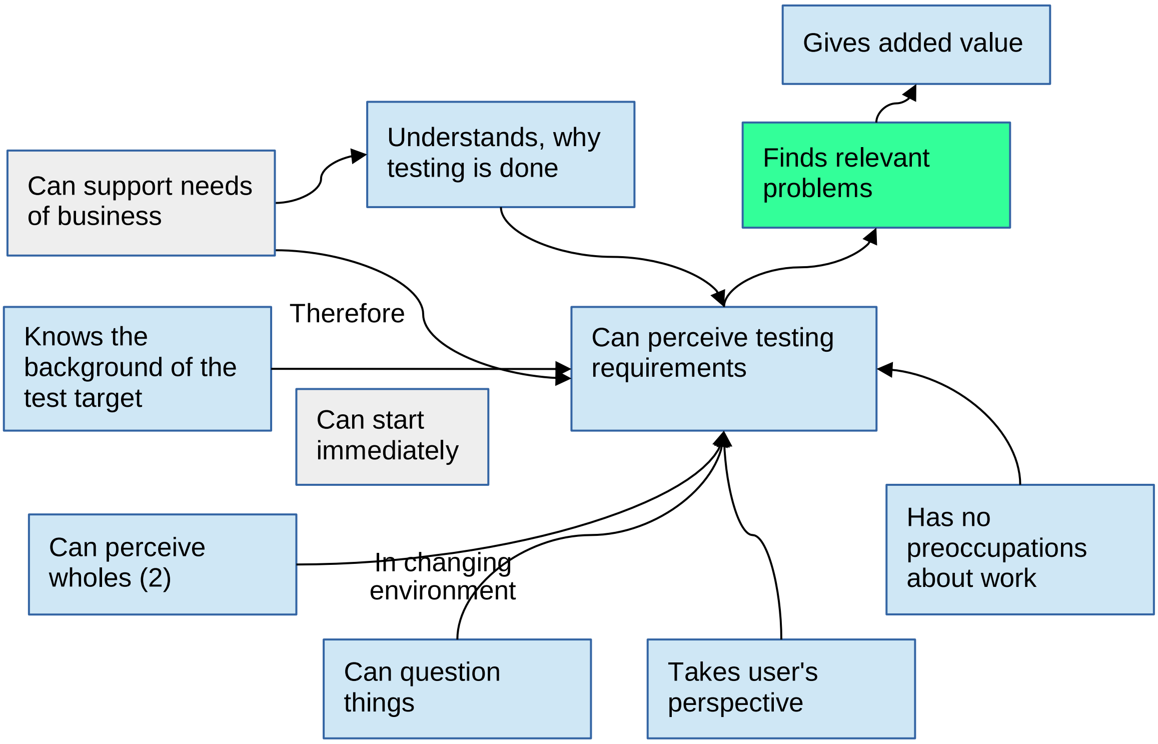

Gives added value
Finds relevant problems
Understands, why testing is done
Can support needs of business
Therefore
Knows the background of the test target
Can perceive testing requirements
Can start immediately
Can perceive wholes (2)
In changing environment
Has no preoccupations about work
Can question things
Takes user's perspective

Figure 75. Goals of work – Why-> What -> How.

This diagram of the answers reminds us about how the world we live and work in has changed in a profound way. “The previous world” was stereotypically a technological world. It was stable, manageable by doing the same things always more accurately and better. Things would change slowly in a cumulating manner. Competences and competence needs evolved linearly. There were big stories and one only truth.

But “the new world” is different. It is a world of values and meaning. There is a business and value viewpoint to everything. Level of thinking has often risen from engineering to real product development. The environment is seen as unstable and chaotic. It is also perceived as systemic with various kinds of actors. Thus there is a need to understand wholes and avoid local optimisations. All this is managed by doing new things differently than before. It is a world of big risks. There can be several different realities, with different rules. Organisations can (“may”) find their own style and ways – also in testing.

In the “new world”, the most important question is “why” – that is no longer heard from a distant client, but the answer must be found by oneself, at each level of activity. Based on that one can find out, “what” should be done and finally “how” and “who". Therefore, one must understand business, characteristics of systems, and the needs of various parties – and be able to question old thinking, because everything changes.

The world of systems has changed from technical design to real product development and production of value than matches the needs.

Business and needs are very contextual. The previous world of technology is context free, based on mechanistic paradigms and ideals. One must today identify what is needed and how things pay off to do NOW and HERE. There will be an expansion to what kind of information is searched for – for example A/B testing requires many kinds of new thinking and synergies of competencies.

6.9.5 Business competences

Answers related to business competences are shown in Figure 76.

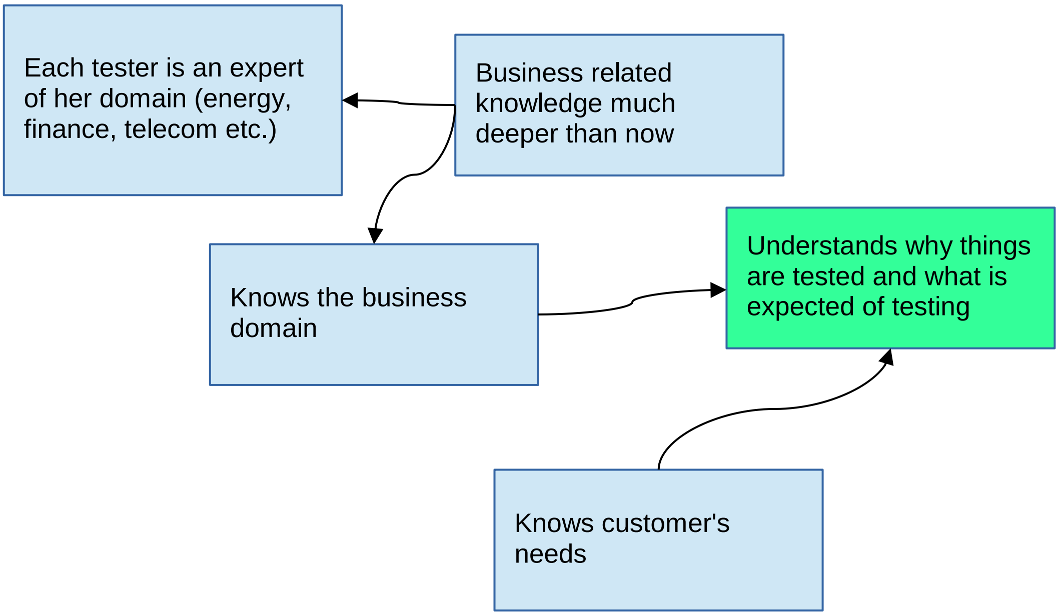

Each tester is an expert of her domain (energy, finance, telecom etc.)
Business related knowledge much deeper than now
Knows the business domain
Understands why things are tested and what is expected of testing
Knows customer's needs

Figure 76. Business knowhow.

Business competence must not be understood wrong. It is not about budget calculations, understanding about product pricing, becoming an economist, throwing away ideas for gaining fast profits or agreeing about everything with product managers. Misunderstandings like that are easy to make, because even the business culture in Finland has been calculation oriented – but that changes too, when the world changes.

Instead, business competence is about understanding about things. One must not be able to do everything business related, but understand the general state of things so that one can support the right things.

6.9.6 Generic ICT competences

Any person working on ICT needs ICT related competences. Answers related to that are mapped in Figure 77.

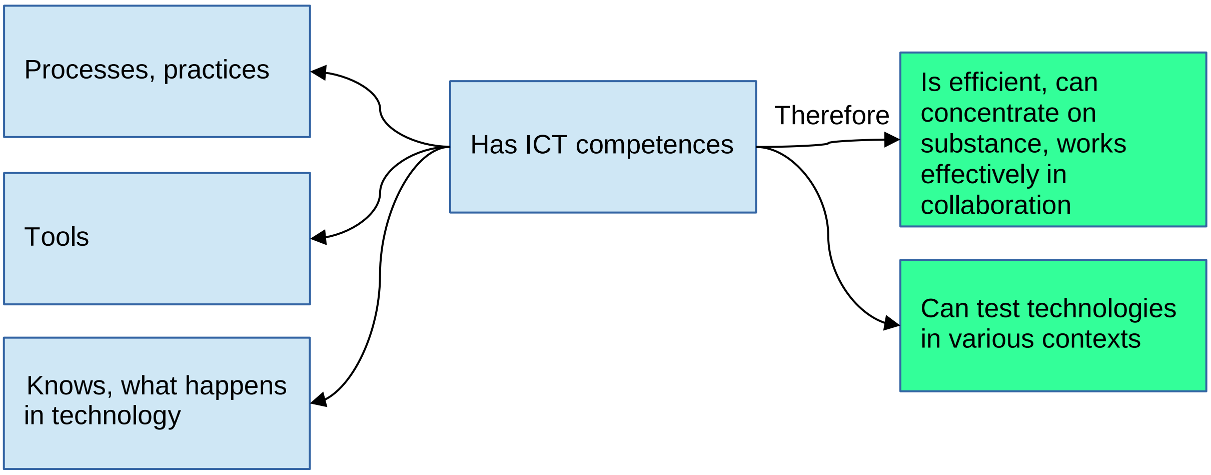

Processes, practices
Tools
Knows, what happens in technology
Has ICT competences
Therefore
Is efficient, can concentrate on substance, works effectively in collaboration
Can test technologies in various contexts

Figure 77. Competence of the ICT world.

The ICT world as such is a competence area, because things in it interleave with what is tested and the operating environment of testing. Challenges are here similar as for the modern software developer. General understanding about systems is needed, because there is a movement to more implicit requirements from thick requirement lists. Therefore, one must understand how computer programs work in general and how they are used and what expectations are attached to them. One must be able to read between the lines – what do the requirements really mean. ICT cultural literacy! In teams, someone must be ICT knowledgeable. Testing is also ICT intensive – communication tools, information management, testing tools one must be able to work with them well.

6.9.7 Effectiveness and efficiency

Answers related to effectiveness and efficiency are mapped in Figure 78 and Figure 79.

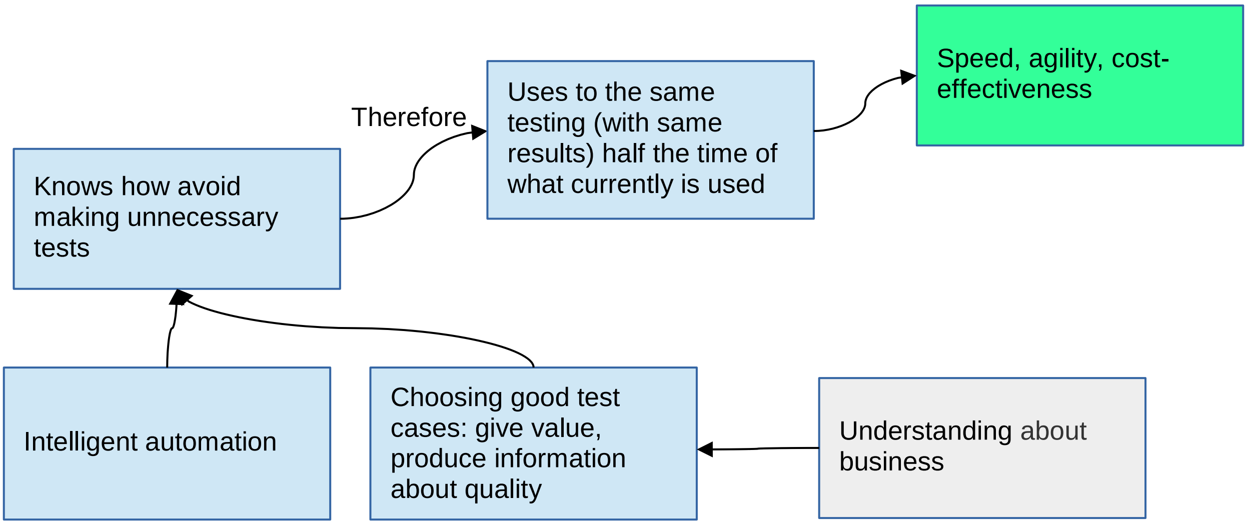

Speed, agility, cost-effectiveness
Therefore
Uses to the same testing (with same results) half the time of what currently is used
Knows how avoid making unnecessary tests
Intelligent automation
Choosing good test cases: give value, produce information about quality
Understanding about business

Figure 78. Effectiveness and efficiency based on making choices.

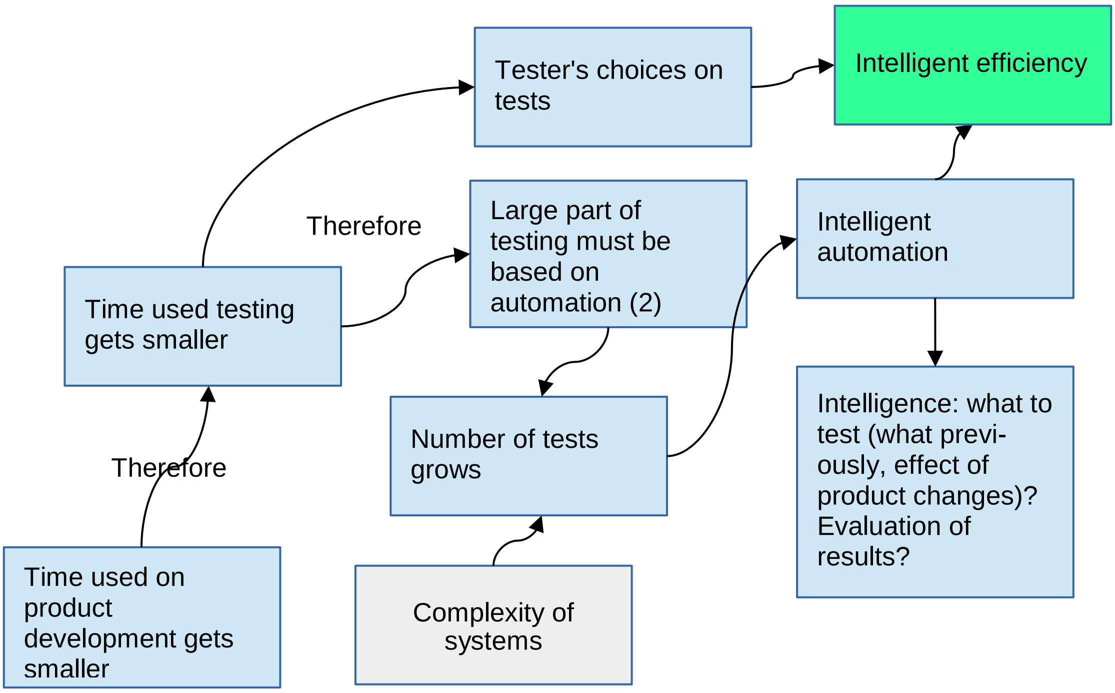

Tester's choices on tests
Intelligent efficiency
Therefore
Large part of testing must be based on automation (2)
Intelligent automation
Time used testing gets smaller
Number of tests grows
Intelligence: what to test (what previ-ously, effect of product changes)? Evaluation of results?
Therefore
Time used on product development gets smaller
Complexity of systems

Figure 79. Intelligent efficiency.

Efficiency has traditionally been sought by aiming to do things faster. The main idea has been to do everything faster during the work day; let's do rapid test automation, even if it were a little worse. Yet it is wiser to do right choices and to focus on important things. Doing less allow for doing the important things properly and doing the whole testing faster. Concentration improves understanding about the entire product. When one concentrates, there is no such hurry. One should emphasise the quality of tests, not their quantity. It depends, however, on the domain and situations, how much for example optimising of test automation is emphasised this regard. Making choices has risen up in all kinds of testing. It is essential that the choices must not leave big holes in testing. Freedom of choice is always easy to misuse. Can anyone be given the freedom? What are the criteria? One element in making choices is saying "NO" to something. Ability to do so is also assumed to be a central starting point in innovation – when it is decided to do something, one does not do something else. Innovativeness is important for our future. In testing one must be very careful with the choices.

Examples of focusing:

- Test big risks rather than small ones.
- Study the unknown more than the known.
- Focus on what is needed RIGHT NOW.
- Prefer good practices instead of bad ones.
- Do new tests rather than repeat old ones.

Don't do:

- Too much anticipation.
- Avoid getting too excited.
- Don't go to unknown – get information.
- Don't shoot oneself in the foot.
- Don't be one-sided.

Remembering:

- Testing is service.
- Testing is not a gate keeper, it does not decide.
- Testing doesn't have a big ego.

6.9.8 Communication skills

Answers related to communication skills are mapped in Figure 80.

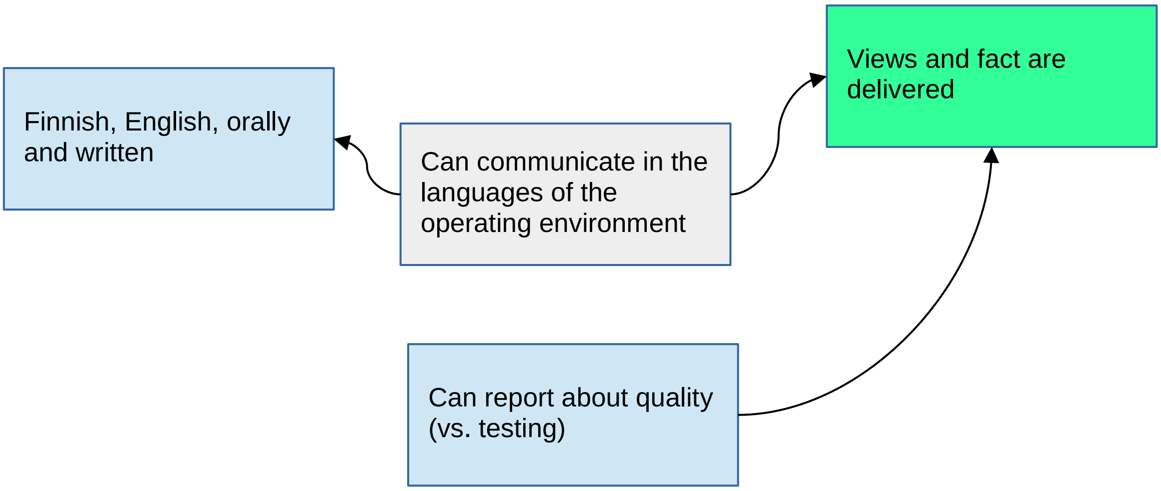

Finnish, English, orally and written
Can communicate in the languages of the operating environment
Views and fact are delivered
Can report about quality (vs. testing)

Figure 80. Communication skills.

The perceived needs for communication skills have evolved during decades. In the 1990's it was emphasised that a tester must be able to write clear test reports and bug reports. In the beginning of 2000's it was understood that a tester must be able to communicate orally in meetings and with the team. Next phase is the realisation that a tester must be able to communicate with the language that the business understands, about the things that are important to business, based on self-found, convincing facts. One must remember that “nobody is interested" in tests and test results, but the reality: what kind of problems do we have? What is their influence? What risks are there? Testing also can’t afford self-satisfied slang.

6.9.9 Gaining competences during education

It was raised up that some testing competence should be given during education (Figure 81).

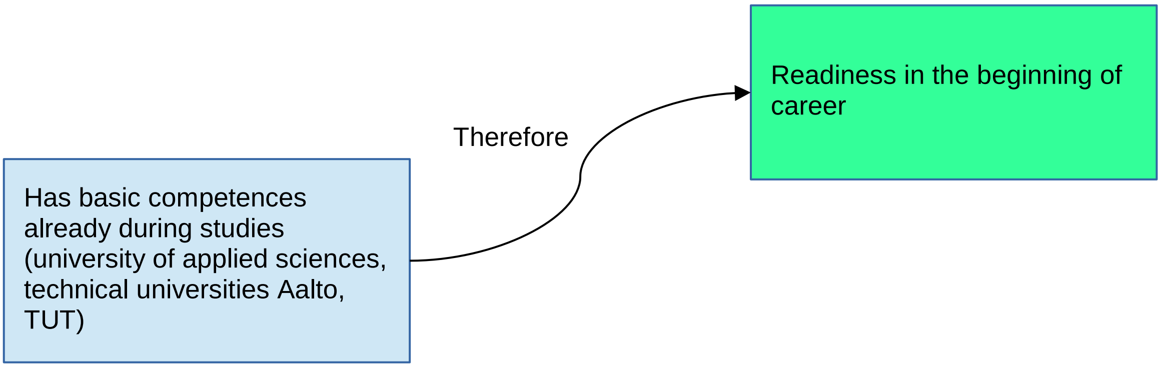

Readiness in the beginning of career
Therefore
Has basic competences already during studies (university of applied sciences, technical universities Aalto, TUT)

Figure 81. Sources of competences.

It is a big thing that more people will get some level of test education already in school and not just during their first jobs. This is already changing the competences in the industry in a positive way and will continue to do so. University level education will produce not only “testers”, but people who understand testing and quality in the product development and business planning.

6.10 Things under test

6.10.1 Central things to test in the future

Answers that were related to the ability to test things that are important in the future are shown in Figure 82.

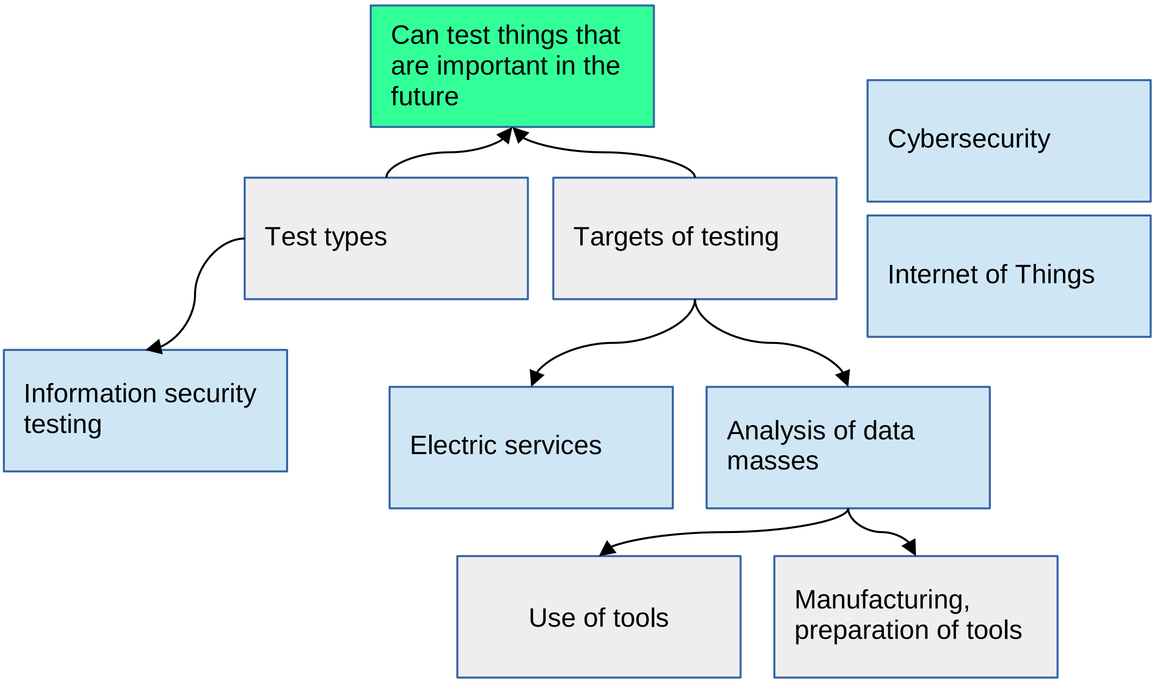

Can test things that are important in the future
Test types
Targets of testing
Cybersecurity
Internet of Things
Information security testing
Electric services
Analysis of data masses
Use of tools
Manufacturing, preparation of tools

Figure 82. Central things to test.

The world of quality characteristics expands continuously. The nature and characteristics of systems change. Traditionally, the acknowledgement of the new things in testing comes much later. It would be better to be proactive in this. Examples of this include human-like robots (Vuori, 2014a), artificial intelligence systems, many implementations of Internet of Things and so on.

The respondents noted all currently hyped areas: information security, cybersecurity, IoT, Big Data and there definitely is a need for professionals that understand those areas.

6.10.2 Nature of the systems that are tested

Figure 83 maps the answers related to the nature of the systems that are tested.

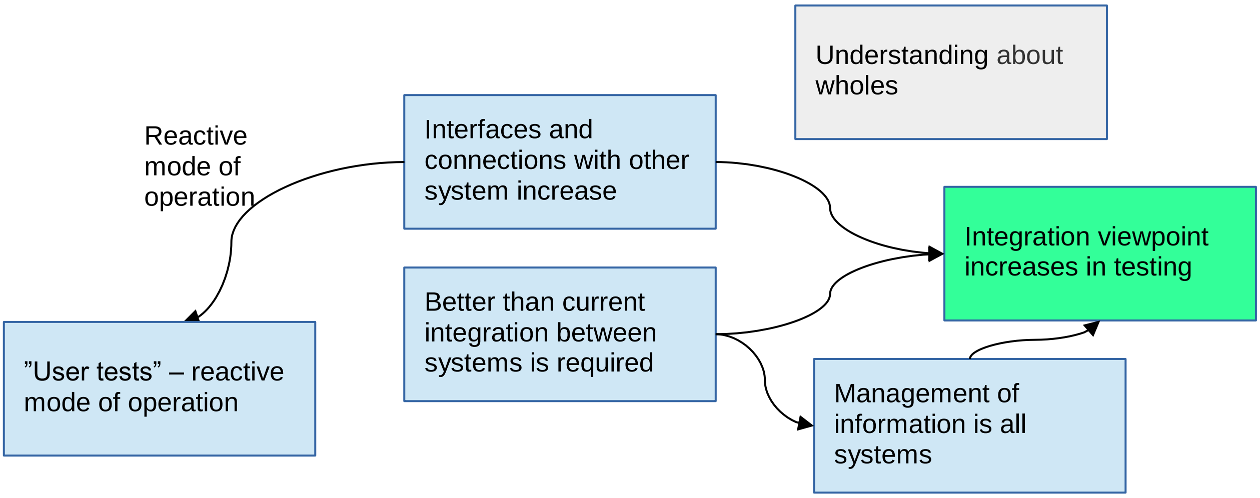

Understanding about wholes
Reactive mode of operation
Interfaces and connections with other system increase
Integration viewpoint increases in testing
Better than current integration between systems is required
”User tests” – reactive mode of operation
Management of information is all systems

Figure 83. Nature of systems that are tested.

Our world is a world of connected systems. Few things to test operate in isolation. When we test some programmatic entity, we also test its relation to other entities – of which we may sometimes know only a little, and that "everything else" can work in any possible way. One must understand technological whole, and even more generally, relationships between things. The wholes are not only complex, but also dangerous. In testing, one must invest in "assuring" robustness – with the assumption that the other system elements can and will do whatever they like. Things that are often missing include systems thinking, assessment of wholes and even the slight paranoia with which the tester protects herself and her choices (the same paranoia that business managers and project managers should have).

6.10.3 Deeper understanding of systems under test

All in all, the changes in the nature of the systems and the things to test will require a deeper understanding of systems under test, some areas of which are mapped in Figure 84.

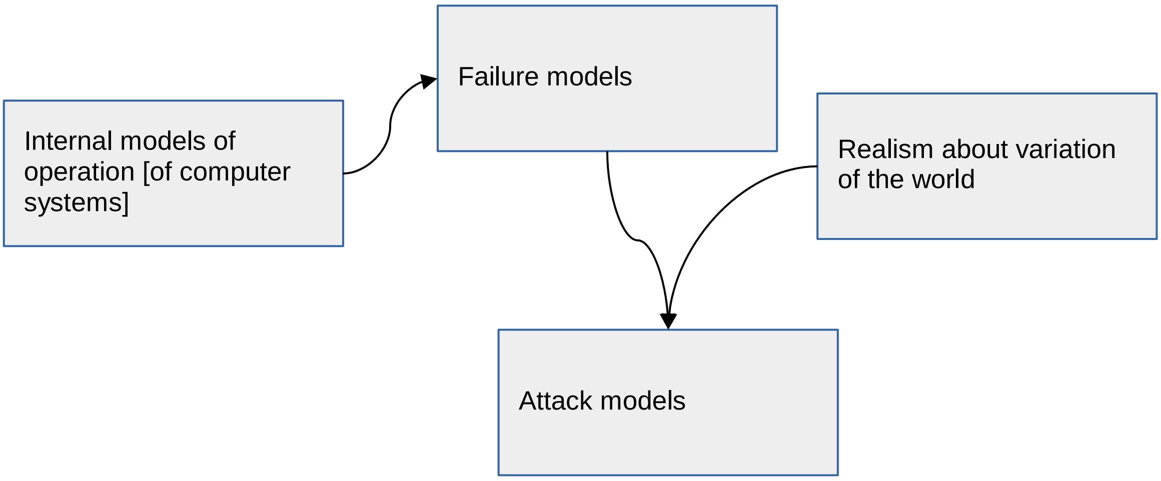

Failure models
Internal models of operation [of computer systems]
Realism about variation of the world
Attack models

Figure 84. Deeper understanding of technology.

“In the old times” testing was categorically divided into black box and white box testing. It has been forgotten that a tester needs to have a mental model of what happens below the bonnet, even when the source code is never seen, for example these kinds of things:

- What generally happens in programs between input and output?
- What happens in operating systems, what things can go wrong?
- How do the networks work? How about virtual machines?
- File formats, character sets.
- What do installation programs do?

The reason for forgetting the need to understand computer systems is in the background: it has been assumed that people have acquired this understanding from their education. Yet many don't have – short prepping would help a lot. Also, understanding about the development activities is important. For example, the business people involved in testing would benefit from understanding how computer programs made (in projects).

6.11 Contexts

6.11.1 Acting in the organisation

Testing is done in an organisation and especially dedicated testers have a role in the organisation. Answers related to that are mapped in Figure 85.

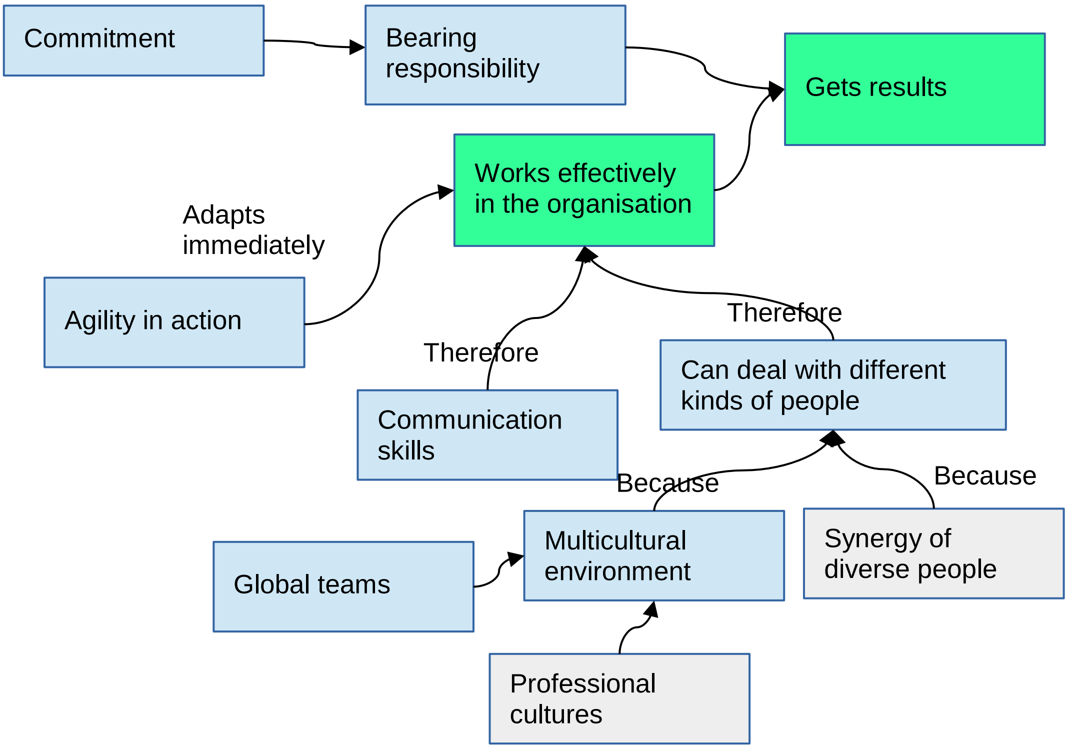


Figure 85. Acting in the organisation.

Many of the challenges of working in an organisation are similar than for other occupational groups, and the respondents noted commitment, communication skills and ability to work in diverse environments as essential. Testers have a special role of bearing responsibility, because testing must, in spite of everything, be a balancing force for various goals of other parties. This is not a contradiction, but a necessary dialogical interaction for common success. Ability to take responsibility is also often thought to be a Finnish strength.

In subcontracting – external or internal – the theme of dependability is often raised. But the changes in the word change the nature of dependability: In subcontracting one must do what has been promised. Yet, in a team one must do what must be done, whether that was explicitly promised or not.

6.11.2 Acting in projects

Projects are the main domain of action. Answers related to acting in projects are mapped in Figure 86.

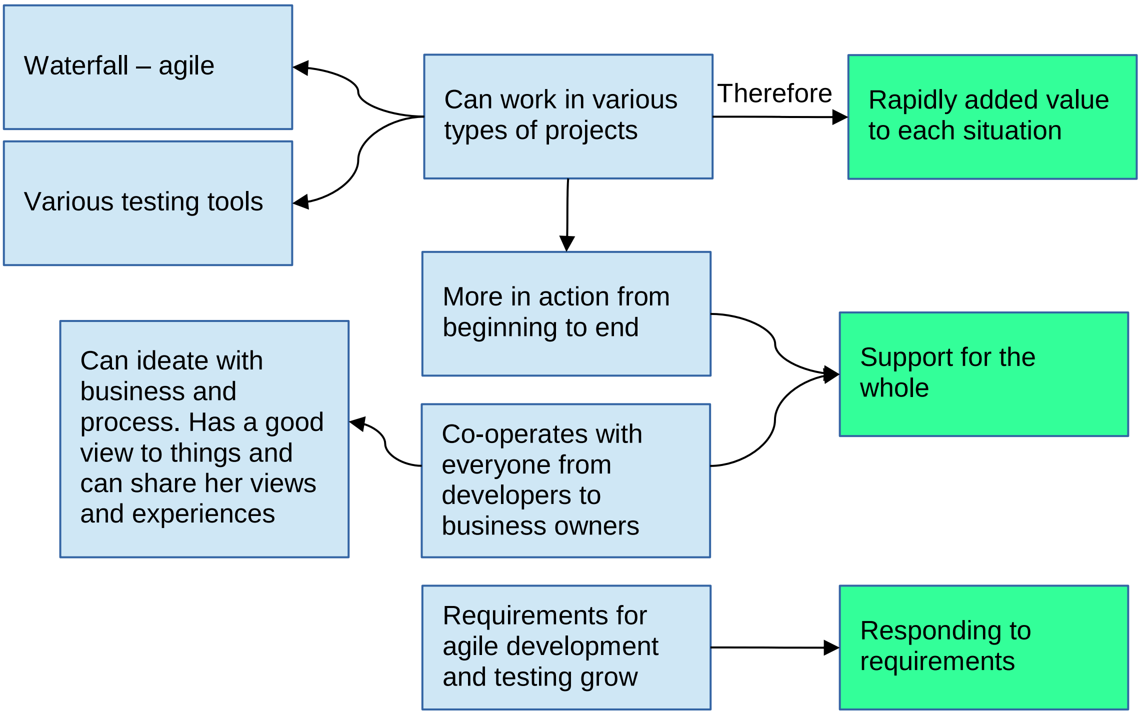

Waterfall – agile
Various testing tools
Can work in various types of projects
Therefore
Rapidly added value to each situation
More in action from beginning to end
Can ideate with business and process. Has a good view to things and can share her views and experiences
Co-operates with everyone from developers to business owners
Support for the whole
Requirements for agile development and testing grow
Responding to requirements

Figure 86. Acting in projects.

Respondents note that there are and will be various project types and testers need to be able to work in any of them. They also see that testers need to be involved during the whole project. This is an old ideal, but still not reality in many cases. That includes being involved in the product and business definition.

A big question is, who maintains organisational “memory” when product managers and developers change all the time? If the tester's role expands it is not only about testing at the end of development, there will finally be a rationale for being in the process during the whole lifecycle. All the time one should not only test but spar, remind, participate in designing, do risk analyses etc. helping the project in many ways. The traditional tester's competence is not sufficient for that. Many new competences are needed in the form of T-shaped competence profile.

6.11.3 Adaptation to contexts

Context change continuously and all actors in organisations and projects need to adapt to the changes. Answers related to this are mapped in Figure 87.

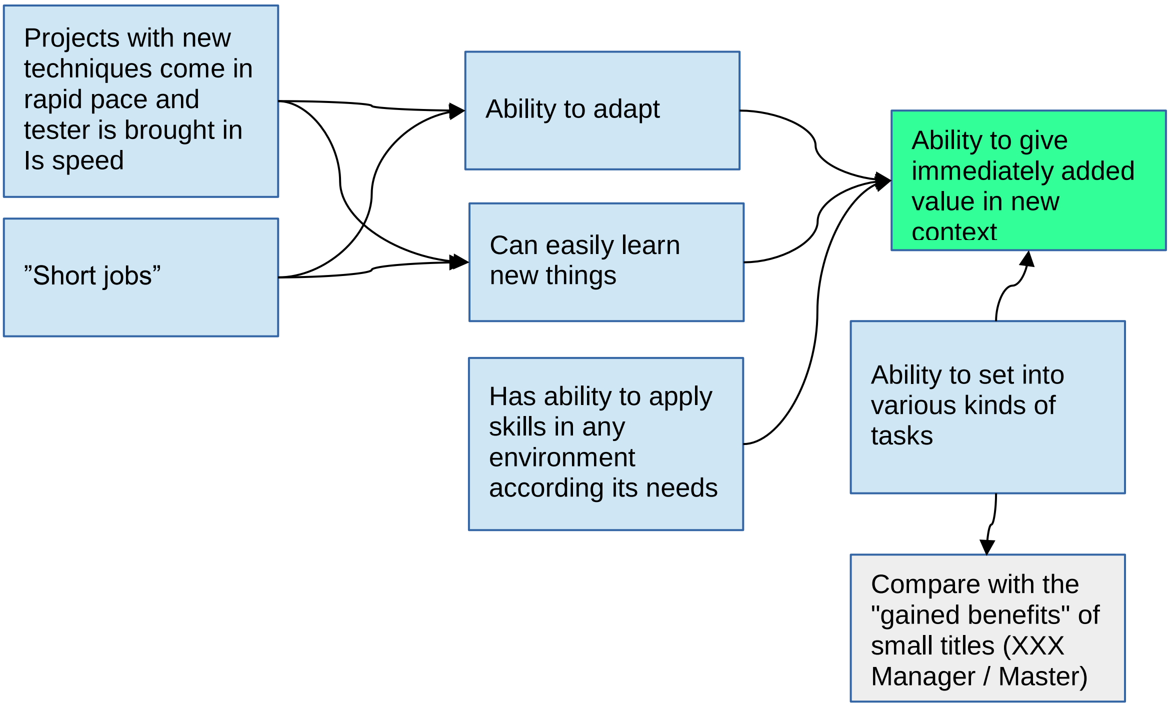

Projects with new techniques come in rapid pace and tester is brought in Is speed
”Short jobs”
Ability to adapt
Can easily learn new things
Has ability to apply skills in any environment according its needs
Ability to give immediately added value in new context
Ability to set into various kinds of tasks
Compare with the "gained benefits" of small titles (XXX Manager / Master)

Figure 87. Adaptation to contexts.

The contexts change even faster: customers' domain, product type, project type, participants and the way of organising the activity. A tester must adapt to new situations immediately and be able to change her own habits into ones that the context needs. For example, if risk analyses have not been done previously, one must realise that one is now needed. Or, change her own style of communicating to the ways of the new context. Needs for change in more general include: Ways of action, roles, goals, methods, collaboration, communication, priorities of testing methods, testing techniques, tools. Essential things here include:

- Ability to abstract competence so that it can be transferred to a new context.
- Understanding about characteristics of systems, mental models of technologies, needs, how people work, business.
- A flexible mind – no fixing to rigid practices.
- Instead of a process approach, a service attitude, a problem solving approach – what is needed in each context and how that need is filled.
- Small ego.
- A large personal mental "toolbox".

6.11.4 Working environments and test environments

Answers related to the working environment are mapped in Figure 88 and answers related to test environments are mapped in Figure 89.

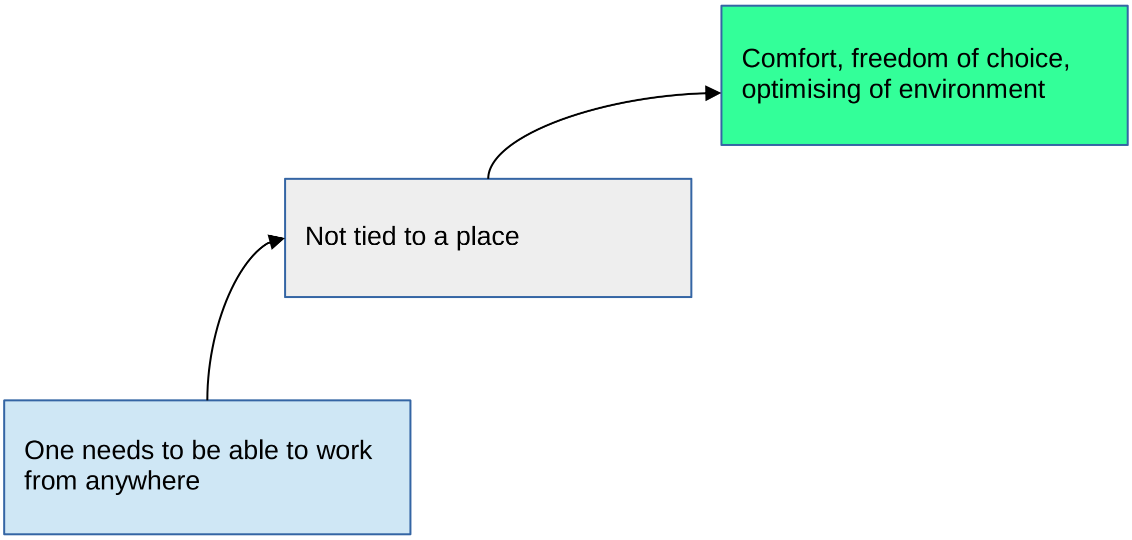

Comfort, freedom of choice, optimising of environment
Not tied to a place
One needs to be able to work from anywhere

Figure 88. Work environments.

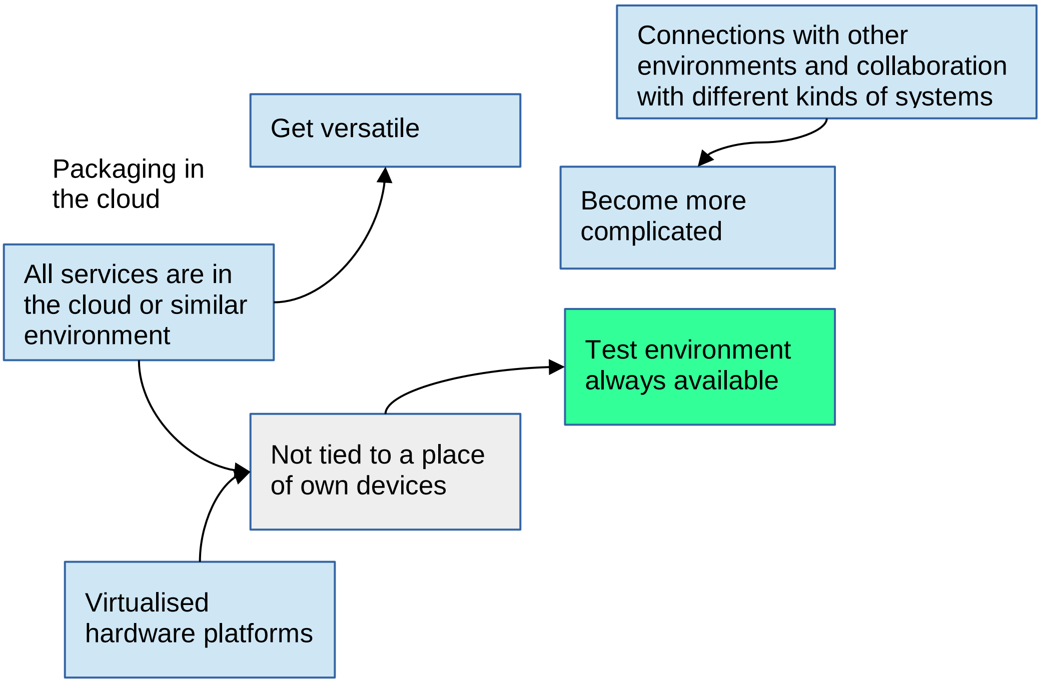

Connections with other environments and collaboration with different kinds of systems
Get versatile
Packaging in the cloud
Become more complicated
All services are in the cloud or similar environment
Test environment always available
Not tied to a place of own devices
Virtualised hardware platforms

Figure 89. Test environments.

According to the respondents, the generic future of work reflects in testing. Remote work has been "future" for a long time. Plenty of it is happening today: Outsourcing is remote work and distributed teams do remote work at team level. Much of the software work is done in virtual and cloud environments and it does not technically matter where they are accessed. There is no longer a need to be in an office to use the hard drive cloning facilities or private local area networks. In general, the world of test environments is changing positively. There are different things and they can be taken into use rapidly, according to what is needed. Managing the new environment types requires competence. When the environments are in shape, people can concentrate on the substance of testing. Will soon "everything" be in the cloud abstracted and virtual?

However, the agile way of working prefers presence with others. Team work is organic work with people. Tacit knowledge and rich communication do not work in electronic communication. But if work is done with headphones on, as is often the case in development teams, people are not that much present in the physical space, yet they could be immersed in a virtual collaboration space.

Digital presence in virtual environments (verbal) has revived and will revive things. How about visual 3D presence?

6.11.5 Nature of companies

The respondents described the nature of companies in the future this way:

- Fully agile.
- They have disciplined action, like Lean.
- There is a random mix of processes and methods
- Their types are fragmented.
- In some areas there is a multi-vendor environment.
- Companies are not human; their practices break people.

Generally, the quality of the work life has continuously improved and it is easy to think that the trend will continue, even though there are economic depressions and other disturbances every now and then. Agility as a trend seems to continue and is maturing to a more real thing also at business level. “Lean” is a very much misunderstood thing, but actually its core is the approach to rethink things to reach goals, to find a unique way of action and acting within it in a sensible, skilled and disciplined way – and improving continuously. A hunt of little things is against Lean!

6.11.6 Testing services in companies

Answers related to testing services in companies are mapped in Figure 90.

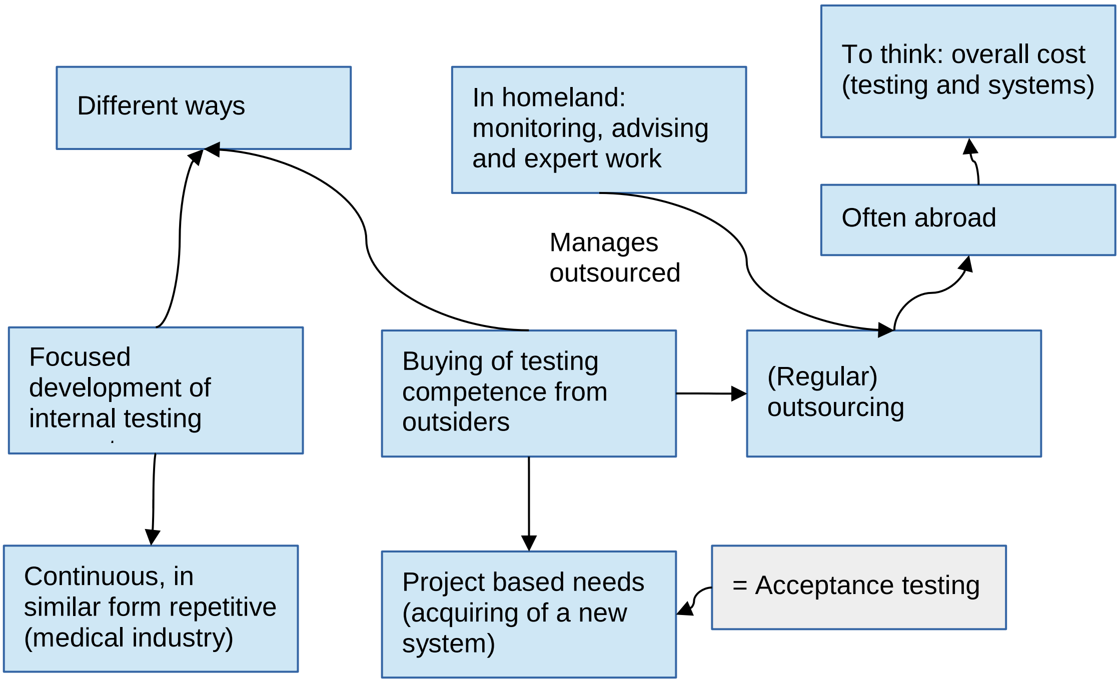

Different ways
In homeland: monitoring, advising and expert work
To think: overall cost (testing and systems)
Often abroad
Manages outsourced
Focused development of internal testing
Buying of testing competence from outsiders
(Regular) outsourcing
Continuous, in similar form repetitive (medical industry)
Project based needs (acquiring of a new system)
= Acceptance testing

Figure 90. Testing services in companies.

Testing arrangements, such as outsourcing, were not mentioned much in the survey responses. It is a world, where there is traditionally some "stutter”. Sometimes outsourcing is preferred in a domain, sometimes it is not. Small companies – and the new start-up culture – need in any case light and rapid external service. Service models that have been built to serve large clients, will not suffice in the future. Time will tell, how the new global testing services change the situation.

6.11.7 Testing service providers

Figure 91 maps the answers related to testing service providers.

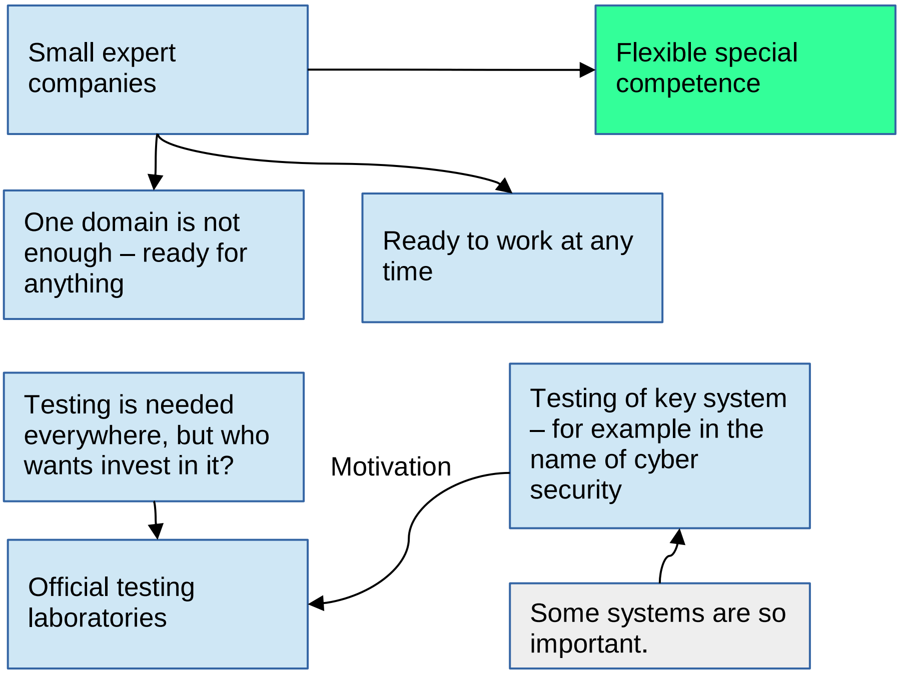

Small expert companies
Flexible special competence
One domain is not enough – ready for anything
Ready to work at any time
Testing is needed everywhere, but who wants invest in it?
Motivation
Testing of key system – for example in the name of cyber security
Official testing laboratories
Some systems are so important.

Figure 91. Testing service providers.

6.12 Generic changes in the environment

About methods and isms

One respondent noted two things:

- New methods are born, with varying life span.
- Number of isms grows

There will always be isms and proposed silver bullets (which they never are). Mature professional must see through them. Yet sometimes they can provide real added value. Yet they are never silver bullets. One must make choices in how to react. One must forget her own ego in all cases. The new ways of thinking can provide lots to learn and when learning one has to leave something old behind, even though that would seem to shrink her own identity and history.

The turning point of society

Overall, the respondents had identified the following changes in society:

- Robotisation: Will robotics also replace managers?

- Will a turning point start, when only a minority will work and the rest has just occasional jobs and citizen salary?
- Testing would be needed everywhere and it is open who wants to invest in it and at what level.
- Will there be new games and applications to devices every day from same companies and the rate of consumption accelerates?
- Data processing is needed and there will be more applications, from cars to refrigerators and wearable technologies.

Note that these are clearly outside the primary scope of the survey, but the answers are interesting nevertheless.

A survey and a dissertation such as this can only handle a subset of the changes in the environment. For example, economic and political changes in the society are somewhat outside the scope of this dissertation and also the expected competence of the author and the respondents. That is why we don't go into these matter more here.

6.13 Overall reflections of the responses

As noted, all respondents had their unique view to the future, which outlines the idea that there are many different areas of product development where the cultures and pressing matter vary. Because of that, any generalisations should be done very carefully. Still, there are plenty of similarities in the answers. The ten top things that were raised in the responses are:

1. Different competence profiles are and will be needed.

2. Understanding about business is critical.

3. Flexibility in contexts and tasks.

4. Holism and multi-skilledness are important.

5. Professionalism, tester's ethics and mind-set are needed.

6. Seeing of wholes in systems, integration thinking.

7. The core competence of testing are still critical.

8. Prioritisation, concentration and making choices are essential skills.

9. The challenges of the changing world which are common for all occupations.

10. In different domains and contexts, the challenges will vary.

7 Evaluation and conclusions

7.1 Time to revise stereotypes

Old stereotypes are strong and we people notice them, not to mention the problems in how the stereotypes restrict our thinking and actions. Testing is still seen as an activity whereby implementations are verified and validated. Mostly the focus has been in verification of how implementations work against their specifications and validation against users' expectations or "real" requirements of business and work has been left in user acceptance testing and in the product validation in safety-critical contexts. Critically, the validation of ideas, not to mention experimentation, have not been seen as a core purpose of using testing.

This stereotype shown by how people talk about testing. They immediately form mental pictures about something that is ideally automated. The context in people's mind is clearly in the implementation domain of development. There are some changes in the mentality. Verification is always activity after some implementation, but designs are implementations of thinking and the sooner they are tested, the better. Among the testing community there is talk about "moving testing to the left", which refers to the development lifecycle and can mean getting system testing done as early as possible, but also testing everything possible as early as possible. But even the position more to the left is still in the traditional context of testing, after the requirement specification and after any free-form concept development.

Of course, there have been exceptions: usability testing has, at least among experts, seen as a tool for validating user interface ideas and to find out information about users' behaviour in some pre-defined situations. Proof of concept testing, on the technical domain, is all about creating a controlled experiment.

Testing is much about handling risks of product development. The view of testing as low-level activity against implementations mean that often there is no relation to product level risks: desirability, usability etc., but the risks handled are low-level, isolated risks of some individual function not working. This is visualized in international testing literature too, as even Kaner & Fiedler's (2016) test design workbook discusses risks at this level (although, positively, reminding the audience that one should always find out about the users and the business context before testing to understand what is expected about the software). In the safety-critical domain, the risks are assessed at the product level, but again the view is into functional characteristics of the system. There should be a clear reflection of the business risks into all testing. At the lowest level of testing there may be a risk that some value will overflow the data structure or variable type is should be assigned to. That is relevant to creating robust systems, but the interesting link to business level risks may get hidden. Obviously, test cases should be linked to high level specifications and risks too, but in practice the link can be weak at operational and at mental level.

One part of the stereotype problem is the view into the systems under development. Functional testing mostly sees the system under test as a processor into which inputs are fed through some "filter" that should omit erroneous inputs, and outputs are generated. This is perfectly good in many situations, but makes testing focus on single element of activity and perhaps fail to see the overall situation. Exploratory testing often emphasises another system view, where the system under test is a collection of many elements and testing should expect any action on the system to produce a change of behaviour anywhere in the system. This is not just a methodological viewpoint, but a different view into how technical system work and what is their essential nature. In practice, it is good to utilise different views and it is harmful to use only one stereotypical paradigm guiding testing.

There has been a tendency to support "one best view" about what testing is, but gradually there are signs that diversity is respected more. That means people with different ideas about testing, people who are proficient with different approaches and have different education, training and certifications. More viewpoints are better in testing and they also bring dynamism that will help testing evolve.

There are historical reasons for that, in the history on testing and the cultural emphasis on systematic engineering design. History has, though, temporal nature and things and thoughts change. The engineering level testing is absolutely critical, but should not be a narrow, limiting stereotype.

Until the stereotypes and mental models about testing are changed, our actions in product development will not change.

7.2 Evolution of testing

To further warm up to the evaluation of what has been found out in the dissertation, we for need to remind us that there is quite a lot to do in testing and for the quality during the development and acquisition of systems. Sometimes people has a misconception that testing is just simple test case creation and execution, perhaps all done automatically. But that is just a tip of the iceberg, so to speak. Table 50 lists a selection of testing related activities under domains of activity. The domains are not phases in a project, but purposeful contexts during the lifecycle. They may even be domains of mental approach more than visible, specific activities. Many of them are deeply integrated in the development activities and manifest themselves in ways that depend on the particular lifecycle models or processes used. Furthermore, what needs to be done or should be done in a project, classically depend on the criticality of the project – trivial and small projects should be executed in simple ways, whereas large and safety or business critical projects require more methodological and disciplined action.

Table 50. Selection of testing related activities during a system’s lifecycle.

Domain of activity	Examples of testing	Examples of activities that supports testing
Innovation and concept development domain	Prototype tests (rough) Minimum Viable Product experiments/tests	User / customer studies Defining competing products for “baseline” Project vision, goals Project concept level risk analysis Other product analyses
Activity planning	Plan reviews	Process definitions Quality policy Testing infrastructure Information system infrastructure Role and responsibility specifications Outsourcing arrangements
Management domain		Work management Development management Quality management Test management Defect management

Domain of activity	Examples of testing	Examples of activities that supports testing
Collaboration and general work domain		Testing role specifications Backlog prioritisation Reviews Integration of testing in workflows Definition of “done” Proactive quality work
Requirements gathering	Testing of competitors’ products Usability testing of existing product	Analysis of competitors’ products Prioritisation of requirements Acceptance criteria formulation Review or requirements Requirements management
Design domain	Prototype tests of various fidelity Proof of concept tests for new technology Preference tests A/B tests	Prioritisation of features Hypothesis building Testability review Reliability analysis Safety analysis Security analysis Design reviews Architecture analysis Interaction / environment models Status information for features, components Design documentation
Implementation and testing domain	Functional testing at various levels (unit, integration, system, system integration) -ility tests (usability, user experience, security, performance, stress, interoperability, co-existence) Configuration tests Regression tests Comparison tests with competitors	Design models Implementation review Code review Implementation documentation Test management Test generation Test optimisation

Domain of activity	Examples of testing	Examples of activities that supports testing
Deployment domain	Release tests Tests in deployment pipeline Tests for the deployment infrastructure Pilot tests, alpha & beta tests	Configuration management
Acceptance domain	User acceptance tests ICT acceptance tests	Test environments
Usage domain	Health tests of system in use Problem analysis tests Other tests for system in use	Monitoring Data collection
Maintenance domain	Validation tests for defects Testing for repairs Testing for new / changed features Regression testing	Impact analysis
Learning domain		Communication Lessons learned Competence development and transfer Training

There clearly are potentially lots of things to do. And the set is dynamic. It has evolved through decades. The history of testing started in the mechanical era and the ideas and principles have evolved through many generations when the understanding of testing and the needs for it have changed. The evolution is visualised in Figure 92.

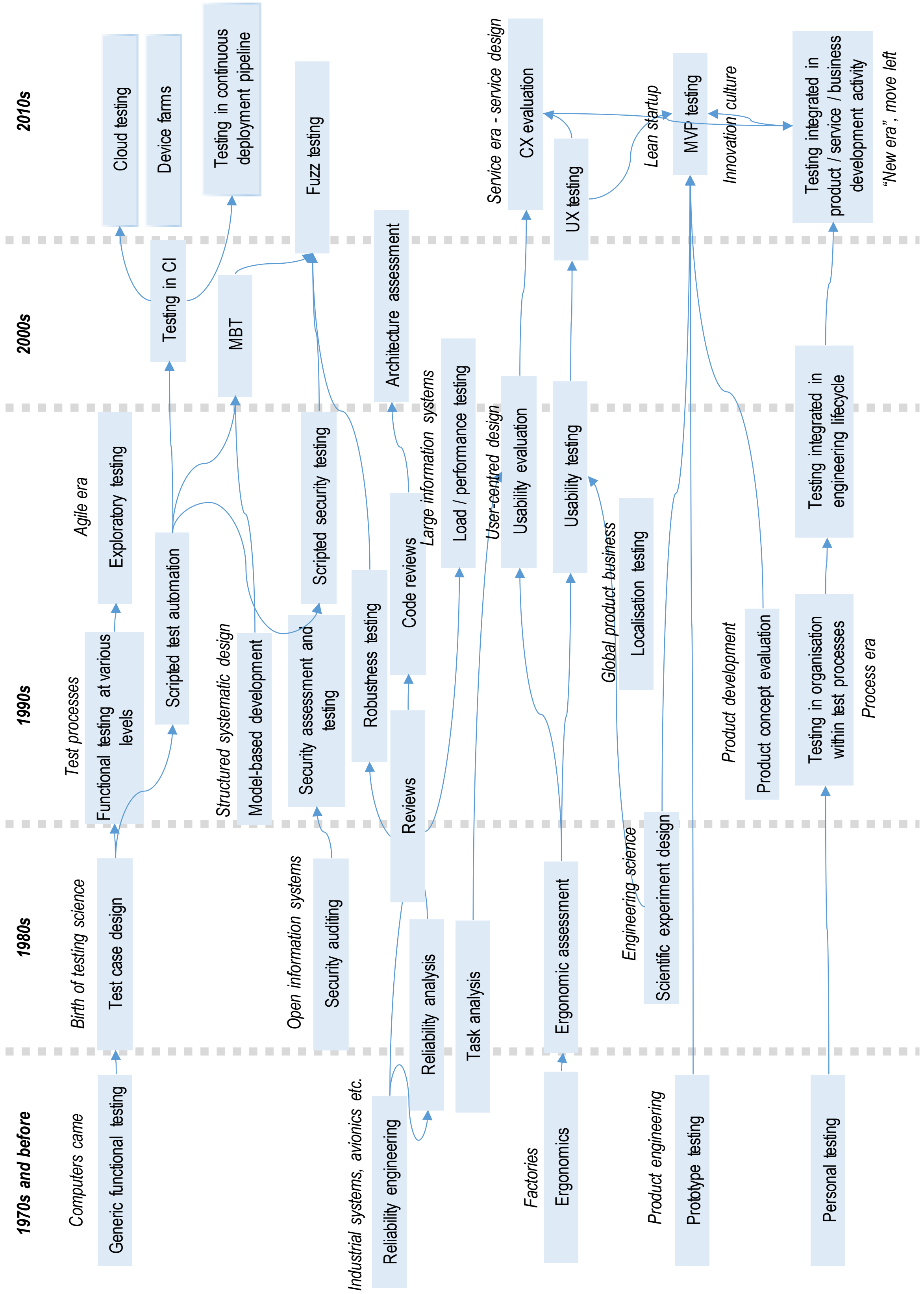


Figure 92. Evolution of testing (simplified).

The figure visualises how even the new activities have their roots deep in the history. New test methods are just part of evolution that manifests itself when the time is ripe. Rarely do methods or actions appear from emptiness, but they often combine elements from one or two existing things. Sometimes the preceding practices disappear, but most often they remain alive, just with reduced priority and resource usage.

7.3 Creating opportunities to evolve

One significant problem is that the number of testing related activities has grown, but the resources in companies have not. That means that somehow testing needs to spend less time on some activities to make room for the new things. Many companies in many occasions have stated that test automation is a way to spend less time on routines and let testers focus on important issues. Test automation is very helpful in regression testing that used to tie up whole teams doing repetitive task for every build. Now that can largely be automated (being careful to do the automation so that its maintenance does not become a new resource hog) and testers can focus for example on exploratory functional testing, but also on testing that responds to the new focus areas. This is also one reason why companies need a balanced set of competences. Good, effective and maintainable test automation requires skills, as does effective assessment of product concepts.

Obviously, on the software development process side, there are agendas such as Lean that aims at making the overall process more efficient and effective so that less effort is used on things that have no value, but can even be called "waste". Similarly, cloud computing and virtualisation – and testing as part of them – are efforts to reduce time used on infrastructure management, which can then be used on something productive.[29] And perhaps most importantly, agile development in any of its forms should help companies to develop only as much that they can manage. Overall the situation has gotten much better than on pre-agile days.

Yet, there is a need for conscious planning of how to use resources effectively and the main approaches are:

- Using of well-designed test automation for regression testing and other repetitive tasks, including test infrastructure management. Cloud-based and virtualisation techniques should be considered.

[29] To visualise that, testers used to, at the change of century, spend plenty of time formatting hard drives, copying drive images and such, which is now often replaced by the creation of a virtual machine in a private or public cloud in just a couple of seconds.

- Consideration for testability of the product and its technologies is essential for efficient and effective automated and manual testing.
- Risk-based testing and test prioritisation for focusing on the issues with high risks the most.
- Consideration of the company's life cycle when planning testing. Startups should focus on user and customer experience and develop any technical testing infrastructure only after they understand their product and the workflow needs in its development.
- Even though testing has been more and more integrated into the development teams' daily work, getting external help as needed is recommendable.
- Focusing on the development of a good product / system concept helps in keeping it simple and more stable, reducing overall work in all development and testing activities. Assessments of product concepts are critical for this.

Saving resources in tasks only does that. It does not mean that the saved time is used on anything important. It could only be used for being faster with current practices and mindsets.

Process development is traditionally used for creating opportunities for better activities. For example, if we see that assessment of product concepts is critical, there should be a phase in development where concepts are developed and during the development, properly assessed. That is not necessarily the case even in industrial design -driven new product development, not to mention information system development. Companies seem to need buzzword to even try such traditional approaches to design, and at this time, the buzzword is Lean Startup and its Minimum Viable Products. Buzzwords come and go and to keep momentum after them, real design approaches and competences are needed.

Process development is one area that requires most of all orientation from directors, managers and development personnel who "own" the processes. There needs to be awareness of the elements of quality of the overall product and the practices needed.

7.4 General changes in testing and quality assurance

7.4.1 Environmental changes

After all the detailed analyses we can now afford to make summary of how Finland has changed. It is done here by presenting the “stated of the nation” as it was pre-2010 and now and in the near future. Caricatures help us see the big picture from the details and build a holistic mental model of the situation.

Pre-2010, from the software product and systems development perspective, Finland was centred around large companies and working in their ecosystem. Innovation was tightly located in one place and flowed from top down to the contractors as definitions of product specifications and for work. Companies either used contractors or were one.

Everything was done by processes and process thinking which had evolved through the years. Quality was managed by reviews, developer testing and QA testing for product versions every couple of months. Testing at system level was scripted manual testing and test automation. Work was done with proprietary, expensive tools that only the experts had access to, in closed infrastructure.

There were ideals for occupations that had high level of competence in their disciplines, but nothing more. The expected competence profiles were quite I-shaped. If there was a task, there were specialists and managers for it. All in all, everything was managed and stable – and perceived as good.

Post-2010 the era of giants ended, as symbolised by the end of Nokia as mobile phone vendor. The new economy consists of smaller companies, in a similar profile as in other countries. The new companies are leaner (in the general meaning of the word) and thus competence profiles need to be more T-shaped. There are less dedicated testers, which is also influenced by the rise of more and more test automation due to continuous integration. Therefore, testing needs to be integrated into lean (in the general meaning of the word) organisations and practices and needs to be understood fully so that the organisations and activities are effective. There is a need for shared understanding of quality and testing and for competences that can adapt fast to changes. And changes there will be: startups who are finding their identity and products change in focus and also professionalism when turning into the growth phase; professionals change their companies more and more often; technologies and concepts are changed rapidly when opportunities arise.

The work is still done around disciplined processes, but this time they are agile, not waterfall-like or evolutionary as previously. Much of the work is more distributed, often done in open source repositories using open source tools that anyone can have at their disposal. The environment for good, effective engineering is excellent and supported for huge growth of testing skills.

But now the new companies need to find their place in the market – home market and global market. That requires innovation and movement away from the engineering culture and into real product development culture. The companies need to think for themselves. Testing offers new opportunities here: it will not only pinpoint defects, but can support in assessing concepts and their good characteristics – and experimentation is the very domain of testing. Testing as an activity has blended into every business process. While testing is often automated, exploratory testing has released the brainpower of testers to goof, effective use. At the same time, technology advances and products become more complex and critical – both for security and business risks and for user experience. That requires new level of professionalism and discipline for the business and product risks to be managed.

How about the next steps? The movement around 2010 was radical, because there was so much pent-up energy released. When rigid mental and operational structure collapse, the change can be violent. As there is new dynamism in the environment, the next changes will likely be more incremental. There is no basis for even trying to think of concrete, independent scenarios at this level of analysis. We can only look into the emerging issues that we see having potential and try to implement them further – trying to make the changes into opportunities. Those depend on a particular context, but we can see some generic issues, such as:

- Blending of testing into business and product definition processes and innovation in general. That should be supported as much as possible.
- Transformations of industries. Transformations are risky and testing needs to support them by providing information about the changes and innovations.
- Rising experimentation culture. We can advance that and more it from somewhat ad-hoc style into professional, valid activity (while not sacrificing speed).
- Building T-shaped professionals that can test, participate in innovation and any practical needs a small company or unit has.
- Support the growth of startups by teaching them tackle the right quality-related issues at any phase of their lifecycle.
- Building process-independent competences that help new companies find their unique way of action. This implies solid core competences that can be used in any context and situation, combined with contextual awareness.

As a summary, the main differences in the caricatures of the time periods can be summarised in

Table 51. Comparison of the main defining characteristics of testing in caricature of three periods.

Element	Pre 2010	Post 2010	Desired near future
Flow of innovation and work	From large clients to contractors who do designs and implementations based on client's requirements and context	In SMEs independent development of platforms, products Emerging startup culture	In SMEs independent development of platforms, products Mature startup culture
Environment	Basically stable	Dynamic	Dynamic

Element	Pre 2010	Post 2010	Desired near future
Product development lifecycle	Waterfall-like, with multiple internal deliveries	Agile	Companies find their own style to support their unique approach Mixed models Concept development stronger
Quality assurance	QA organisation, QA processes, assurance of designs and implementations	Integrated with development, assurance of implementations	Assurance of concepts emphasised
Quality paradigm	Technical quality	Customer centred quality	Holistic: Business, customer, user, technology
Ideals	Planning, management, control, efficiency	Agility, flexibility, speed, effectiveness	Innovation, agility, flexibility, speed, effectiveness
Positioning of testing in processes	Reviews pre-development, unit / development testing, QA testing	Unit to system testing during sprints, QA testing	Concept testing pre-development, unit to system during sprints, QA testing
Test rhythm	Test rounds every couple of months	Continuous	Varies, continuous in implementation
Test basis	Specification-based	Design & implementation based	Success factors, risks, requirements
Goals of testing	Finding defects	Providing timely information about quality, allowing deployments	Sensemaking, understanding concepts, comparison, providing timely information about quality, allowing deployments
Goals in startup companies	Make new innovation solid for selling	Help in finding company’s focus and make product and processes solid in growth phase	Help in finding company’s focus and make product and processes solid in growth phase
Who tests	Dedicated testers, often in separate teams	Developers, sometimes a tester in team	Varies: in every context someone who knows how to; access to expertise essential
Role of testing	Process phase, occupation of some	Process task, part of “done”	Integrated element of tasks, processes, development phases

Element	Pre 2010	Post 2010	Desired near future
Testing style	Tightly scripted manual and automation	Test automation, exploratory testing	Managed experimentation, exploratory testing, test automation
Critical test types	Functional testing, performance testing	Functional testing, security testing, stress testing	Concept / idea validation, UX testing, functional testing, security testing, stress testing
Biggest risks	Schedule risks Budget risks Key person risks	Technological risks Market risks	Security risks Market risks
Challenges in testing	Large batches of new functionality	Fitting testing in sprints, technical debt, new technology	Complexity and diversity of systems, new technology, new concepts and domains
Ideal tester	Focused, methodological expert, independent of developers, follows orders	Working tightly with developers, has iniative	Understands business and user, multi-skilled, independent team player
Main testing related skills	Test methods, accuracy, reporting	Test methods, team skills, reporting, use of tools	Test analysis, risk analysis, use of tools, communication skills
Commonness of testing competences	Few trained testers of developers, functional testing mostly	Majority of developers and testers have some training or education, functional testing mostly	Everyone should have basic knowledge; more understanding about user experience, security and experiment design
Learning areas for testers	Doing testing, test design, test automation of test management,	Doing testing, exploratory testing, test automation, agile development, scripting	All types of testing with some specialisation, test automation and deployment, product development, business, experiment design
Testing tools	Proprietary, expensive, dedicated users	Open source, free, anyone can use	Open source, free, anyone can use

Element	Pre 2010	Post 2010	Desired near future
Technology	Single source, domain specific technologies, in-house development	Mashup of 3rd party components, generic technologies	Mashup of 3rd party components, generic technologies

7.4.2 Towards a new cultural phase

If there really is a deep change happening in the society and in the environment, it should clearly show in the elements of culture. Schein (2004) proposes that:

“The culture of a group can now be defined as a pattern of shared basic assumptions that was learned by a group as it solved its problems of external adaptation and internal integration, that has worked well enough to be considered valid and, therefore, to be taught to new members as the correct way to perceive, think, and feel in relation to those problems.”

Culture is a way to share the reality with others. It is the way to think about things, to act, to be together as an organisation, as the sum of the remains of thousands of experiences.

We have analysed the changes in thinking and actions a lot, but have not yet tackled culture as a separate issue. That is because culture “pulls together” all the other changes and summarises them in the level of meanings. It is also commonly said that “culture eats strategy for breakfast”, meaning that the quality of our actions is defined by culture instead of the purposed ways of action. Culture is obviously present in all contexts and their models (especially the triangle model in action research is very much a model of an “activity culture”) but in the analysis of those it becomes too diluted and loses its significance.

Therefore, it is now time to look at the changes at the level of culture.

Cultural aspects are often separated into levels, from actual practices to shared thinking patterns and here we do the same. There are frameworks for the levels, but here a level system is used that is tailored to this situation. Using it, we outline the characteristics of the “previous time”, now and potential near future, thus gaining a continuum of change. Of course it is now understood that cultures may vary. Industrial domains can have a very different culture than entertainment or communications domains and companies in the same domain can have quite different cultures. An assessment will necessarily be just a very rough generalisation and visualisation only. But as such it can show the deeper undertones in what is happening. One such visualisation is in Table 52.

Table 52. Comparison of the cultural elements in three periods. Illustrative generalisation.

Element	Pre 2010	Post 2010	Desired near future
Symbolic level			
Terms that carry meaning	Quality management system Process Defect Coverage Validation and verification Cost	Technical debt Automation Speed, velocity Value Waste Productivity	Value Validation Learning Risk
Rituals	Review Test round	Add issue to backlog	Experiment Celebration of success
Cultural artefacts	Instruction Test documentation Test case Certificate Quality policy Process chart	Backlog Test script Constant integration	Test / experimentation setting
Values	Professionalism Maturity Long-term success	Agile values (Agile Manifesto)	Agility Innovativeness Success Clan, adhocracy values
Action level			
"Things people do"	Plan, act, review	Act, review	Plan, act, reflect
Allocation of tasks based on	Occupation Role in company	Occupation Role in team	Competence Role in team
Team ethnography	Homogeneous teams for different activities	Heterogeneous teams	Teams diverse on many dimensions
Role of customer and user	Necessary evil Pays the wages	Partner in the process Someone to satisfy	Object of understanding Someone to satisfy
Formers of shared experience	Project work Finding defects	Project work Finding defects	Collaboration Experiment

Element	Pre 2010	Post 2010	Desired near future
Introduction to culture	Reading instructions Training Observation	Collaboration Observation Reflection	Collaboration Observation Reflection Dialogue
How do we define the right product	By careful analysis	Learning by building things gradually	By experiments By building things gradually
Things to work on	Requirement Specification	Story Requirement	Opportunity Idea
Assumptions			
Nature of the world	Managed, stable, defined by "big stories"	Partly managed / understood, partly complex and complicated, changing	Chaotic, complex and complicated; in constant change; self-defined
Nature of humans	Rational, objective	Rational, subjective, social	Irrational, subjective, social
What is "quality"	Technical characteristic Defined by standards	Measure of value Defined by customer	Measure of value Measure of opportunity Defined by many parties
Assumptions about how quality is produced	Adherence to process Controlling technology Best practices	Collaboration with customer Teamwork Control technical debt Best practices	Understanding customers Design thinking Controlling risks Practices suited to needs
Work should be	Systematic Efficient Conforming	Agile Efficient Effective Fast	Intelligent Agile Fast Effective
Ideal programmer	Systematic engineer Specialist	Craftsman Generalist Full-stack developer	Full-stack developer Full-quality developer Collaborator Tester
Testing should be	Routine Specification-based	Automatable	Automatable Intellectual Context-dependent

Element	Pre 2010	Post 2010	Desired near future
Assumptions about own possibilities to act	Role-defined Role-limited	Anything is possible in local context	Anything is possible
Expected tester’s ethic	Accuracy Effectiveness	Helping others be effective Helping business	Helping others assess ideas Helping business Controlling risk

There are all signs that we are moving into a new cultural era in software product and systems development. As always, the previous eras are in some form present and the new is emerging so gradually that it is hard to detect and can clearly be seen only afterwards. But the cultural changes are deep. We know that they emerge from the experience from the previous phases and thus are based on real situations and are not imaginary or wishes. Note that small “ripples”, for example new methodology fads, are only cultural as artefacts that represent something more stable. When we find the deeper level changes, they give a platform on which to build the future. In this case, they give us more proof that there is a need to change, and a rationale for any proposed changes.

After these preliminaries we can again turn to the practical issues.

7.4.3 Changes in product development phases

After the analyses we can see several general changes in testing and quality assurance. Here we look into those from the viewpoint of creating good products in three practical main activities in product development:

1) Concept development. New products need good concepts, ideas that beat the competition with their perceived value.

2) Development – the phases of activity where concepts are turned into solid products.

3) Deployment – the activities for rapidly, but controllably letting customers and users start using the products or their new versions.

Concept development

This phase is about assessing ideas, prototypes, demonstrators. Those are done by experiments, that include Lean Startup practices, common prototyping and so on. The goal is to test customer reactions, customer behaviour, satisfaction and feedback for the ideas. Good experiments are valued here, because experiments need to be proper to have any value. Note that good doesn't mean that the experiments would not be rapid. User experience testing is another view into this phase. All in all, this is an area where testing does not concentrate on finding defects, but getting any information about the product's quality and generally making sense of the context if the domain or concept are very unique. At concept phase, errors are found more by analysis of the concept: does it have any weak point. Analysis and testing should usually be combined, as either by itself can produce a lacking basis for decision making.

Overall, at this phase there is a need for testing competences that are able assess the product ideas for business, customer and user values and product reliable information about those. Moving focus of testing is what is sometimes called “moving to left”.

In general, testing is more experimenting than validating or verifying and as the experimenting culture advances, proper experiment design and execution competences are of great importance. The learnings from user experience and usability testing are valuable here

Development

During development, ideas are turned into designs and designs into implementations. The work is done at two levels: low level activities in the software developers' tasks and at the level of user and customer experience. The low level activities are the domain of traditional testing activities with automatic and manual testing. This is an area of change and changes benefit from exploratory testing before test automation. The testing at low level aims at keeping the system solid – indeed, solid so that it can tolerate changing. But at the same time there is the higher level of customer and user experience that demands another view. All new features should be assessed not only for their technical robustness, but also for their value for the product: how usable, pleasurable and desirable they are by themselves and how they influence the product in these respects. This includes user experience and usability testing and also A/B testing. All of those utilise experimentation competences, but A/B testing requires also expert level configuration management and deployment skills. Note that all those skills just need to be available for the team, but not in the same person.

Information security analysis is essential at this stage for all types of products. Everyone should understand the need for security and some people need to be able to do professional security analysis and testing. For innovative "embedded" systems, safety has a similar role.

For the products that are by nature "services", there needs to be skills for validating all areas of service design.

Overall, at this phase there is a need for "technical" testing competences and user-oriented competences. Excellent test automation skills and exploratory skills complement each other.

Deployment

Especially for information system products, deployment has risen to a critical phase because fast and easy deployment can make value provision faster and allow for A/B testing and similar. Automated deployment utilises mostly automated tests to validate that a version or a change can be deployed, but in many cases, exploratory testing of the changes is needed too, as is user experience assessments, but here we locate those to the development phase.

7.5 Most essential competences in relation to the activity system

7.5.1 General

This thesis is expected to produce a view about the most important competences. But for that a warning is in order. System thinking reminds us that a system requires all kinds of elements and while their role might not seem as having a high profile, they are necessary. Consider four-colour printing of magazines. We could look at how much inks of different colours are used for printing photographs and see that magenta and cyan are used much more than yellow. Yet, without yellow the photographs would look awful. Thus, saying that it is of less importance than the other primary colours would be a grave error. In the same sense, there are competences that look like having a side role, but they may be something that keeps the whole together. So we need to be very careful about them. Of course, in some cases we can say that competences related to technologies or practices that have disappeared are not important at all, but even those may have been blended in other competences and still exist in that form. In fact, this emphasises one very essential competence: the ability to understand contexts and business situations and to identify what is critical in them and then plan and act accordingly.

A full list of competences based on the activity system -based competence architecture is included as appendices 4-7 which are sorted by various criteria. They form a sort of database in which to assess the whole range of competences.

In this chapter, we shall show just glimpses of the most essential competences and reflect their relation to the activity system and the change vectors analysed in the preceding chapter.

7.5.2 Self

Contextual sensitivity. This is a broad issues and relates also to other elements of the activity system. As contexts, requirements and teams are all dynamic, there is a need for sensitivity about those: ability to identify the essentials, to understand what should be done under what priorities; what actions and competences are needed, what should one's own role and responsibilities be, and so on. After that there is an opportunity to design action appropriately, to use or acquire needed competences, and sometimes more critically: share the understanding with other. Below those competences, everyone needs the basic orientation and knowledge about the diversity or the environments and situations they work in; that there are no one-size-fits-all approaches.

Forces related to this: fluctuating environments, more dynamic situations in team, work market dynamics.

Broadness of competences. The environments are dynamic in various means. The contexts for testing and quality assurance vary dynamically in vertical sense, ranging from hard technology to new business ideas. They also vary horizontally as the product concepts are more diverse. At the same, time projects are shorter as are the lifespans of business and employment. Thus, adaptation skills are very critical. Those are supported by broad competences, perhaps a T-shaped competence profiles. People who just have a deep, focused competence are not sufficiently capable of adapting to new concepts, making sense of those, communicating, planning the necessary actions and so on.

Thus, specialised testers need to have more than one competence area. If they work mostly on the business level, they need to expand their competences to some supporting area, such as security or some generic product technology or customer and user related skills. The same applies to people working on the level or technology. They should have vertical reach to the domain level issues.

Forces related to this: teamwork, smaller units, need for speed and efficiency, collaboration and communication.

Understanding own limits. Professional level or action requires professional skills. Everyone must understand her own limits regarding for example user experience or security.

Forces related to this: teamwork and dynamic role and task allocation, changes in customer contexts and product contexts,

Business orientation. All people who do testing need to raise the abstraction level of their thinking from engineering or testing tasks to the business level. They need to raise issues such as: how can I with my testing competences improve the business the current activity is about? This is mainly an orientation competence and does not imply any concrete skills.

Forces related to this: need for innovation, competition and dynamics in markets, small companies and startups.

Personal competence creation competences. It is impossible to think that any professional would have sufficient competences for the rest of her career after graduating from a school. Previously, employers have provided some extra training to support competence and career development in the reasonably stable contexts and multi-year job profiles. Now, the professionals need a will and also the skill to plan their competences ahead. One important strategy here is not to follow just one, clearly focused competence development path, but to take care of the diversity of competences. And there is a need to think about the future broadly: what are extensions of competences – the horizontal reach of the “T-shaped” competence profile – that would support employment and also personal growth?

Forces related to this: constant change in technology, business; need for lifelong education, lacking training provisions in companies.

7.5.3 System under test

Special technological domains. Finland has hopes and potential in the development of advances systems for Industrial Internet and also robotics in many industrial domains. The domains need engineering excellence and also testing competences related to the technologies used, which can be expected to be a combination of automation technologies, network connectivity, sensors and artificial intelligence. Many companies in the working machine sector have been in the transformation of changing from a metal machine builder to a developer of intelligent software-based systems. This transformation will continue and enrichen.

Forces related to this: new technology, renewal of industry, new opportunities, need competence cluster need world-beating competences and products.

Capability of product concept, user experience and business level testing. Testing has moved its focus from engineering level to the level of assessing product ideas for their business potential. There is a need for people who can understand concepts and analyse and test them. User experience testing is essential. It is known that usability testing, spanning from the ergonomic principles, never reached the volume and status that was hoped for it, but now that the goals and hopes are in innovation, the same must not happen for UX testing. User experience testing is often done for some implementations, be it a minimum viable product or a prototype, but more concept analysis competences would be very useful, as the first minimum viable products can lead the development to a wrong track, if the development team does not understand the conceptual issues at hand. In general, many business level testing contexts require good experimenting competences – planning, facilitating, executing and assessing experiments. Some people will be experts on this, but everyone who does testing, needs to think about the tests from the business viewpoints: how does a feature relate to the success of the product and the satisfaction of the customer? How to design and select tests that give the most information about that?

Forces related to this: change from engineering to product development level, user experience is the main differentiator, product understanding from special units to small teams, critical for market entry for startups.

7.5.4 Development goals

Testing of positive factors. Testing has traditionally been negative in nature. If test cases pass, there is not much to say. Usability testing and user experience testing point out positive aspects of the system under test too. There is a need for more business / concept level testing that will identify unique positive aspects about a product, that can compare a product to the competitors or its versions. Testing should be used for finding out what’s good, what’s fantastic in the product. This is needed now that we are in the era of innovation. This is much related to growing experimentation and practices in Lean Startup.

Forces related to this: support for product development, innovation, success in markets is always relative to competitors.

Understanding quality. Relevant quality factors expand all the time. Everyone must understand the overall scope of quality, including user experience, security and reliability and the principles of how they are designed into the systems and how the designs and implementations are validated. That understanding is needed for effective teamwork, planning of activities and understanding when to get external help for real experts. This understanding lags always some years behind the needs – companies started considering for example security, usability and user experience with a delay.

Forces related to this: overall quality required for world-class products, diverse concepts, growing requirements, risk management.

From quality management to quality advocacy. The dynamics in organisations, including development teams have not changed. Developers, product owners and others are under terrible pressure to deliver new functionality. There needs to be a counter-force to that, sometimes called leadership. Someone must emphasise quality, try to sell ideas and actions related to that, help the team to avoid quality debt, criticise the product and so on. Traditionally this role has been focusing on meeting requirements and doing process improvements, but now there is a need for more coaching and persuasion type of action. One important domain for this is the world of the startups that at some point need to raise their action to new level. This happens at the point where they understand what they are doing, what their product is and try to turn into a growth mode. And good fact-based product criticism is absolutely valuable in all contexts.

Forces related to this: teamwork, collaboration, empowerment.

Security and safety competences. Robotics, industrial Internet and multi-device environments in general are safety-critical areas and there is a need for people who can do design, analysis and testing of security, safety and reliability issues. There is a need for people who can do safety, security and reliability risk analyses and people who can do very professional security assessments and security testing using proper techniques and tools. Those areas simply cannot be neglected any more.

Forces related to this: security critical for all connected products, IoT, industrial Internet, cyberwar, safety critical products and machines are software-base.

7.5.5 Organisation

Understanding business. Every actor needs to understand the company's business at some level. What are the goals? What do the customers and users value? What information do the business people need in their work and when? What do they expect from testing? This competence allows people to understand what should be done at any time in a real context.

Forces related to this: business orientation, no mediators any more, product development, user experience.

Understanding the lifecycle needs of company and business. Especially in the startups, all activities need to focus on the pressing needs of the company. When the startup is formed, the focus of testing needs to be on understanding the user and gaining information that helps the company form focus and to decide what their products and services should be like. But after that, focus needs to be shared with for example test automation that helps keeping the product solid during the growth phase. The competences are about orientation towards the business and selecting the practices that provide the most value at each phase.

Forces related to this: company's focus, startups, dynamic development, resources needed for test automation.

Testing competences for all – but in a new way. Testing as an occupation will remain important in many contexts, as the systems really need people focused on testing. But the need to do testing is blending in all occupations and tasks. That testing is not in nature the traditional validation of requirements, but more the planning, designing and assessing experiments. Every professional should have some skills in that.

Forces related to this: effectiveness, integrated work profiles, lean organisations.

Diversity. Diversity even in the actions and actors related to quality is very positive for adaptability to new contexts and for successful productisation of innovations. It will also make any stable situation more rich and robust for problems. Diversity can include paradigms, approaches, practices, methods and tools.

Forces related to this: creativity, multiple viewpoints reduce risk on negligence, broad competence set in teams, adaptability to new contexts, needs.

7.5.6 Teamwork

Going for testing opportunities. Teams are prone to group-think and problems and utilising expert skills. Group dynamics may lead a team to under-utilise critical skills and to product mediocre results. Teams need a shared understanding of the development goals and the activities related to that. This is a shared meta-skill: understanding goals, understanding the actions required to reach them, dividing roles appropriately in a value-based way (in contrast to resource utilisation and working-hour sheets) and planning activities so that quality-related tasks have an opportunity to get done. This is also related to the experts' skills of "selling" their value to the team.

Forces related to this: self-organising teams, task identification by discussion, low regard for standards – less mandatory testing.

7.5.7 Processes

Data analytics and experimentation. Many new product domains are data intensive. Data volumes generated are huge and will provide opportunities for understanding the uses better. This data collection and monitoring can be seen as an extension of testing. While it will not replace the need for excellent design skills or any of the development phase testing activities, data analysis is valuable in A/B testing and long term assessment of designs and implementation. The industry sees a general need for data analytics skills and those skills need to be combined with experimentation skills, which together will form a new type of testing competences.

Forces related to this: Big Data, monitoring and analysis opportunities, moving to service business.

7.5.8 Tools and methods

Intelligent and efficient functional testing. Testing of complex systems requires skills of observation that only a human currently has. While test automation is important, nothing has changed in that the test automation is not good at finding new defects and is inflexible when designs change. Good exploratory testing is thus essential and relying on test automation, especially on low level test automation, is a mistake. It should be noted that exploratory testing, while finding defects on new implementations, also helps in targeting the automated regression tests to areas and test conditions where it really is needed, making it more effective and efficient.

Forces related to this: complex systems, challenges with test automation, multiple paradigms bring effectiveness and adaptability.

7.6 Competence lumps

The focus in providing competences to activities have moved from occupations to roles and from those to just having the competences available as needed. There is less needs for a tester's occupation and in some domains there is no "room" for even such role. Instead, there is a need in every occupation and every role to have in some measure testing competences. Those are at best not just single competences, such as being able to do a certain type test, but broader collections of competences that combine orientation for testing – a deep feeling that testing is something that is essential in given circumstances, an understanding of what kind of testing should be done, and the ability to do or arrange such testing.

Any description of essential competences will produce a fragmented view that does not sufficiently consider that competences exit in "lumps" – some competences are closely associated with each other by their nature, goals of their utilisation and how they are expected to exist in some practical role in product development. Identification of such lumps will also help in the creation of training programmes, career development and recruitment. Two such lumps often appear in discussions, namely business/user oriented tester and technically oriented tester who would have related competences. But those are i role based, coarse and empty definitions and we now need to see what kind of lumps could be identified based on the analyses.

If it is possible to define good competence lumps they would allow for creating extensions to the competence expectations in some roles and training programmes that produce good, broad abilities, and as the final result, professionals who with their competences can get things done in practice.

The lumps as are such that a person may have various of them, in varying measure, and they broad, which helps persons have competences that link into each other (as in the T-shaped competence profile idea), thus enhancing collaboration, communication and effectiveness of any shared or individual activities. Yet, they may "officially" have little overlap, which is good for e.g. training programme development.

The following competence lumps are now proposed that would capture some essential characteristics of the competences for the coming years.

Table 53. Proposed competence lumps for testing domains.

Competence lump	Competences
Business-supporting testing competence	Experimentation competences #A. Comparison testing competences #A. User and customer experience evaluation and testing competences #A. Business understanding #U. Product culture understanding #U. Product development understanding #U.
Security and safety testing competence	Security assessment and testing #A Cyber risk analysis #A Understanding human error #U Understanding technical vulnerabilities #U Understanding business risks #U
Test automation	Test automation #A Creation of virtual environments #A Understanding cloud systems #A Using testing tools #A Understanding deployment #U
Testing of intelligent system	Understanding of AI technologies #U Risk analysis #A Safety analysis #A Understanding Big Data #U Data analysis #U Technology evaluation #A
Quality information exploration skills	Exploratory testing #A User experience testing #A Personal understanding of quality #U Safety, security and reliability analysis #A
Independent team player	Quality advocacy #A Role finding #A Test support for team #A Test support for management #A Quality process development #A Quality communication skills #A

Competence lump	Competences
Fast entry to new context	Role finding #A Team skills #A Business understanding #U Understanding of product cultures and technologies #U Adaptive arrays of testing approaches #A Contextual sensemaking #A Technology-agnostic competences #A

In many contexts the division of work into roles will guide the understanding of what lumps are essential. There is one exception, though. Startups are a new phenomenon and may need special consideration. This is why there is a special lump specification for them. It emphasises the competences needed in a situation where support of the emerging business is critical and as critical is to have the mindset to change the startup's approach when the business ideas has matured and it is time to move on to the growth phase and to create processes and systems to support the scaling either in clientele or in the product's feature set.

Table 54. The startup testing lump.

Lump	Competences
Startup testing	(Business-supporting testing competence lump) Exploratory testing #A User experience testing #A Personal understanding of quality #U Quality process development #A – to be able to drive the transformation into growth phase

One may think, is there or should there be a more formal link between the lumps and all the analysis, lists and models that were presented beforehand? Shouldn't a scientist work that way? Here we are in a synthesis phase of the process and that is by nature design. What we know of designing products, models or any artefacts is that the creation of this kind on concepts is done mentally, utilising various describable and tacit processes which shouldn't and cannot be formalised. That is the mystery of design. Instead, after presenting such new artefacts they can be scrutinised and that is best done in some particular context. In the general case, all these lumps are supposed to be approximations and need to be specified an analysed further in conjunction of a broader scoping competences that describes the overall competence "portfolio".

7.7 End of low-competence professionals

This is something that needs to be put in writing even though the situations has been quite clear to practitioners for a while. At the turn of the century, it was common to think that people who work in tester's occupation need not be that skilled. After all, testing was supposed to be simple manual work of inputting data into a system and comparing outputs to some specification. And that was done over and over again in regression testing. That kind of work definitely does not require much skills.

The days of that kind of tester occupation are over. Test automation takes care of simple repetitive tests and intelligent forms of testing (e.g. exploratory testing) are used when new functionality is assessed. The new world requires good competences, whether the person is full-time testing professional or does that on the side of other tasks.

7.8 Learning-related competences

Learning is obviously very important when things are changing around us and the ability to learn is one general competence for all "knowledge workers". There are many different types of learning that have all shown clearly in the preceding analyses.

- Continuous, life-long learning of new approaches, new technology areas and technologies and supporting competences.
- Fast learning of essentials of any new context that one enters.
- Shared learning in a work community,
- Learning of the products under development (including learning about the relation of the product to its users, and similar).

Those are the needs and in this dissertation we are looking into practical competences that fill those needs, especially from the viewpoint of testing. One clear signal in the analyses of the environment is that there is not much room any more for any dedicated learning activities. Instead, the elements of learning need to be built into the productive activities. That would mean that all the future practices need to have elements that support personal and collective learning; creation of new knowledge, transfer of tacit knowledge, good communication about things from various viewpoints and so on. That will bring benefits not only for the person, but for the current work community (Figure 93).

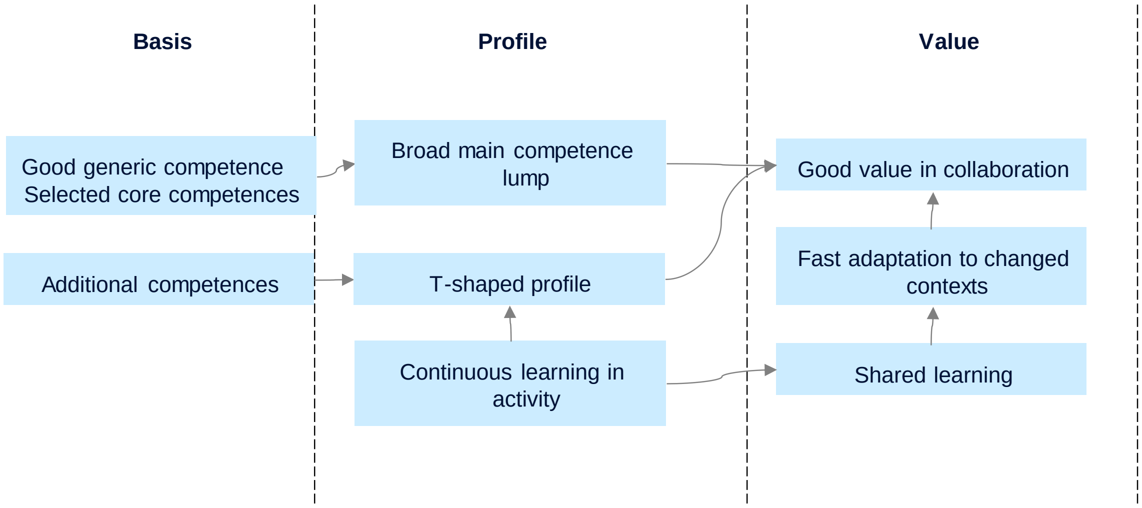

Basis
Profile
Value
Good generic competence
Selected core competences
Broad main competence lump
Good value in collaboration
Fast adaptation to changed contexts
Additional competences
T-shaped profile
Continuous learning in activity
Shared learning

Figure 93. Competence profile and learning as value providers

The quest for T-shaped (or any similar shape) professionals Is important, as it is hard to learn anything if there is no basis for collaboration.

Experiments and exploration are all about learning. The move into more experimental culture should aid shared and personal learning greatly, if the experiments are done in a good way. The Lean Startup proponents talk about validated learning and that may sound too excessive in product development, but learning really can be such a critical asset that it is a reasonable expectation for experiments.

Exploratory testing is nowadays very common and not news anymore. Yet, there are cultures that don't appreciate it as testing approach, but only see automated testing as real testing. They will lack the benefits of exploration and the understanding it can create – including the understanding that helps create good automated tests.

User experience and customer experience analyses and tests are essential for learning because they form a link between technology and business. Such links are essential to create for the dialogue between different disciplines and domains of action.

One very critical lesson the industry learned with Nokia was a too platform-specific competence set. Platforms change and knowing only platform-specific tools and techniques causes a serious problem when the change happens, and before that, restricts dynamic interaction between domains. Testing practices, luckily, are platform-agnostic, but when learning tools and taking them into use, platform-agnostics is a value. At least the tools should support generic ideas that are portable. For example, unit testing tools than are based on the xUnit paradigm are such. All in all, trainings should emphasise the ideas the tools represent and not the particular implementations.

7.9 Changes in core competences?

Core competences are such that are assumed to be needed in every context, in (most) every project. They are limited to the discipline in question and the understanding of what they are is defined by tradition and culture. For example, core competences in testing are usually thought to be similar to what the ISTQB Foundation Level syllabus presents, such as understanding the test process, testing software using test cases, reporting defects and so on. General communication skills, for example, may be core competences for a modern professional, but not core competences in testing.

Additional competences are the ones that are needed only in some contexts, in some situations and some projects. They can also be replaced with some other competences if they are not available. For example, ability to do model-based testing is not needed in every project and even when it might be very beneficial, it is not mandatory. Likewise, user experience testing can be absolutely critical in some projects, but there is no need for it or value from it in testing non-interactive very technical system components.

Note that the core competence on personal level is a different thing than an organisation's core competence as defined by Prahalad 6 Hamel (1990). That means a competence that 1) provides potential access to a wide variety of markets, 2) should make a significant contribution to the perceived customer benefits of the end product and 3) is difficult to imitate by competitors.

So, is there any change on the core competences? In every context there is still a need to do behavioural tests on a system, that is, give some input or stimulus on it and see what happens? Functional testing is based on that and still forms the backbone of the testing culture. But we need to see that security testing follows the same principle. In both cases the core includes test analysis, the tasks than produce the understanding of what should be tested.

User experience and usability testing on the other hand are not based on giving the technical system a stimulus, but on giving a human a task and monitoring and analysing how well the overall human-technology-system works. Of course this is analogous with functional system level testing where – at best – one system element is given the stimulus, but the whole system is the unit under observation. We could say that a systemic approach is a new core skill, as it can and should be used in most testing situations. There should be orientation towards systems thinking, ability to understand systems and ability to design and execute tests that reveal the behaviour of the whole system.

This rising of abstraction level is also related to the need for understanding the purposes of the system under development even on business terms. That has traditionally been somewhat hidden on some testing domains, but gradually we see that without such understanding, effective testing is not possible. There is a general trend of moving the goals of testing from neutral recording of observations to forming understanding of what they mean. Ability to do that reliably can be considered a core competence if any.

However, clearly the limits of core competences are getting blurred and it seems that a sharp distinction between core and other competences may not be wise; it may even be risky.

7.10 How much competence do we need?

Can there be too much of a good thing? Can there be too much competence? Each competence people have, usually means that it is lacking somewhere else, or at least the effort spent on gaining that competence could have been spent gaining some other competence.

Each competence is utilised for some purpose, so we might need to look into those purposes. Kano's model of customer satisfaction (Berger, 1993) is a classic in the product development cultures (Figure 94).

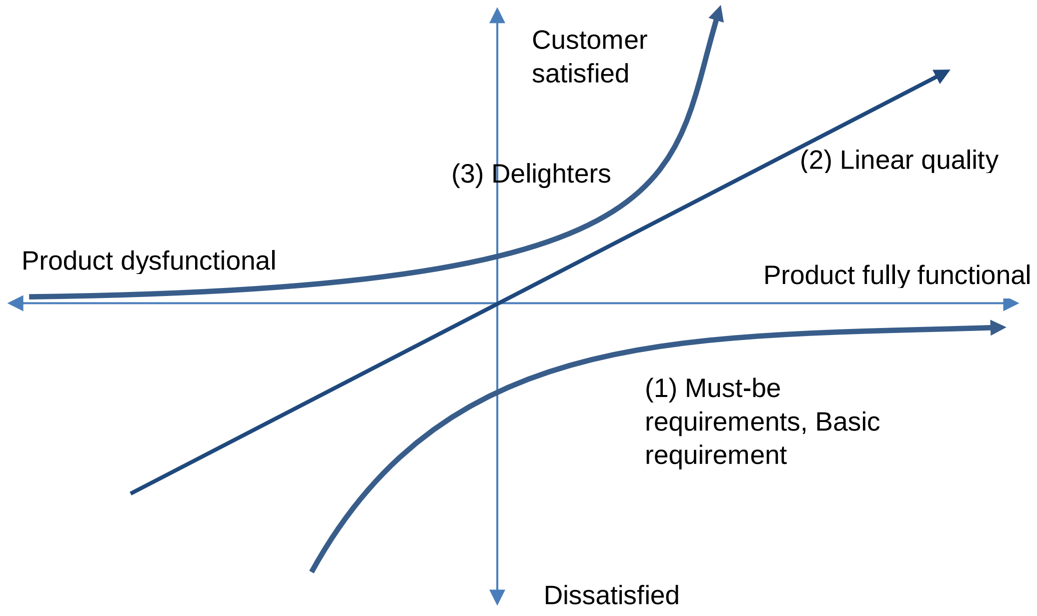


Figure 94. Kano's model of customer satisfaction (Berger, 1993), here in modified and simplified form. Note that there are many variations of the basic model, with terms tailored for any particular case, but each delivering the same message.

According to the model, there are three kinds of product requirements. First there are the "must-be requirements" (1). Those are mandatory things the customer expects. For example, the application must work in Windows. There is not much to do about it once the application does that, so there is no need to spend much effort on it or hire the best experts to work on it. Then there are requirements that are linear (2) in a sense that they can be improved markedly and each improvement brings along more customer satisfaction. Response time of an information system could be like that. The user can tolerate a slow response, but any speedups will be seen as such and will improve satisfaction. Then there are the "delighters" (3). They are characteristics that with some effort will differentiate the product and make it desirable and satisfying. A well working user interface solution would be like that.

Mapped into those types, we can see that functional quality is of type 1. If the software crashes all the time, it is intolerable, but after it gets more and more reliable, the dividends from improvements get smaller and smaller and improvements really make no sense. That means that if certain level is reached with some competence, there is not much motivation for anything better. That can sometimes mean that the sufficient functional quality can be reached by developers that understand testing, without anyone who is an expert in it. But then there are situations where the company fights rising complexity and there is a need for really excellent testing skills that build new testing arrangements, optimise test sets and so on.

Security is somewhat similar, with the exception that it is a more critical type of quality, as even a small vulnerability can be costly. The focus on it needs be integrated into all development, but every now and then there needs to be specialists looking into issues. They may be internal or external consultants. If security can reach a reasonable level, the organisation can focus less on it, yet there is always a risk and a need for constant awareness.

User and customer experience is the area of delight. There can't be too much of it. Every input in it should pay back with customer satisfaction and better business. That's why there can also not be too much expertise in that area, both in design and in testing. And any development that uses any experimentation mixes those paradigms.

So the message should be clear. In the customer and business related testing there should be as high competences available as practically possible, but on some areas, sufficient expertise is good for the sufficient efforts, leaving opportunities to put the critical areas into more focus – in collaboration by all participants, not just by the experts.

8 Assessment of the dissertation

8.1 Main contributions of the dissertation

The main contributions of this dissertation are:

- The development of the competence architecture, which has types of competence that support the ideas and needs of modern collaboration. The types used are orientation #O, understanding #U and ability to actually do the needed things #A.
- The architecture of linked change-competence snippets that help in analysing the changes and essential competences and would enable “what if” analysis of a context – If this changes or doesn’t change, what would be the competence implications?
- The concept and definitions of competence lumps that combine related competences in unified wholes.
- Use of the triangle model of the activity system used in action research in a software development and competence development context.
- Reasonably deep analysis of many essential changes in our environment that will enable researchers and practitioners to understand issues.
- An approach of research that looked into testing as an element of activity that is very tightly integrated into its surrounded social activity.

8.2 Reliability of the research

This dissertation is about the near future and quite holistic in nature. That always causes some differences to the expectations for reliability. First of all, nobody knows about the future, we can only see some continuums and signal about what the coming years will be like. There is always a risk of gaining consensus about the future by finding distilled consensus about matters but that would be a safety-seeking approach and would lose the essential insecurity and richness of ideas and thought. Reflecting to software engineering culture, the consensus-approach would be like seeking an executable requirements specification that all parties would be willing to implement. There are times and places for that, for example in making national policies and action plans, but a dissertation should be a different kind of thing. This research did look into what the national experts think, but it was not seeking for consensus, but more for finding another reflection base.

There is not much literature that looks into the issues in an open-ended way and therefore this research relies on analysis. The idea is to go into as much detail as necessary to show the essence of a phenomenon and its implications and to allow the readers to see any mistakes in the analysis. Yet, a rigid formalism was avoided as it would lessen the readability of the analyses and perhaps simplify things too much. The analysis of the reflective survey applies the “coding” practices of qualitative research, but even that is done informally.

This research aims at looking at detail level to the environment and activity in the world of product and software development and finding things that need assessment, things that raise questions. After that comes the need for answers. For that there is a competence model, which should be seen more as an example than a definitive truth. The model is based on relations between the actors and the elements of the activity system. One sees immediately that it is not based on a process of testing or similar, but the models of activity systems, which are based on long-term research and are very stable. That is in contrast to processes and workflows that are changing all the time,

The dissertation had actually one goal: to understand the competence needs for testing and quality assurance in Finland and to that end, the work analyses the elements of Finland and the related activities on many levels and many viewpoints. There are little limitations on those (expect political conditions), which means that the reality around us has been considered as fully as possible, without resorting to just parts of the environment or the activity systems. That seems to be rarer than restricting the focus for example in task characteristics of processes. That should enhance reliability, as long as the analyses are of good quality.

How about alternatives to the research process? Could the thesis have been based on purely on questionnaires for experts for identifying the priorities? If we accept that we culturally don’t understand the issues at hand and that all experts are focused on their narrow fields, then a survey-centred work would product a misleading compromise that would be a fine example of bad science and research. A similar result would come from relying on lists of change vector and priorities made by others. It would be similar to doing testing based on test cases made by others. If that is not a good idea on a more controlled, low level activity, how could it work on the broad level of this thesis?

The situation in this dissertation is similar to the famous painting by Hanabusa Itchō (1652–1724) where blind men touch an elephant to learn what it is like.

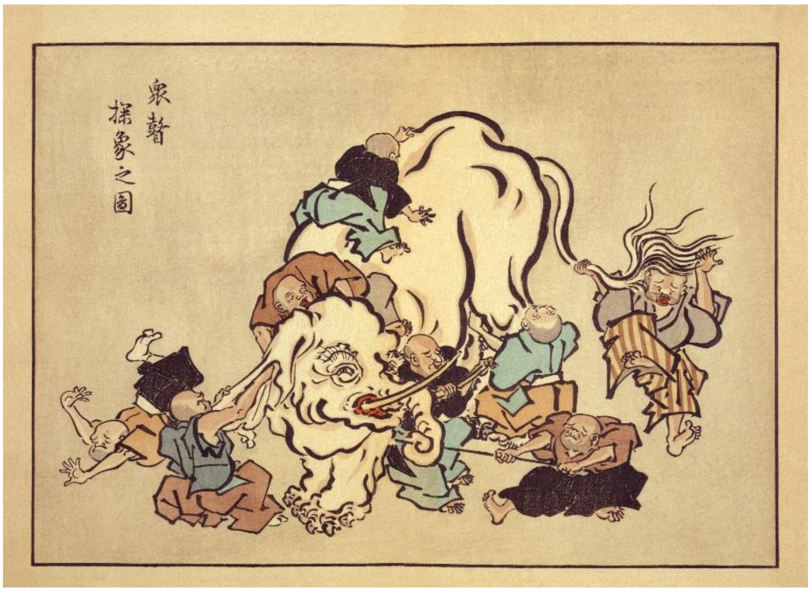


Figure 95. "Blind monks examining an elephant". Hanabusa Itchō (1652–1724). (https://en.wikipedia.org/wiki/Blind_men_and_an_elephant).

The painting has been re-drawn in Western business contexts many times, as it nicely portrays the problems of understanding unfamiliar issues. The analogies between this dissertation and situation in the painting are the following:

- Testing experts are “blind” to the whole as they are deeply focused on their particular domain, culture and discipline.
- The elephant is an unknown thing like the future and one can really grasp it by looking from one’s own perspective (or “handling it” like in the painting).
- If we were to make a synthesis of what an elephant is from the statements of the blind monks, the result would not make any sense.

Also, we need to remember the focus to the future. That means in the cases of the priorities that they simply are not known and cannot be known. All changes may or may not last and their magnitude may vary. It is all about the choices we make and in that sense, all the factors that enable us make the choices are most important. That means that the things happening in the workplaces are much more important and interesting than any changes in particular product domains or technologies. But still we need to see any critical changes in technologies that absolutely are present everywhere and some of those have been singled out in the analysis.

Due to the dynamic characteristics of our environment, changes are prone to change (!). For that reason, while the actual changes assessed offer a glimpse to the possible reality, we need to be able to change our ideas about what changes and what is relevant. Because of that, the interlinked change-competence snippets offer a mechanism for doing “what if” analyses. It is possible to introduce new changes, remove some or refactor them, execute the graph- and table-building macros again and see how another version of the world would look like.

Another alternative would have been to use proper futures research approach with varying levels of scenarios. In fact, this dissertation uses a high level positive scenario of Finland as given and tries to see how the “seen” changes would relate to that – support that and get support from that.

The Delfoi method is often used in this kind of works, but as the thesis does not focus on a single domain, there would have been difficulties and the result could have represented the “famous” demographic studies that describe what an average human is, and would have lost the diversity of the nations at the process. And maintaining the diversity seems to be critical here.

8.3 Reflection on the quality criteria of the dissertation

In the beginning of the dissertation, a list of quality criteria was presented. Now it is time to shortly see how the work looks against them:

The rationale, basis for the competences architectures used. Are they founded in a solid thinking?

The architectures (the levels, the relationship models) are all based on proven ideas, but “remixed” here for the goals of this work.

How well do the competence architectures help us understand the issues of competence in this context?

The change-competence view offers a direct link between some phenomenon and the competences that are relevant to it and which ones are the most essential. It is a very “operative” view. But the model where competences are linked with the elements of an activity system offer another view, where one is led to more deeply understand the activity system. Both are insufficient without the other; they are complimentary.

One of the main ideas here and elsewhere is that testing and QA competences need to be integrated with other competences. While many companies are breaking down strict occupational and role boundaries, they exist and once the new companies grow in size, need more consideration. It would have been nice to assess the competence division between roles and be more specific about many things such as what an account manager really should understand about experiments, UX testing and so on, but that is best to leave for other studies. It might not require deep analysis, but some analysis about it should improve our understanding.[30]

How sensitive is the analysis regarding how the world changes? After all, this is about the future and that is generally somewhat unknown.

The substance analysed – the changes – are something that is prone to change. That is something to expect and accept.

Quality of the analysis of the selected changes.

The analyses are somewhat subjective and based on views that have grown from experience. The analyses are detailed for the purpose of exposing their assumptions and weaknesses too. They form the philosophy of qualitative research and their quality is the total package: subjects, analysis, conclusions, the display of factors and their relationships and causal factors and so on.

This criterion includes the quality of the classification system for the competences. The competences were derived in a bottom-up way. That makes the naming conventions somewhat unharmonised there is variation in them in the different parts of the dissertation. Thus, the taxonomy is lacking for serious use in industry or in education. That is a deficiency by design. Creation of a harmonised, good-quality classification system would be a separate project.

How well does the work overall help us understand the contexts in which testing is done, their characteristics and what testing could be like in them?

Doing that was a main attempt in the dissertation, because no amount of formal analysis and presentation of results will matter if there is not a shared understanding of the reality.

Reusability of the methods used?

The methods are quite reusable. The analyses require on knowledgeable person who understands quality and testing. The architectures for collecting and linking data are substance-independent and the existing analyses can be expanded and tailored as needed.

[30] This could be more a personal wish for clarifying the issue than a real generic pressing research need.

9 Conclusions

This dissertation is about future and dealing with that is never easy, for obvious reasons. That is why there was only a hypothesis about it:

“It is possible to assess the current views regarding personal and organisational competence and to formulate a new view that gives an improved insight to the current and future needs for competence and capability. That view will thus give guidance to the improvement of the competencies and capabilities. That is turn will lead to better performance of testing and quality assurance in software development.”

Things could have turned out so that the conclusion would have been that the environment is just too turbulent and nothing could be said about it except some trivialities. Luckily, it seems that it is possible to do such assessments and gain improved insight to the situation with the methodology used. Key point in it was a deep analysis of – subjectively selected – changes in our environment, which seem to link into each other and form a basis of opportunities and needs. And we don't need to use any buzzword to coin the whole (such as Something 4.0). Much of the opportunities are there for us to make happen, but they don't happen automatically. There have been enough talk about death of testing and trying to turn it into some automated activity. That would be a grave error. When we see that the society needs innovation, rethinking of many areas of technology with the help of new kinds of systems and new consumer products that people would love globally, that is the playground for testing too. We all are tied to old memes in our mind and may think that product experiments are not testing. So, what are they but testing? If a thing looks like testing (or a duck!), is based on same principles as testing, has the same goals as modern testing, then it is testing. But seeing that requires a change in mindset. We must stop seeing testing as an activity of executing low level tests on an already done implementation.

In fact, the whole nation seems to have fallen in love with innovation, but they seem to expect miracles and lack a differentiation between ideas and innovations. There are plenty of ideas, but execution of them into products is what counts. There is the emerging exploration culture, but is focused on speed and lacks analysis of the ideas. The principle of "creation miracle" and validation by testing is itself very valid, even though systematic design is very important in many situations too. So testing of the concepts and their varying levels is critical for real innovations and that is something that the nation should learn to do better. The Lean Startup brings along it the MVP and the idea of validating it, but the validation part is in practice weak. There is a culture of shallow interaction with users during development and as well a shallow culture of working with product concepts. The engineering country likes to work with artefacts, such as code, instead. In fact, much of the discussion about testing is focused on test automation, which is easy and exiting, but advances in it solve only some of the challenges. There is clearly a need for change at the cultural level. That is where education comes in, because it can gradually change the culture in companies by "producing" new professionals with improved mental models about product development and thinking patterns that can be turned into action in companies.

There is talk about moving testing to the left and that is needs, but not only moving it into the software-engineering-left, but the product-development-left. People still often have difficulty in differentiating those. It is now clear that there is plenty of work in the world on MVPs and fuzzy front ends. We have a national tendency to think that an expert can only be hired if she can fill 100 % of her weekly hours in "productive work". That is of course insane. The idea is to create successful products that bring more income than just what is needed to pay the wages. Did Apple monitor how Steve Jobs spent his days? The question of productivity is important on many domains, but for intellectual work researchers have for decades agreed that working hours are not a reliable metric of a company's success, yet it remains an easy topic in political rhetoric as seen in Finland lately.

The same rationality-gone-wrongs shows in many industrial thought patterns. Lean was supposed to be about waste hunt, but we now understand that the idea was to find ways for a small car manufacturer to reach the volumes of American giants. Completely different goal.

So, in a way this dissertation points out that testing is essential for the creative part of product creation, but also as a balancing force that also needs changing at the entry to a more innovative culture.

The dissertation had a practical goal of "finding, with some reliability, what the necessary personnel competencies and organisational capabilities that makes it possible" to do the excellent product business that the nation needs. It now strongly looks like that goal has been reached and the results point those competencies out from many perspectives.

As part of results are the proposed "competence lumps" that provide guidance for the elements of competence for different types of new professionals, be they focused mostly on testing or just doing it on the side of something else. They also may be focused on the business side of products or in the new risky areas of security, safety and reliability, which are a necessary part of the backbone of e.g. robotics. Without security, safety and reliability, digitalisation in its many forms, will fail and as well will fail the product businesses.

None of the competence development areas should be handled blindly, but with thorough understanding of the issues that enable or require them. The analysis of the changes should give support for that and plenty of "food for thought".

The methodological ideas were based on seeing the environment as a system consisting of various context, which are in transition. The system thinking is increasing in all walks of life and clearly is more important as ever, when the world is more complex, fragmented and connected. Looking back to the work, it clearly had a leading idea of finding connections between things and using the connections as one basis for the conclusions about what things are relevant and important. It is hoped that software engineering research moves more to that direction. There have been some positive movements. Originally the research was more about practices as such, without consideration about the actors. Today, the actors, be they developers, testers or something else, are understood to be part of the system, but the presence of organisation is sometimes vague. This it caused by tradition, the separation of disciplines and the resulting world-view of the researchers. But times and thinking patterns change.

Bibliography

Ailisto, Heikki (ed.); Collin, Jari (ed.); Juhanko, Jari (ed.); Mäntylä, Martti (ed.); Ruutu, Sampsa (ed.), Seppälä, Timo (ed.); Halén, Marco; Hiekkanen, Kari; Hyytinen, Kirsi; Kiuru, Eeva; Korhonen, Heidi; Kääriäinen, Jukka; Parviainen, Päivi & Talvitie, Jaakko. 2016. Onko Suomi jäämässä alustatalouden junasta? [Is Finland missing the platform economy train[31]]. Valtioneuvoston selvitys- ja tutkimustoiminnan julkaisusarja 19/2016. 54 p. Available at: http://tietokayttoon.fi/documents/10616/2009122/19_Onko+Suomi+j%C3%A4%C3%A4m%C3%A4ss%C3%A4+alustatalouden+junasta.pdf/5e1f46ed-415c-4763-a530-633309eafb77?version=1.0

Aaltio, Iiris & Puusa, Anu. 2011. Laadullisen tutkimuksen luotettavuus. In: Menetelmäviidakon raivaajat – perusteita laadullisen tutkimuslähestymistavan valintaan. [Clearers of the method jungle – bases for choosing a qualitative research approach]. JTO Johtamistaidon opisto, p. 153 - 165.

Aalto University. 2015. Collaborative and Industrial Design. School of Arts, Design and Architecture Study Guide 2015–16. Aalto University, School of Arts, Design and Architecture. Available at: http://studyguides.aalto.fi/arts/2015/en/master%E2%80%99s-degree-programmes-of-the-school-of-arts,-design-and-architecture/collaborative-and-industrial-design.html. [Checked 2015-12-16]

Ackoff, Russell L. 1999. Re-Creating the Corporation. A Design or Organizations for the 21st Century. Oxford University Press. 336 p.

[31] Titles written in [ ] are the author's unofficial translations meant to give non-Finnish readers an idea what the publication is about.

ACM/IEEE Joint Task Force on Computing Curricula. 2004. Software Engineering 2004. Curriculum Guidelines for Undergraduate Degree Programs in Software Engineering. 44 p. + appendices.

ACM/IEEE Joint Task Force on Computing Curricula. 2013. Computer Science Curricula 2013. Curriculum Guidelines for Undergraduate Degree Programs in Computer Science. A Volume of the Computing Curricula Series. 518 p.

ACM/IEEE Joint Task Force on Computing Curricula. 2014. Software Engineering 2014. Curriculum Guidelines for Undergraduate Degree Programs in Software Engineering. A Volume of the Computing Curricula Series. 133 p.

ACM. 2015. Software Engineering Code of Ethics and Professional Practice Available at: https://www.acm.org/about/se-code. [Checked 3.5.2015]

Ahvenjärvi, Hannu. 2011. ICT-alan tarvekartoitus 2011. Yhteenveto työvoiman ja koulutuksen tarvetutkimuksen ja C&Q-osaamistarvekartoituksen haastatteluista. Elinkeino-, liikenne- ja ympäristökeskus. Raportteja 20/2012. 32 p. Available at: http://www.doria.fi/bitstream/handle/10024/74502/Raportteja_20_2012.pdf?sequence=1 [Checked 13.2.2012]

Ailisto, Heikki (ed.); Mäntylä, Martti (ed.); Seppälä, Timo (ed.); Collin, Jari; Halén, Marco; Juhanko, Jari; Jurvansuu, Marko; Koivisto, Raija; Kortelainen, Helena; Simons, Magnus; Tuominen, Anu & Uusitalo, Teuvo. 2015. Finland – The Silicon Valley of Industrial Internet. Publications of the Government's analysis, assessment and research activities 10/2015. 38 p.

Alanko, Markku & Salo, Immo. 2013. Big data Suomessa [Big data in Finland]. Liikenne- ja viestintäministeriön julkaisuja 25/2013. http://urn.fi/URN:ISBN:978-952-243-358-9. 36 p.

Alasoini, Tuomo; Järvensivu, Anu & Mäkitalo, Jorma. 2012. Suomen työelämä 2030 – Miten ja miksi se on toisennäköinen kuin tällä hetkellä [Finnish working life 2030 – How and why does it look different than at this moment]. Ministry of Employment and Economy. TEM raportteja 14/2012. 38 p.

Alasuutari, Pertti. 2011. Laadullinen tutkimus 2.0 [Qualitative research 2.0]. Vastapaino. 331 p.

Amer, Muhamma; Daim, Tugrul U. & Jetter, Antonie. 2013. A review of scenario planning. Futures 46 (2013). p. 23-40.

Applause. 2015. [Applause website] www.applause.com. [Checked 20.10.2015]

Arnott, David. 2006. Cognitive biases and decision support systems development: a design science approach. Info Systems Journal (2006) 16, p. 55-78.

Astigarraga, T.; Dow, E.M.; Lara, C.; Prewitt, R.; Ward, M.R. 2010. The Emerging Role of Software Testing in Curricula. Transforming Engineering Education: Creating Interdisciplinary Skills for Complex Global Environments. pp. 1-26.

Athey, Timothy R. & Orth, Michael S. 1999. Emerging competency methods for the future. Human Resource Management, Volume 38, Issue 3, p. 215–225.

Bach, James. 1999. Heuristic Risk-Based Testing. Software Testing and Quality Engineering Magazine, 11/99. 9 p.

Bach, James. 2008. Schools of testing… Here to stay. James Bach's Blog. Available at http://www.satisfice.com/blog/archives/134 [Checked 26.3.2012]

Barab, S., Evans, M. A. and Beak, E. 2004. Activity theory as a lens for characterizing the participatory unit, in D. H. Jonassen (ed.) Handbook of research on educational communications and technology: a project of the association for educational communications and technology (pp 199-214). London: Routledge.

Barile, Sergio; Franco, Giacomo; Nota, Giancarlo & Saviano, Marialuisa. 2012. Structure and Dynamics of a "T-Shaped" Knowledge: From Individuals to Cooperating Communities of Practice. Service Science, Vol. 4, No. 2, June 2012, p. 161–180.

Barile, Sergio; Saviano, Marialuisa & Simone, Cristina. 2014. Service economy, knowledge, and the need for T-shaped innovators. Springer Science+Business Media, New York. p. 1177-1197.

BBST Testing Course. 2014. Course web site at:
http://www.testingeducation.org/BBST/ [Checked 19.3.2014]

Beck, Kent; Beedle, Mike; Bennekum, Arie_van; Cockburn, Alistair; Cunningham, Ward; Fowler, Martin; Grenning, James; Highsmith, Jim; Hunt, Andrew; Jeffries, Ron; Kern, Jon; Marick, Brian; Martin, Robert_C.; Mellor, Steve; Schwaber, Ken; Sutherland, Jeff & Thomas, Dave. 2001. [Referenced 2011-03-01] Available at: http://www.agilemanifesto.org/.

Beizer, Boris. 1990. Software Testing Techniques. Second Edition. Van Nostrand Rheinhold. 550 p.

Beizer, Boris. 1995. Black-Box Testing. Techniques for Functional Testing of Software and Systems. John Wiley & Sons. 294 p.

Berger, Charles; Blauth, Robert; Boger, David; Olster, Christopher, Burchill, Gary, DuMouchel, William; Pouliot, Fred; Richter, Reinhart; Rubinoff, Allan; Shen, Diane;

Timko, Mike & Walden, David. 1993. Kano's methods for understanding customer-defined quality. Center for Quality Management Journal (Fall), p. 3-35.

Bertolino, Antonia. 2007. Software Testing Research: Achievements, Challenges, Dreams. Future of Software Engineering(FOSE'07). 18 p.

Bloom, B.S.; Engelhart, M.D.; Furst, E.J.; Hill, W.H. & Krathwohl, D.R. 1956. Taxonomy of Educational Objectives: Handbook 1 Cognitive Domain. Longmans, Green and Co Ltd, London, 1956.

Bolton, Michael. 2009. Two Futures of Software Testing. Presentation at Software Testing and Performance Conference. Available at: http://2009.stpcon.com/pdf/Bolton_Two_Futures_of_Software_Testing.pdf [Checked 13.2.2012]

Bourque, Pierre & Fairley, Richard E. (eds.). 2014. Guide to the Software Engineering Body of Knowledge, Version 3.0, IEEE Computer Society, 2014. 335 p.

Bradfield, Ron; Wright, George; Burt, George; Cairns, George & Van Der Heijden, Kees. 2005. The origins and evolution of scenario techniques in long range business planning. Futures 37 (2005). p. 795–812.

Buckley, Jim & Exton, Chris. 2003. Blooms' Taxonomy: A Framework for Assessing Programmers' Knowledge of Software Systems. Proceedings of the 11th IEEE International Workshop on Program Comprehension (IWPC'03). 10 p.

Buxton, Bill. 2014. Innovation Calls For I-Shaped People. Business Week INSIGHT July 13, 2009 (Revised July 8, 2014). Available at: http://www.businessweek.com/innovate/content/jul2009/id20090713_332802.htm?chan=innovation_innovation+%2B+design_top+stories. [Checked 2.2.2016]

Chydenius, T. & Gaisch, M. 2015. Worklife Interaction Skills: An exploration of definitional and functional perspectives within the Austrian and Finnish ICT industry. In Proceedings of Cross-Cultural Business Conference 2015. School of Management, Steyr Campus, p. 315-325.

Cilliers, P. (2006). A Framework for Understanding Complex Systems. http://www.tsama.org.za/index.php/documents/category/2-td-summer-school-jan?download=19:cilliers-2006-framework-for-understanding-complex-systems. 4 p. [Checked 2016-6-14]

CMMI Product Team. 2010. CMMI® for Development (CMMI-DEV), Version 1.3. CMMI Institute. 482 p.

Cockburn, Alistair. 2007. Agile Development – The Cooperative Game. Second Edition. Addison-Wesley. 467 p.

Cooper, Brant & Vlaskovits, Patric. 2010. The Entrepreneur's Guide to Customer Development. 96 p.

Coplien, James. 2009. Balancing the tension between lean and agile? Presentation slides. Presented in Projektinhallintapäivä 2009 [Project Management Day 2009] at Tampere University of Technology. Available at: http://www.cs.tut.fi/tapahtumat/projektinhallinta09/JimCoplien_12082009.pdf.

Coplien, James O.; Bjørnvig, Gertrud. 2010. Lean Architecture: for Agile Software Development. Wiley. 376 p.

Coplien, James. 2011. Agile 10 years on. InfoQ. This article is part of the Agile Manifesto 10th Anniversary series that is being published on InfoQ. [Referenced 2011-03-11] Available at: http://www.infoq.com/articles/agile-10-years-on.

Corbin, Juliet & Strauss, Anselm. 2008. Basics of Qualitative Research, 3rd edition. Techniques and Procedures for Developing Grounded Theory. SAGE Publications. 379 p.

Craig, Rick D. & Jaskiel, Stefan P. 2002 Systematic Software Testing. Artec House Publishers. 536 p.

Crispin, Lisa & Gregory, Janet. 2009. Agile Testing. A Practical Guide For Testers and Agile Teams. Addison-Wesley. 554 p.

Cross, Nigel. 2006. Designerly ways of knowing. Springer. 114 p.

Crowne M. 2002. Why software product startups fail and what to do about it – Evolution of software product development in startup companies. In: Proc. Int. Engineering Management Conference (IEMC '02), IEEE CS 2002, p. 338-343.

Dande, Anuradha; Eloranta, Veli-Pekka; Hadaytullah; Kovalainen, Antti-Jussi; Lehtonen, Timo; Leppänen, Marko; Salmimaa, Taru; Sayeed, Mahbubul; Vuori, Matti; Rubattel, Claude; Weck, Wolfgang; Koskimies, Kai. 2014. Software Startup Patterns – An Empirical Study. http://dspace.cc.tut.fi/dpub/handle/123456789/22067. 67 p.

De Bono, Edward. 1985. Six Thinking Hats. Penguin books. 178 p.

De Bono, Edward. 1990. Lateral Thinking. Penguin books. 260 p.

De Bono, Edward. 1995. Serious Creativity. Using the Power of Lateral Thinking to Create New Ideas. HarperCollinsBusiness. 338 p.

De Coi, Juri L.; Herder, Eelco; Koesling, Arne; Lofi, Cristopher; Olmedilla, Daniel; Papapetrou, Odysseas & Siberski, Wolf. 2007. A Model For Competence Gap Analysis[C/OL]. Proceedings of 3rd International Conference on Web Information Systems and Technologies (WEBIST). Available at: http://dspace.learningnetworks.org/bitstream/1820/1119/1/model_for_competence_gap_analysis.pdf [Checked 15.2.2012]

Dellana, Scott A. & Hauser, Richard D. 1999. Towards Defining the Quality Culture. Engineering Management Journal, 11:2, 11-15, DOI: 10.1080/10429247.1999.11415022, p. 11-15.

Deterding, Sebastian; Dixon, Dan; Khaled, Rilla & Nacke, Lennart. 2011. From Game Design Elements to Gamefulness: Defining “Gamification”. Proceedings of MindTrek’11, September 28-30, 2011, Tampere, Finland. ACM. p. 9-15.

Draht, Rainer & Horch, Alexander. 2014. Industrie 4.0: Hit or Hype? IEEE IndustrIal ElEctronIcs magazine, June 2014, p. 56-58.

Dvorak, Daniel L. (ed.) 2009. NASA Study on Flight Software Complexity. Final report. 55 p.

Ebert, Christof & Jones, Capers. 2009. Embedded software: Facts, figures , and future. COMPUTER, April 2009. pp. 42-52.

Eloranta, Veli-Pekka. 2015. Techniques and Practices for Software Architecture Work in Agile Software Development. Dissertation. Tampere University of Technology. Publication;1293. 153 p.

Encyclopedia of Philosophy. 2015. Available at: http://www.iep.utm.edu/ethics/. [Checked 1.5.2015]

Engeström, Y. & Miettinen, R. 1999. Activity theory: A well-kept secret, in Engeström, Y; Miettinen, R. and Punamäki-Gitai, R. L. (eds) Perspectives on Activity Theory (pp 1-15). Cambridge University Press.

Engeström, Yrjö. 2006. Activity theory and expansive design. In: S. Bagnara & G. Crampton-Smith (eds.): Theories and Practice of Interaction Design. Hillsdale: Lawrence Erlbaum, pp. 3–23.

European e-Competence Framework. 2014. European e-Competence Framework 3.0. Available at: http://www.ecompetences.eu/wp-content/uploads/2014/02/European-e-Competence-Framework-3.0_CEN_CWA_16234-1_2014.pdf. 53 p. [Checked 24.9.2015]

European e-Competence Framework. 2014a. User guide for the application of the European e-Competence Framework 3.0. Available at:

http://www.ecompetences.eu/wp-content/uploads/2014/02/User-guide-for-the-application-of-the-e-CF-3.0_CEN_CWA_16234-2_2014.pdf. 44 p. [Checked 24.9.2015]

European Qualifications Framework. 2015. Descriptors defining the levels in the European Qualifications Framework. Available at: https://ec.europa.eu/ploteus/content/descriptors-page. [Checked: 24.9.2015]

Eurostat Press Office. 2016. Almost 8 million ICT specialists employed in the EU in 2014. Eurostat Newsletter 15/2016, 21 January 2016. Available at: http://ec.europa.eu/eurostat/documents/2995521/7141198/4-21012016-AP-EN.pdf/f366dacf-bff5-467c-b8cd-ebfba6a44d5b. [Checked 10.2.2016]

Fagerholm, Fabian; Guinea, Alejandro Sanchez; Mäenpää, Hanna & Münch, Jürgen. 2014. Building Blocks for Continuous Experimentation. International Workshop on Rapid Continuous Software Engineering (RCoSE 2014), June 3, 2014, Hyderabad, India. p. 26-35.

FiSTB. Finnish Software Testing Board. [A national board of ISTQB]. http://www.fistb.fi

Foresight 2030, 2013. http://tulevaisuus.2030.fi/en/ [Checked 5.8.2015]

Garousi, V.; Mathur, A. 2010. Current State of the Software Testing Education in North American Academia and Some Recommendations for the New Educators. 2010 23rd IEEE Conference on Software Engineering Education and Training (CSEE&T). pp. 89-96.

Gartner. 2014. Gartner's 2014 Hype Cycle for Emerging Technologies Maps the Journey to Digital Business. http://www.gartner.com/newsroom/id/2819918

Giardino, Carmine & Paternoster, Nicolo. 2012. Software development in startup companies. MSc thesis MSE-2012-99, Blekinge Institute of Technology, Sweden. 233 p.

Glazer, Hillel; Dalton, Jeff; Anderson, David; Anderson, David J.; Konrad, Mike & Shrum, Sandy. 2008. CMMI or Agile: Why Not Embrace Both! Software Engineering Institute. TECHNICAL NOTE CMU/SEI-2008-TN-003. 48 p.

Gothelf, Jeff & Seiden, Josh. 2013. Lean UX. Applying Lean Principles to Improve User Experience. O‘Reilly. 30p.

Graham, Dorothy; Van Veenendaal, Erik; Evans, Isabel; Black, Rex. 2008. Foundations of Software Testing, ISTQB Certification. Course Technology. 258 p.

Hamari, Juho; Koivisto, Jonna & Sarsa, Harri. 2014. Does Gamification Work? — A Literature Review of Empirical Studies on Gamification. Proceedings of the 47th Hawaii International Conference on System Science. IEEE Computer Society. p. 3015-3034.

Hamel G. & Prahalad, C. K. 1992. The core competence of the corporation. Harvard Bus. Rev., vol. 68, no. 3, pp. 79–91, May–June 1990; reprinted in Eng. Manage. Rev., Fall 1992.

Hamel, Gary. 2007. The Future of Management. Harvard Business School Press. 288 p.

Hamel, Gary. 2015. Foreword in: Whitehurst, Jim. 2015. The Open Organization. Harvard Business Review Press. 227 p.

Hammerich, Kai & Lewis, Richard D. 2013. Fish Can't See Water. How National Culture can Make or Break your Corporate Strategy. Wiley. 297 p.

Hansen, Morten T. & von Oetinger, Bolko. Introducing T-Shaped Managers: Knowledge Management's Next Generation. Harvard Business Review, March 2001. 24 p.

Harrison, Neil B. 2010. Teaching software testing from two viewpoints. Journal of Computing Sciences in Colleges archive. Volume 26 Issue 2, December 201. pp. 55-62.

Hartman, John; Manber, Udi; Peterson, Larry & Proebsting, Todd. 1996. Liquid Software: A New Paradigm for Networked Systems. TR 96-11. The University of Arizona. 14 p.

Hasan, Helen & Katzlauskas, Alanah. 2009. Making Sense of IS with the Cynefin Framework. PACIS 2009 Proceedings. Paper 47. Available at: http://aisel.aisnet.org/pacis2009/47

Hassi, Lotta; Paju, Sami & Maila, Reetta. 2015. Kehitä kokeillen. Organisaation käsikirja. [Develop by experimenting. Handbook for the organisation.] Talentum pro. 202 p.

Helakorpi, Seppo. 2005. Työn taidot – Ajattelua, tekoja ja yhteistyötä [Skills of work – Thinking, actions and collaboration]. Hämeen ammattikorkeakoulu. HAMK Ammatillisen opettajakorkeakoulun julkaisuja 2/2005. 208 s.

Hellsten, Kirsi. 2007. Scenario method in anticipating future opportunities of e-business in the forest industry. Master's Thesis. Lappeenranta University of Technology, Department of Industrial Management. 115 p.

Hendrickson, Elisabeth. 2013. Explore it! Raduce risk and increase confidence with exploratory testing. Pragmatic Bookshelf. 162 p.

Hermann, Mario; Pentek, Tobias & Otto, Boris. 2016 Design Principles for Industrie 4.0 Scenarios. Proceedings of 49th Hawaii International Conference on System Sciences. p. 3928-3937.

Hetzel, Bill. 1988. The Complete Guide to Software Testing. Second edition. Wile-Qed. 294 p.

Holmström Olsson, Helena; Bosch, Jan & Alahyari, Hiva. 2013. Towards R&D as innovation experiment systems: A framework for moving beyond agile software development. Proceeding of Artificial Intelligence and Applications 2013. 8 p.

Holmström Olsson, Helena & Bosch, Jan. 2015. Towards Continuous Validation of Customer Value. XP 2015 Workshops, May 25-29, 2015, Helsinki, Finland. 4 p.

Humble, Jez & Farley, David. 2011. Continuous Delivery. The Addison Wesley Signatory Series. 463 p.

IEC 61508-3. 2010. Functional safety of electrical/electronic/programmable electronic safety-related systems, Parts 1-7. International Electrotechnical Commission. Edition 2.0. International Electrotechnical Commission.

Ilkkala, Aki. 2010. Yrityselämän tulevaisuuden vaatimukset ja yritysyhteistyö tietojenkäsittelyn koulutusohjelmassa. Tampereen ammattikorkeakoulu. 25 p. Available at: http://urn.fi/URN:NBN:fi:amk-2010121318024. [Checked 14.2.2012]

IEEE, 2014. Software Engineering Competency Model (SWECOM) Now Open for Public Review. 110 p. http://www.computer.org/portal/web/pressroom/Software-Engineering-Competency-Model-SWECOM-Now-Open-for-Public-Review. [Checked 30.4.2014]

IEEE, 1998. IEEE Std 829-1998 – IEEE Standard for Software Test Documentation. 52 p.

IEEE, 1990. IEE Std 610.12-1990 – IEEE Standard Glossary of Software Engineering Terminology. 84 p.

IEEE, 2008. IEEE Std 829-2008 – IEEE Standard for Software Test Documentation. 118 p.

IEEE. 2015. Towards a definition of the Internet of Things. Revision 1. IEEE Internet Initiative. 86 p. Available at: http://iot.ieee.org/images/files/pdf/IEEE_IoT_Towards_Definition_Internet_of_Things_Revision1_27MAY15.pdf [Checked 29.6.2015]

Industrial Internet Consortium. 2016. Reference Architecture for Industrial Internet. Version 1.7.101 p.

InnoPilotti. 2011. Developing of the uniform innovation course to the educational institution consortium in Demola environment. Project web site at: https://wiki.tut.fi/InnoPilotti/InEnglishhttps://wiki.tut.fi/InnoPilotti/InEnglish

ISO 9001. 2015. ISO 9001:2015. Quality management systems – Requirements. 29 p.

ISO/IEC/IEEE 24765. 2010. Systems and software engineering – Vocabulary. 418 p.

ISO/IEC 90003. 2014. ISO/IEC 90003:2014. Software engineering – Guidelines for the application of ISO 9001:2008 to computer software. 54 p.

ISO/IEC 9126-2, 2001. Software engineering. Product quality. Part 2: External metrics. 112 p.

ISO/IEC 9241-210. 2010. Ergonomics of Human-System Interaction - Part 210: Human Centered Design Processes for Interactive Systems. International Organization for Standardization, Geneva, Switzerland. 32 p.

ISO/IEC 25010, 2011. Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) – System and software quality models. 34 p.

ISO/IEC/IEEE 29119-1:2013. Software and systems engineering – Software testing – Part 1: Concepts and definitions. 58 p.

ISO/IEC/IEEE 29119-2:2013. Software and systems engineering – Software testing – Part 2: Test processes. 68 p.

ISO/IEC/IEEE 29119-3:2013. Software and systems engineering – Software testing – Part 3: Test documentation. 138 p.

ISO/TS 15066. 2016. Robots and robotic devices – Collaborative robots. 33 s.

ISTQB. 2007. Certified Tester. Advanced Level Syllabus. Version 2007. 114 p. Available at: http://istqb.org

ISTQB. 2011a. Certified Tester. Foundation Level Syllabus. Version 2011. 78 p. Available at: http://istqb.org

ISTQB. 2011b. Certified Tester. Expert Level Syllabus. Test Management (Managing Testing, Testers, and Test Stakeholders). Version 2011. 81 p. Available at: http://istqb.org

ISTQB. 2011c. Certified Tester. Expert Level Syllabus. Improving the Testing Process (Implementing Improvement and Change). Version 2011. 75 p. Available at: http://istqb.org

ISTQB. 2012. ISTQB – International Software Testing Qualifications Board. http://istqb.org/display/ISTQB/Home [Checked 13.2.2012]

ISTQB. 2013. Exam Information – Public. Version Version 2013-1.0. Available at: http://istqb.org

ISTQB. 2015. Certified Tester. Expert Level Modules Overview. Version 1.0. 13 p. Available at: http://istqb.org

Itkonen, Juha. 2011. Empirical studies on explorative software testing. Aalto University publication series DOCTORAL DISSERTATIONS 107/2011. 178 p.

Jacobson, Ivar et al. 2012. Essence – Kernel and Language for Software Engineering Methods. 283 p.

Joint IS 2010 Curriculum Task Force. 2010. IS 2010. Curriculum Guidelines for Undergraduate Degree Programs in Information Systems. Association for Computing Machinery (ACM) & Association for Information Systems (AIS). 88 p.

Jääskeläinen, Antti. 2011. Design, implementation and use of a test model library for GUI testing of smartphone applications. Doctoral dissertation, Tampere University of Technology, Tampere, Finland. Publications 948. 68 p + appendices.

Jääskeläinen, Antti; Katara, Mika; Kervinen, Heiskanen, Henri; Maunumaa, Mika & Pääkkönen, Tuula. 2008. Model-based testing service on the web. Proceedings of the Joint Conference of the 20th IFIP International Conference on Testing of Communicating Systems and the 8th International Workshop on Formal Approaches to Testing of Software (TESTCOM/FATES 2008), ser. Lecture Notes in Computer Science, Kenji Suzuki, Teruo Higashino, Andreas Ulrich, and Toru Hasegawa, Eds., vol. 5047. Berlin, Heidelberg: Springer, Jun. 2008, pp. 38–53.

Kaner, Cem; Falk, Jack & Hung Quoc Nguyen. 1999. Testing Computer Software. Wiley. 480 p.

Kaner, Cem; Bach, James & Pettichord, Bret. 2002. Lessons Learned in Software testing. A Context-Driven approach. Wiley. 286 p.

Kaner, Cem. 2001. Improving the Education of Software Testers. NSF Grant EIA0113539 ITR/SY+PE, p. 5, January 24, 2001. Available at: http://www.testingeducation.org/general/nsf_grant.pdf [Checked 13.2.2012]

Kaner, Cem. 2006a. Schools of software testing. Available at: http://kaner.com/?p=15. [Checked 23.3.2012]

Kaner, Cem. 2006b. Assessment objectives. Part 3: Adapting the Anderson & Krathwohl taxonomy for software testing. Available at: http://kaner.com/?p=33. [Checked 17.3.2014]

Kaner, Cem. 2008. Software Testing as Social Science. Presentation at STEP 2008. 50 pages. Available at: http://www.kaner.com/pdfs/KanerSocialScienceSTEP.pdf [Checked 13.2.2012]

Kaner, Cem & Bach, James. 2012. Context-Driven-Testing. http://context-driven-testing.com/ [Checked 2014-03-12]

Kaner, Cem. 2014. A proposal for an advanced certification in software testing. Dated 24.8.2015. Available at: http://kaner.com/?p=392 [Checked 21.3.2014]

Kaner, Cem & Fiedler, Rebecca L. 2014. Foundations of software testing. A BBST workbook. Context Driven Press. 230 p.

Kaner, Cem. 2015. Schools of Software Testing: A Debate with Rex Black. Dated 3.3.2014. Available at: http://kaner.com/?p=437 [Checked 22.9.2015]

Kaner, Cem & Fiedler, Rebecca L. 2016. Test design. A BBST workbook. Context Driven Press. 357 p.

Kantola, Anu. 2014. Branded revolutionaries: Circulated gurus as management tools in soft capitalism. European Journal of Cultural Studies 2014, Vol. 17(3) 258 –274. Available at: http://edge.sagepub.com/sites/default/files/Kantola_0.pdf

Kasurinen, Jussi. 2011. Software test process development. Acta Universitasis Lappeenrantaesis 443. 102 p.

Katara, Mika; Vuori, Matti & Jääskeläinen, Antti. 2015. Software Testing. Slides for software testing course TIE-21204. Available at: http://www.cs.tut.fi/~testaus/s2015/luennot/TIE-21204_2015_en.pdf

Kaupan liitto, Liikenne- ja viestintäministeriö, Tekes, Teknologiateollisuus & Verkkoteollisuus. 2016. Digibarometri 2016. Helsinki: Taloustieto Oy. 72 p. http://www.digibarometri.fi

Kennaley, Mark. 2010. SDLC 3.0. Beyond a Tacit Understanding of Agile. Towards the next generation of Software Engineering. Fourth Medium Press. 280 p.

Kenney, Martin & Zysman, John. 2016. The Rise of the Platform Economy. Issues in Science and Technology. National Academies of Sciences, Engineering, And Medicine; The University Of Texas At Dallas, Arizona State University. P. 61-69. Available at: http://www.brie.berkeley.edu/wp-content/uploads/2015/02/Kenney-Zysman-The-Rise-of-the-Platform-Economy-Spring-2016-ISTx.pdf

Kim, Gene; Behr, Kevin & Spafford, George. 2014. The Phoenix Project: A Novel about IT, DevOps, and Helping Your Business Win. It Revolution Press. 382 p.

Kniberg, Henrik. 2011. Lean from the Trenches. Managing Large-Scale Projects with Kanban. Pragmatic Bookshelf. 157 p.

Kone. 2016. KONE People Flow Intelligence solutions. http://www.kone.com/en/solutions/products/people-flow-intelligence/. [Checked 16.2.2016]

Koen, Peter A.; Ajarnian, Greg M.; Boyce, Scott; Clamen, Mien; Fisher, Eden; Fountoulakis, Stavros; Johnson, Albert; Puri, Pushpinder & Seibert, Rebecca. 2002. Fuzzy Front End: Effective Methods, Tools, and Techniques. In: Belliveau, Pul; Griffin, Abbie & Somermeyer, Stephen (ed.). 2002. The PDMA ToolBook 1 for New Product Development. John Wiley & Sons; 1 edition (4 April 2002). p. 5-35.

Koomen, Tim & Pol, Martin. 1999. Test Process Improvement. A practical step-up guide to structured testing. Addison-Wesley. 215 p.

Korhonen, Hannu. 2016. Evaluating Playability of Mobile Games with the Expert Review Method. Academic Dissertation. School of Information Sciences, University of Tampere. Dissertations in Interactive Technology, Number 24 Tampere 2016. 170 p. + appendices. http://urn.fi/URN:ISBN:978-952-03-0196-5

Koskinen, Johannes; Vuori, Matti & Katara, Mika. 2012. Safety Process Patterns: Demystifying Safety Standards. Proceedings of the IEEE CS International Conference on Software Science, Technology, and Engineering (SwSTE 2012), IEEE CS, Herzlia, Israel, June 2012.

Krathwohl, David R. 2002. A Revision of Bloom’s Taxonomy: An Overview. Theory Into Practice, Volume 41, Number 4, Autumn 2002. 8 p.

Kruchten, Philippe. 1995. Architectural Blueprints — The “4+1” View Model of Software Architecture. IEEE Software 12 (6), pp. 42-50. Available at: http://www.cs.ubc.ca/~gregor/teaching/papers/4+1view-architecture.pdf

Kruchten, Philippe. 2011. The Elephants in the Agile Room. Philippe Kruchten's Weblog. Available at: http://pkruchten.wordpress.com/2011/02/13/the-elephants-in-the-agile-room

Kuhanen, Pirjo; Vuori, Matti; Poranen, Timo; Pippola, Toni; Kairamo, Ville; Raisamo, Roope & Saarinen, Jukka P. 2012. The new way of teaching a project work – an open innovation platform for students. Poster presentation. EAIR 34th Annual Forum 2012, Stavanger, Norway, 5-8 September 2012.

Kumar, Janaki. 2013. Gamification at Work: Designing Engaging Business Software. In: Marcus, A. (Ed.): DUXU/HCII 2013, Part II, LNCS 8013, p. 528–537, 2013.

Kurtz, C. F. & Snowden, D. J. 2003. The New Dynamics of Strategy: sense-making in a complex-complicated world, IBM Systems Journal, Fall 2003. p. 462-483.

Kuusinen, Kati. 2015. Integrating UX Work in Agile Enterprise Software Development. Tampere University of Technology. Publication;1339. 98 p + appendices.

Leeds, Herbert D. & Weinberg, Gerald M. 1961. Computer programming fundamentals. McGraw-Hill Book Company. 368 p.

Leonard-Barton, Dorothy. 1995. Wellsprings of Knowledge: Building and Sustaining the Sources of Innovation. Harvard Business School Press. 334 p.

Lethbridge, Timothy C.; LeBlanc, Richard J. Jr.; Sobel, Ann E. Kelley; Hilburn, Thomas B.; & Díaz-Herrera, Jorge L. 2006. SE2004: Recommendations for Undergraduate Software Engineering Curricula. IEEE Software, November/December 2006, p. 19-25.

Lindell, Rikard. 2014. Crafting interaction: The epistemology of modern programming. Personal and Ubiquitous Computing (2014) 18:613–624. DOI 10.1007/s00779-013-0687-6.

Lowe, Robert A. & Ziedonis, Arvids, A. 2006. Overoptimism and the Performance of Entrepreneurial Firms. Management Science, Vol. 52, No. 2, Entrepreneurship (Feb., 2006), pp. 173-186.

Madhavan, Ravindranath & Grover, Rajiv. 1988. From Embedded Knowledge to Embodied Knowledge: New Product Development as Knowledge Management. Journal of Marketing; Oct 1988, Vol 62 (October 1988), p. 1-12.

Mei, Sarah. 2013. Why You Should Never Use MongoDB. http://www.sarahmei.com/blog/2013/11/11/why-you-should-never-use-mongodb [Checked 2016-08-12]

Meristö, Tarja; Manninen, Anneli & Laitinen, Jukka.. 2012. Foresight in practice – methodological lessons from business in the field of technology industry. Maintenance Problems, 2012, 4, p. 129–137.

Merkel, Robert & Kanij, Tanjila. 2010. Does the Individual Matter in Software Testing? Swinburne University of Technology. Faculty of Information and Communication Technologies. Center for Software Analysis and Testing Technical Report. 19 p.

Mielityinen, Ida (toim.). 2009. Suomi tarvitsee maailman parasta insinööriosaamista. Tekniikan akateemisten liitto. 70 p. Available at: http://www.tek.fi/ci/tekstra/opetuksen_laatu_final.pdf. [Checked 13.2.2012]

Miller, Christopher W. 2002. Hunting for Hunting Grounds: Forecasting the Fuzzy Fron Front End. In: Belliveau, Pul; Griffin, Abbie & Somermeyer, Stephen (ed.). 2002. The PDMA ToolBook 1 for New Product Development. John Wiley & Sons; 1 edition (4 April 2002). p. 37-62.

Mintzberg, Henry. 2005. Managers not MBAs. A hard look at the soft practice of managing and management development. Bettett-Koehler Publishers, Inc. 464 p.

Mohan, Kannan & Jain, Radhika. 2008. Using traceability to mitigate cognitive biases in software development. Communications of the ACM, September 2008, Vol. 51, No. 9, p. 110-114.

Modig, Niklas & Åhlström, Pär. 2013. This is Lean. Resolving the efficiency paradox. Rheologica Publishing. 168 p.

Moll, Melanie. The Quintessence of Intercultural Communication. Springer. 119 p.

Moore, David R.; Cheng, Mei-I; Dainty, Andrew R.J.. 2002. Competence, competency and competencies: performance assessment in organisations. Work Study, Vol. 51 Iss: 6, pp. 314-319.

Moore, David T. 2011. Sensemaking. A Structure for an Intelligence Revolution. National Defence Intelligence College. 195 p.

Moore, Geoffrey A.. 2013. Crossing the chasm. HarperBusiness; 3 edition. 288 p.

Moser, Raimund; Abrahamsson, Pekka; Pedrycz, Witold; Sillitti, Alberto and Succi, Giancarlo. 2007. A Case Study on the Impact of Refactoring on Quality and Productivity in an Agile Team In: B. Meyer, J.R. Nawrocki, and B. Walter (Eds.): CEE-SET 2007, LNCS 5082, pp. 252–266, 2008.

Mulder, Martin. 2011. The concept of competence: blessing or curse? In: Torniainen, Ilona; Mahlamäki-Kultanen, Seija; Nokelainen, Petri & Paul Ilsley (eds.). 2011. Innovations for Competence Management. Conference Proceedings. LAMK Series C Articles, reports and other current publications, part 84. pp. 11-24.

Märijärvi, Jukka; Hokkanen, Laura; Komssi, Marko; Kiljander, Harri; Xu, Yueqiang; Raatikainen, Mikko; Seppänen, Pertti; Heininen, Jari; Koivulahti-Ojala, Mervi; Helenius, Marko & Järvinen, Janne. 2016. The cookbook for successful internal startups. Digile, N4S. 95 p.

Neogames. 2016. Tietoa toimialasta. http://www.neogames.fi/tietoa-toimialasta/ [Checked 2016-08-16]

Nicholson, Scott. 2012. A User-Centered Theoretical Framework for Meaningful Gamification. Paper Presented at Games+Learning+Society 8.0, Madison, WI, USA. 7 p.

Nielsen, Jakob. 1993. Usability engineering. Morgan Kaufmann. 362 p.

Nielsen, Jakob. 1994. Enhancing the Explanatory Power of Usability Heuristics. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '94), p. 152–158. New York, NY, USA: ACM. doi: 10.1145/191666.191729

The NIST definition of Cloud Computing. 2011. National Institute of Standards and Technology Special Publication 800-145. Available at http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf

Nonaka, Ikujiro & Takeuchi, Hirotaka. 1995 The Knowledge-Creating Company. How Japanese Companies Create the Dynamics of Innovation. Oxford University Press. 284 p.

Oivallus. 2011. Oivallus – loppuraportti. Elinkeinoelämän keskusliitto. 42 p. Available at: http://ek.multiedition.fi/oivallus/fi/liitetiedostot/Oivallus_loppuraportti_web.pdf [Checked 14.2.2012]

Oskam, I. F. 2009. T-shaped engineers for interdisciplinary innovation: an attractive perspective for young people as well as a must for innovative organisations. In 37th Annual Conference–Attracting students in Engineering, Rotterdam, The Netherlands, p. 1-4.

Osono, Emi; Shimizu, Norihiko, Takeuchi, Hirotaka & Dorton, John Kyle. 2008. Extreme Toyota. Radical Contradictions that drive success at the world's best manufacturer. Wiley. 306 p.

OWASP. 2016a. Welcome to OWASP. The free and open software security community. https://www.owasp.org/index.php/Main_Page. [Checked 27.1.2016]

OWASP. 2016b. Appendix A: Testing Tools.
https://www.owasp.org/index.php/Appendix_A:_Testing_Tools. [Checked 27.1.2016]

OWASP. 2016c. M-Tools. OWASP Mobile Security Project
https://www.owasp.org/index.php/OWASP_Mobile_Security_Project#tab=M-Tools [Checked 27.1.2016]

PEST analysis. 2014. Wikipedia article. http://en.wikipedia.org/wiki/PEST_analysis [Checked 29.1.2014]

Pettichord, Bret. 2007. Schools of Software Testing. 33 p.

Pippola, T., Poranen, T., Vuori, M., Kairamo, V. and Tuominiemi, J. 2012. Teaching Innovation Projects in Universities at Tampere, in Proceedings of the ICEE 2012. 8 p.

Pirsig, Robert. 1999. Zen and the art of motorcycle maintenance. 25th Anniversary Edition. Vintage.436 p.

Pohjolainen, Pentti A. 2007. The Development of Software Testing in Finland 1950–2000. History of Nordic Computing 2. Volume 303 of the series IFIP Advances in Information and Communication Technology. pp 271-282.

Prahalad, C.K. & Hamel, G. 1990. The core competence of the corporation. Harvard Business Review, v. 68, no. 3, p. 79–91.

Puusa, Anu. 2011. Laadullisen aineiston analysointi. In: Menetelmäviidakon raivaajat – perusteita laadullisen tutkimuslähestymistavan valintaan. JTO Johtamistaidon opisto, p. 114 - 125.

Puusa, Anu & Juuti, Pauli. 2011. Laadullisen lähestymistavan yleistyminen kulttuurinäkökulman myötä. In: Menetelmäviidakon raivaajat – perusteita laadullisen tutkimuslähestymistavan valintaan. JTO Johtamistaidon opisto, p. 31 - 46.

QFD Institute. 2016. http://www.qfdi.org/ [Checked 24.10.2016]

Rahkamo, Susanna. 2016. The Road to Exceptional Expertise and Success. A Study of the Collective Creativity of Five Multiple Olympic Gold Medalists. Aalto University publication series. DOCTORAL DISSERTATIONS 257/2016. http://urn.fi/URN:ISBN: 978-952-60-7177-0. 233 p.

Redmill, Felix. 2004. Exploring risk-based testing and its implications. Software Testing, Verification and Reliability. 2004; 14. p. 3 - 15.

Reinertsen, Donald G. 2009. The Principles of Product Development Flow. Second Generation Lean Product Development. Celeritas Publishing. 294 p.

The R Foundation. 2014. The R Project for Statistical Computing. Project web site. Available at: https://www.r-project.org/. [Checked 14.1.2016]

Ries, Eric. 2011. The Lean Startup. How today' Enterpreneurs Use Continuous Innovation to Create Radically Sucessful Businesses. Crown Business, New York. 312 p.

Rogers, Everett M. 2003. 1962. Diffusion of Innovations. Simon & Schuster International; 5th Revised edition. 512 p.

Riungu-Kalliosaari, Leah; Taipale, Ossi. & Smolander, Kari. 2012. “Testing in the Cloud: Exploring the Practice, ”Special issue on Software Engineering for Cloud Computing, IEEE Software, March/April 2012.

Rosenbaum, Stephanie. 2008. The Future of Usability Evaluation: Increasing Impact on Value. Human-Computer Interaction Series, 2008, Part III, p. 344-378.

Rosenberg, Linda; Stapko, Ruth & Gallo, Al. 1999. Risk-based object oriented testing. In Twenty-Fourth Annual Software Engineering Workshop. NASA, Software Engineering Laboratory. 6 p.

Rothman, Johanna. 2007. Manage It! Your Guide to Modern, Pragmatic Project Management. Pragmatic Bookshelf. 351 p.

Runeson, Per. 2006 A Survey of Unit Testing Practices. IEEE Software, Volume: 23 Issue:4, pp. 22-29.

Runeson, Per & Hölst, Martin. 2009. Guidelines for conducting and reporting case study research in software engineering. Empirical Software Engineering, 14(2), 131-164. 10.1007/s10664-008-9102-8.

Schein, Edgar. 2004. Organizational culture and leadership. Jossey-Bass. 437 p.

Schmidt, Eric & Rosenberg, Jonathan. 2014. How Google Works. John Murray. 286 p.

Scrum Guides. 2015. The Scrum Guide. http://www.scrumguides.org/scrum-guide.html

SFS-EN 614-2. 2009. Safety of machinery. Ergonomic design principles. Part 2: interactions between the design of machinery and work tasks. Suomen standardisoimisliitto SFS. 29 p.

SFS-EN ISO/IEC 17024. 2012. Conformity assessment. General requirements for bodies operating certifications of persons (ISO/IEC 17024:2012). 24 p.

Shippmann, Jeffery S.; Ash, Ronald A.; Batjtsta, Mariangela; Carr, Linda; Eyde, Lorraine D.; Besketh, Beryl; Kehoe, Jerry; Pearlman, Kenneth; Prien, Erich P. & Sanchez, Juan I.. 2000. the practice of competency modeling. Personnel Psychology. Volume 53, Issue 3, p. 703–740.

Singer, Leif & Schneider, Kurt. 2012. It Was a Bit of a Race: Gamification of Version Control. Proceedings of GAS 2012, Zurich, Switzerland. p. 5-8.

Sivasubramanian, Narashima Boopathi. 2016. Managing across cultures with Cultural Intelligence Quotient (CQ). Study of Finnish business leaders experience in India. Acta Wasaensia, 363; Business Administration, 144. 360 p.

Sommarberg, Matti. 2016. Digitalization as a Paradigm Changer in Machine-Building Industry. Tampere University of Technology. Publication; Vol. 1436. Tampere University of Technology. 211 p. + appendices. http://urn.fi/URN:ISBN:978-952-15-3875-9

Spohrer, Jim; Golinelli, Gaetano M.; Piciocchi, Paolo & Bassano, Clara. 2010. An Integrated SS-VSA Analysis of Changing Job Roles. Service Science 2(1-2), p. 1-20.

Stanford University. 2016. One Hundred Year Study on Artificial Intelligence (AI100). Stanford University, September 2016. 52 p. https://ai100.stanford.edu/sites/default/files/ai_100_report_0901fnlc_single.pdf

Stephens, Matt & Rosenberg, Doug. 2003. Extreme Programming Refactored: The Case Against XP. Apress, Expert's Voice series. 432 p.

StudyInEurope.eu. 2015. The ECTS System. http://www.studyineurope.eu/ects-system [Checked 26.11.2015]

Surakka, Sami. 2005. Needs assessment of software systems graduates. Helsinki University of Technology, Department of Computer Science and Engineering. Laboratory of Information Processing Science A. TKK-TKO-A43. 242 p.

Sydänmaalakka, Pentti. 2014. Yllätyksellinen ja kompleksinen tulevaisuus; miten selvitä tulevaisuudesta [The surprising and complex future; how to survive in the future]. In: Sydänmaalakka, Pentti (ed.). 2014. Tulevaisuuden johtaminen [Future management]. Pertec. p. 18-37.

Systä, Kari; Vuori, Matti; Järvinen, Hannu-Matti; Ahtee, Tero & Stén, Harri. 2016. Project types and industrial collaboration in project-based learning. 44th SEFI Conference, 12-15 September 2016, Tampere, Finland. Accepted paper. 13 p.

SWECOM, 2014. The Software Engineering Competency Model (SWECOM). https://computer.centraldesktop.com/home/viewfile?guid=53076640805C22BA4AC16312FABAAB1D33917BDEA&id=30862727 [Checked 30.4.2014]

TeliaSonera Finland. 2015. Uuskasvun polut – Digitalisaation lupaus. B2D. Business to Digital. 279 p.

TestausOSY – FAST. Finnish Association of Software Testing. Community home page: http://wwww.testausosy.fi/

Thompsen, Joyce A. 2003. Achieving Return on Critical Talent – A Case Study of a Software Development Organization. In: Proc. Engineering Management Conference (IEMC '03), IEEE CS 2003, 67-71.

Tippins, Michael J. & Sohi, Ravipreet S. 2003. It Competency and Firm Performance: Is Organizational Learning a Missing Link? Strategic Management Journal, Vol. 24, No. 8 (Aug., 2003), p. 745-761.

Tiwana, Amrit; Konsynski, Benn & Bush, Ashley A. 2010. Research Commentary -- Platform Evolution: Coevolution of Platform Architecture, Governance, and Environmental Dynamics. Journal of Information Systems Research Volume 21 Issue 4, December 2010, p. 675-687

Twitter. 2013. User interface of the service at http://www.twitter.com.

van de Kamp, Pepijn. 2014. Holacracy – A Radical Approach to Organizational Design. In: Elements of the Software Development Process – Influences on Project Success and Failure, Chapter: 2, Publisher: University of Amsterdam, Editors: Hans Dekkers, Wil Leeuwis, Ivan Plantevin, pp.13-26.

Van Veenendaal, Erik (ed.) 2010. Standard glossary of terms used in Software Testing. International Software Testing Qualifications Board. 51 p.

van Veenendaal, Erik (ed.). 2012. Test Maturity Model integration (TMMi). Release 1.0. TMMi Foundation. 219 p.

Wania, Christine E.; Atwood, Michael E.; McCain, Katherine W. 2006. How do design and evaluation interrelate in HCI research? Proceedings of the 6th conference on Designing Interactive systems. ACM New York. p. 90-96.

Weick, Karl E. 1979. The social psychology of organizing (2nd ed.). Reading, MA: Addison-Wesley. 294 p.

Weick, Karl E. 1999. Conclusion: Theory Construction as Disciplined Reflexivity: Tradeoffs in the 90s. The Academy of Management Review, Vol. 24, No. 4 (Oct., 1999), pp. 797-806. http://www.jstror.org/stable/259.355

Weick, Karl. E. & Sutcliffe, Kathleen M. 2007. Managing the Unexpected. Resilient Performance in an Age of Uncertainty. Second Edition. Wiley. 194 p.

Weingerg, Gerald. 1988. Understanding the Professional Programmer. 240 p.

Weingerg, Gerald. 1992. Quality Software Management. Volume 1. Systems Thinking. Dorset House Publishing. 318 p.

Weingerg, Gerald. 1998. The Psychology of Computer Programming. Silver Anniversary Edition. Dorset House Publishing. 292 p.

Vesiluoma, Sari. 2009. Understanding and Supporting Knowledge Sharing in Software Engineering. Tampere University of Technology, Publication 843. 158 p. Available at: http://dspace.cc.tut.fi/dpub/bitstream/handle/123456789/6174/vesiluoma.pdf

Whittaker, James A. 2001. How to break software. A Practical Guide to Testing. Addison-Wesley. 178 p.

Whittaker, James A. & Thompson, Herbert H. 2004. How to break software security. Pearson Education Inc. 185 p.

Whittaker, James A. 2010. Tips, tricks, tours, and techniques to guide test design. Addison-Wesley. 224 p.

Whittaker, James; Arbon, Jason & Carollo, Jeff. 2012. How Google tests software. Addison-Wesley. 281 p.

Williams, B. & Figueiredo, J. 2011. Strategy and technology management: An innovation-leader case study. 2011 IEEE International Technology Management Conference (ITMC), 27-30 June 2011. pp. 806 – 811.

Winterton, Jonathan; Delamare-Le Deist, Françoise & Stringfellow, Emma. 2005. Typology of knowledge, skills and competences: clarification of the concept and prototype. Centre for European Research on Employment and Human Resources Groupe ESC Toulouse. Research report elaborated on behalf of Cedefop/Thessaloniki. Final draft. 111 p.

Wohlic, Claes; Runeson, Per; Höst, Martin; Ohlsson, Magnus C.; Regnell, Björn & Wesslén, Anders. 2000. Experimentation is software engineering. Kluwe Academic Publishers. 204 p.

Von Henning, Kagermann & Wolf-Dieter Lukas. 2011. Industrie 4.0: Mit dem Internet der Dinge auf dem Weg zur 4. industriellen Revolution. http://www.vdi-nachrichten.com/Technik-Gesellschaft/Industrie-40-Mit-Internet-Dinge-Weg-4-industriellen-Revolution

Wong, W. Eric; Bertolino, Antonia; Debroy, Vidroha; Mathur, Aditya; Offutt, Jeff & Vouk, Mladen. 2011. Teaching software testing: Experiences, lessons learned and the path forward. 24th IEEE-CS Conference on Software Engineering Education and Training (CSEE&T). pp. 530-534.

Woodruffe, Charles. 1993. What Is Meant by a Competency? Leadership & Organization Development Journal, Vol. 14 Iss: 1, pp.29 – 36.

Vuolle, Harri & Alanen, Esa. 2016. ICT-koulutustarveselvityksen loppuraportti 2015. Tampereen kaupunkiseudun elinkeino- ja kehitysyhtiö Tredea Oy. 13 s.

Vuori, Matti. 1994. Oppimisesta ja suunnittelusta, suunnitteluprosessi oppimisprosessina suunnittelijalle ja suunnitteluun osallistujille. Hyvä suunnittelukäytäntö -tutkimusohjelma, projekti Suunnittelun ergonomisen laadun parantaminen kokoonpanoteollisuudessa. Available at: http://www.mattivuori.net/julkaisuluettelo/liitteet/opetus2.pdf

Vuori, Matti. 1998. Tuotekehityksen kokonaisvaltaisella kehittämisellä kestäviä tuloksia. Esitys KATTI- ja KÄYPRO-hankkeiden seminaarissa ”Käyttäjä- ja käyttötiedot tuotekehityksessä” 6.10.1998. Esityksen jälkeinen, täydennetty versio. VTT Automation. Available at: http://www.mattivuori.net/julkaisuluettelo/liitteet/semv1098.pdf

Vuori, Matti & Kivistö-Rahnasto, Jouni. 2000. Tulevaisuuden tuotteiden ja käyttöliittymien kehittäminen. Opas yrityksessä sovellettavaan kehittämisprojektiin. VTT Automation. 21 p. Available at: http://www.mattivuori.net/julkaisuluettelo/liitteet/tulev-kl-pikaopas.pdf.

Vuori, Matti; Kivistö-Rahnasto, Jouni & Toivonen, Sirra. 2001. Developing of Future User Interfaces. In: Smart machines and systems – Recent Advances in Mechatronics in Finland. Helsinki University of Technology, Publications in Machine Design 1/2001, p. 345-384.

Vuori, Matti. 2004. Work profile and competencies of a test engineer. Training slide set. 26 p. Available at: http://www.mattivuori.net/julkaisuluettelo/liitteet/work_profile_of_test_engineer.pdf

Vuori, Matti. 2009. Miten ollaan hyvä alihankkija? [How to be a good subcontractor?] Training slide set. 36 s.

Vuori, Matti. 2010. Tietopohjainen testaus ja silmien avaaminen kohteen ilmiöille. [Knowledge-based testing and opening eyes for the phenomena in the test target] 8 p. Available at: http://www.mattivuori.net/julkaisuluettelo/liitteet/tietopohjainen_testaus.pdf.

Vuori, Matti. 2010b. How to recognize an expert? 5 p. Available at: http://www.mattivuori.net/julkaisuluettelo/liitteet/how_to_recognize_an_expert.pdf

Vuori, Matti. 2010c. Työ intohimona – caseina testaus, käytettävyys ja riskienhallinta. Available at: http://www.mattivuori.net/julkaisuluettelo/liitteet/tyo_intohimona.pdf

Vuori, Matti. 2010d. Testaajan eettiset periaatteet. Slide set. 16 p. Available at: http://www.mattivuori.net/julkaisuluettelo/liitteet/testaajan_eettiset_periaatteet.pdf

Vuori, Matti. 2010e. Testaus organisaatiossa – eri osapuolten näkökulmia laadunvarmistukseen ja testaamiseen. Slide set. 24 p. Available at: http://www.mattivuori.net/julkaisuluettelo/liitteet/nakokulmia_testaamiseen.pdf

Vuori, Matti; Virtanen, Heikki, Koskinen, Johannes & Katara, Mika. 2011. Safety Process Patterns in the Context of IEC 61508-3. Tampere University of Technology. Department of Software Systems. Report 15. 128 p. Available at the Tampere University of Technology DPub system: http://dspace.cc.tut.fi/dpub/.

Vuori, M. 2011a. Agile Development of Safety-Critical Software. Tampere University of Technology, Department of Software Systems. Report 14. Tampere, Tampere University of Technology. 95. ISBN (PDF): 978-952-15-2595-7 URN: http://URN.fi/URN:NBN:fi:tty-2011061414702

Vuori, Matti. 2011b. Reflections on principles of good testing in the context of safety-critical development. In: Malm, Timo; Vuori, Matti; Rauhamäki, Jari; Vepsäläinen, Timo; Koskinen, Johannes; Seppälä, Jari; Virtanen, Heikki; Hietikko, Marita & Katara, Mika: "Safety-critical software in machinery applications", VTT, vol. VTT Tiedotteita – Meddelanden – Research Notes, no. 2601, pp. 111 p. + app. 10 p., Espoo, 2011.

Vuori, Matti. 2011c. Tiedosta ja tietämyksen luomisesta testauksessa. Presentation in a TestausOSY seminar. Available at:
http://www.mattivuori.net/julkaisuluettelo/liitteet/tieto_testauksessa.pdf

Vuori, Matti. 2012. Grounded Theory. A description and some analysis of the method, mostly based on Corbin, Juliet & Strauss, Anselm. 2008. Basics of Qualitative Research, 3rd edition. Techniques and Procedures for Developing Grounded Theory. SAGE Publications. 379 p. 33 p. Available at:
http://mattivuori.net/extra/grounded_theory_vuori.pdf

Vuori, Matti. 2013. About the applicability and benefits of robot assisted testing. RATA project report. Available at:
http://www.cs.tut.fi/~ratac/rata/PublicationsAndDownloads/about-applicability-of-robot-assisted-testing.pdf

Vuori. Matti. 2014a. Testing of human-like robots. RATA & ATAC project report. Available at: http://www.cs.tut.fi/~ratac/rata/PublicationsAndDownloads/testing-human-like-robots.pdf

Vuori, Matti. 2014b. Finnish testing competence in the future – The view of the testing community. Version for Testing Day (Testauspäivä) 2014-06-03; English translation. Testing Day 2014. Tampere 3.6.2014. Available at:
http://www.cs.tut.fi/tapahtumat/testaus14/kalvot/vuori_testing_competence_survey_slides.pdf

Vuori, Matti. 2014c. Support from testing for fast and dynamic software development. ATAC report. 21 pages.
http://www.cs.tut.fi/~ratac/atac/ATAC_report_testing_in_fast_product_development.pdf

Vuori, Matti. 2014d. Trajectories of testing – situation in 2014. Slide set. 110 p. http://www.mattivuori.net/julkaisuluettelo/liitteet/trajectories_in_testing_2014.pdf

Vuori, Matti. 2015. Suhteestamme uuden teknologian laatuun ja sen varmistamiseen [About our relation to the quality of new technology and to assuring it]. Laatu & Testaus 1/2015. p. 4-10. http://testausosy.fi/wp-content/uploads/2015/04/LT-Vol4Ed1.pdf

Välimäki, Antti; Kääriäinen, Jukka & Koskimies, Kai. 2009. Global Software Development Patterns for Project Management. Proceedings of EuroSPI 2009, CCIS 42, pp. 137-148.

Whittaker, James A. 2008. Exploratory software testing. Tips, tricks and techniques to guide test design. Addison-Wesley. 224 p.

Wikipedia. 2015. Innovaatio [Innovation]. http://fi.wikipedia.org/wiki/Innovaatio. Checked 6.4.2015.

Wikipedia. 2016. The Toyota Way. *http://en.wikipedia.org/wiki/The_Toyota_Way.* Checked 18.2.2016

World Economic Forum. 2015a. The Human Capital Report 2015. Employment, Skills and Human Capital Global Challenge Insight Report. World Economic Forum. 319 p.

World Economic Forum. 2015b. The Future of Jobs. Employment, Skills and Workforce Strategy for the Fourth Industrial Revolution. World Economic Forum. 147 p.

Ylén, Mira. 2015. Labor, Amor, Vincit – Ohjelmistotalon käytännöt ammatillisen toimijuuden tilana. Aalto-yliopisto, Kauppakorkeakoulu. Pro gradu –tutkielma. 94 p.

APPENDIX 1: Contents of ISTQB Foundation syllabus, 2011

Source: ISTQB. 2011a. Certified Tester. Foundation Level Syllabus. Version 2011. 78 p. Available at: http://istqb.org

1. Fundamentals of Testing (K2)

1.1 Why is Testing Necessary (K2)

1.1.1 Software Systems Context (K1)

1.1.2 Cause s of Software Defects (K2)

1.1.3 Role of Testing in Software Development, Maintenance and Operations (K2)

1.1.4 Testing and Quality (K2)

1.1.5 How Much Testing is Enough? (K2)

1.2 What is Testing? (K2)

1.3 Seven Testing Principles (K2)

1.4 Fundamental Test Process (K1)

1.4.1 Test Planning and Control (K1)

1.4.2 Test Analysis and Design (K1)

1.4.3 Test Implementation and Execution (K1)

1.4.4 Evaluating Exit Criteria and Re porting (K1)

1.4.5 Test Closure Activities (K1)

1.5 The Psychology of Testing (K2)

1.6 Code of Ethics

2. Testing Throughout the Software Life Cycle (K2)

2.1 Software Development Models (K2)

2.1.1 V-mod el (Sequential Development Model) (K2)

2.1.2 Iterative-increment al Development Models (K2)

2.1.3 Testing within a Life Cycle Model (K2)

2.2 Test Levels (K2)

2.2.1 Component Testing (K2)

2.2.2 Integration Testing (K2)

2.2.3 System Testing (K2)

2.2.4 Acceptance Testing (K2)

2.3 Test Types (K2)

2.3.1 Testing of Function (Functional Testing) (K2)

2.3.2 Testing of Non-functional Software Characteristics (Non-functional Testing) (K2)

2.3.3 Testing of Software Structure/Architecture (Structural Testing) (K2)

2.3.4 Testing Related to Changes: R e-testing and Regression Testing (K2)

2.4 Maintenance Testing (K2)

3. Static Techniques (K2)

3.1 Static Techniques and the Test Process (K2)

3.2 Review Process (K2)

3.2.1 Activities of a Formal Review (K1)

3.2.2 Roles and Responsibilities (K1)

3.2.3 Types of Reviews (K2)

3.2.4 Success Factors for Reviews (K2)

3.3 Static Analysis by Tools (K2)

4. Test Design Techniques (K4)

4.1 The Test Development Process (K3)

4.2 Categories of Test Design Techniques (K2)

4.3 Specification-based or Black-box Techniques (K3)

4.3.1 Equivalence Partitioning (K3)

4.3.2 Boundary Value Analysis (K3)

4.3.3 Decision Table Testing (K3)

4.3.4 State Transition Testing (K3)

4.3.5 Use Case Testing (K2)

4.4 Structure- based or White-box Techniques (K4)

4.4.1 Statement Testing and Coverage (K4)

4.4.2 Decision Testing and Coverage (K4)

4.4.3 Other Structure-based Techniques (K1)

4.5 Experience-based Techniques (K)

4.6 Choosing Test Techniques (K2)

5. Test Management (K3)

5.1 Test Organization (K2)

5.1.1 Test Organization and Independence (K2)

5.1.2 Tasks of the Test Leader and Tester (K1)

5.2 Test Planning and Estimation (K3)

5.2.1 Test Planning (K2)

5.2.2 Test Planning Activities (K3)

5.2.3 Entry Criteria (K2)

5.2.4 Exit Criteria (K2)

5.2.5 Test Estimation (K)

5.2.6 Test Strategy, Test Approach (K2)

5.3 Test Progress Monitoring and Control (K2)

5.3.1 Test Progress Monitoring (K1)

5.3.2 Test Reporting (K2)

5.3.3 Test Control (K2)

5.4 Configuration Management (K2)

5.5 Risk and Testing (K2)

5.5.1 Project Risks (K2)

5.5.2 Product Risks (K2)

5.6 Incident Management (K3)

6. Tool Support for Testing (K2)

6.1 Types of Test Tools (K2)

6.1.1 Tool Support for Testing (K2)

6.1.2 Test Tool Classification (K2)

6.1.3 Tool Support for Management of Testing and Tests (K1)

6.1.4 Tool Support for Static Testing (K1)

6.1.5 Tool Support for Test Specification (K1)

6.1.6 Tool Support for Test Execution and Logging (K1)

6.1.7 Tool Support for Performance and Monitoring (K1)

6.1.8 Tool Support for Specific Testing Needs (K1)

6.2 Effective Use of Tools: Potential Benefits and Risks (K2)

6.2.1 Potential Benefits and Risks of Tool Support for Testing (for all tools) (K2)

6.2.2 Special Considerations for Some Types of Tools (K1)

6.3 Introducing a Tool into an Organization (K1)

APPENDIX 2: Raw answers on the survey to Finnish testing community about future competences

Answers are translated from Finnish with minimum of editing aiming to keep the style as much like the original as possible, including non-optimal sentence structures and such. The order of the answers has been scrambled at the different parts of this attachment so that they cannot be combined and the respondent identified.

1) In your opinion, what would a good tester be in Finland in 2025 like? What are all the things that she does? What things is she good at? What are the special things that she can do? What does she concentrate in?

A good tester knows thoroughly the business area she tests in. Also a good tester must have information technology competences, or ICT competences, both on processes and tools. A good tester has the ability to see whole and question things.

Can use Finnish, English in writing and orally (more languages are a plus).

Skills are needed on information security testing, test automation and testing of electronic services – some coding skill would not be bad, just as she should have some “hacker skills”.

Must concentrate on the end user viewpoint in testing.

Must be able to efficiently analyse the volume of data. Skills that are emphasised are efficient use of tools and even making/tuning the tools.

A good tester sees the overall picture of testing requirements, takes the tasks without preoccupations and can apply her competences in ways that suit different projects (waterfall – agile, different testing tools).

Knowledge must be broad and even a basic tester needs to know the criteria for the testing at all test levels (cf. the foundation level ISTQB certification material, where all of that is tackled a little). Specialization in some sub-area is desirable, but there is a danger of getting stuck in that area.

Competences need to be developed continuously and the tester's work profile will in the future be continuously changing and continuously containing new things.

A combination of tester's mindset and coder's SW developing skills. Communication skills very important. Concentrates on test automation, CI/similar systems. Can understand and report about system's quality vs. report about testing. So, moving more from just testing towards quality assurance.

In 2015 [sic] a good tester knows how to avoid making unnecessary tests = test avoidance. She uses to testing (with same output) only half of the time that a good tester spent in 2013. Time savings do not only come from intelligent automation, but also from the skill of choosing only the test cases that produce value, that is, the ones that give comprehensive feedback about the quality of the tested product. Doing test analyses (= test need analysing) is a basic skill of a good tester.

In the product sense, Virtualized HW Platforms are a naturel working environment for a good tester.

Let us say as a side note that she probably has got her basic skills already during her studies (occupational education, Aalto, TUT etc.), so she knows what testing techniques to use at certain situation and above all knows why testing is done.

Technical testing competence is at a good level (automation, security, etc.). Tasks require specialized skills.

Each tester is an expert in his field (energy, financial, telecomm, etc.) and knows her own capabilities and the needs of the customer.

Can code (automation) and also understands the technical side of the manual testing.

Carries the title of testers' occupancy with pride.

In the terms of competence, she usually can work with test automation, and could code too, but wants to be a holistic ambassador for the quality in software projects, and therefore does all necessary things in a project.

____

Year’s 2025 tester is thinking, experimental, bold, curious, professional, and proud of her profession.

____

There are 11 years to 2025. Certainly testing environments get richer (they also become more complex). Interfaces to other systems and cooperation with different systems will increase. Most likely there will be new methods; some are to live for a longer time than others. Requirements for agile development and testing will grow.

____

A good tester would be adaptive and would learn easily. Projects with new technologies come at a rapid pace, with the duration of 3-12 months, so the tester may be taken along at the beginning, the middle or at the end of either. The tester should be able to apply her testing view in any environment, and to act according to the needs and good testing practices of any environment. Thus, the tester can participate in a project in a variety of tasks such as a project lead, documenter and even as a helper of the development. In some sense I would think that the “tester” and discrete role term could even disappear and the tasks will blend more than today. A good tester may not even know that she is in a “tester” role, but a good final result will tell about the competence. I believe that she will focus on solving uncertainties and bringing issues on the table as soon as early as possible in the project, so that the project can do something about them.

Of course, it would be nice to mention also the automation and modelling – maybe in a few years' time there will be development environments in which all integrates, though then the control of the toolchains will be another area of tasks.

____

Can do quality work peacefully. Does not resort to compromises in quality even when there is hurry. Gives more value than just testing. Ideates for helping business, other ICT and processes. Because the tester is in role with broad visibility to activities, she can share her views and experiences.

2) What are the most important differences with the current situation? What are the main differences in tasks and ways of working? What kind of (new?) competences are emphasised in the near future?

Agility and interaction skills. Ability to get along with different kinds of people.

One needs training and practical doing to do all that was mentioned above [Q1]. Myself I have not during the past 10 years needed much skills for security testing (something small obviously) or test automation. I don’t have any coding skills to build for example automatic tests. Not to mention hacker skills.

[Competence needs get focused]; someone who is already well familiar with the background of the test target is expected to fill role of a tester.

Nowadays the testers have been able to do testing in a quite straightforward way only at one test level and according to a certain model.

In the future focusing the competence is surely a good thing, but every tester needs to be able to adapt to various models (waterfall, agile…) and to testing tasks at various test levels.

Breadth of competence is emphasised: a tester needs to know “a little about everything” and continuously getting trained and continuous learning of new things will be emphasised in the work.

Seeing the whole, in my opinion in the future there will be less and less manual UI testers with a certain area of responsibility. Work will more be done in smaller teams, and then the multi-skilledness will be emphasized (automation, coding/scripting skills) and the importance of communication can never be emphasized too much.

Also, the ability to work in global teams and taking and carrying responsibility.

Because of the Earth's rotational speed seems only to accelerate, and thus the time spent on product development (also on testing) is reduced, it is obvious that a large part of the [testing] work must be based on test automation.

In the reality of continuous deliveries, a big part of testing is of regression nature. The amount of regression is growing like a snowball when the number of new functions increases in the product as a result of the development.

Test automation’s ability to run tests is limited. Test cases cannot be indefinitely added to automatic regression testing. What is therefore needed is someone, or something, to limit the number of tests. For her part, a good tester knows how to choose the right tests, but she also has access to the test automation, which has the intelligence to make similar choices. In part, this is based on the results of the previous regression tests, but also on the knowledge of how the development of new features has been subjected to the product.

In addition to this, the test automation component can on the basis of the test results, infer when the result is ok and when it is not. This saves time on the analysis done by the tester.

Even though today the execution of tests is largely automated, analysis of the results of the test runs requires a great deal of time. Test automation is not always able to say whether the result of ok or not. Not even nearly all of tests give results as BOOLEAN (1,0). For example in robustness-type testing analysis of test results is laborious and time consuming. There may not always be a requirement related to the test result, and then the result may be accepted if there is nothing serious happening. This problem is largely gotten rid of by 2025, thanks to the intelligence built into the test automation.

Virtualized HW Platforms (cloud) seems to be a present trend, so by 2015 it should be more widely everyday life. This is a big change for embedded SW product development. Is no longer necessarily to produce the whole package (HW + SW) by ourself, but the hardware can also come for a third party supplier. This changes significantly the ways of working, tasks and methods. Cooperation with third-party providers, perhaps new roles, new testing environments, perhaps more simulation (in current terms service virtualization), etc.

____

The tester is an expert in some particular area and her business competence is much more in-depth than is currently the case. This gives added value and secures the tester's future as the basic testing moves increasingly to foreign countries.

____

All in all, testers’ competence increases on test automation and/or in technical understanding of testing i.e. technical implementation of manual testing.

____

More about being involved in the project from the beginning to the end, and about more working together in the team with everyone, from the developers to the business owners (CIO, etc.).

____

I think that the same skills are needed in the future too.

The rate of development of things and the number "isms" are more likely to increase than decrease. The proportion of automation testing will increase. Similarly, the [number & role of] interfaces rises.

I guess the working life is constantly changing and the jobs are divided into different categories. There are the “basic testing tasks” which will be mostly done as micro tasks or outsourced to the cheapest to the ones who are easiest to employ. Then there are the upper level senior quality testing tasks, where the most challenging testing tasks are done: automatization, model-based testing etc. If and robotics/analytics will begin to be used more, the role of test manage may disappear. Possibly there will be new testing roles, with the task of parametrizing and configuring test robots or administering the tasks between the basic vs. quality testers. Learning, learning, learning is the most necessary thing. [There will be] short projects that need to jumped in suddenly. Therefore, to be able to find (at least in the Western countries) quality assurance tasks that require human work requires collecting a large personal knowledge arsenal.

The same things are now in the main roles, but there is a pressing hurry and offshore will do the testing at a lower hour price. But what is the total cost testing and what is the total cost of the system's life cycle? Especially in the demanding system areas such as some of the banking and insurance systems.

The tester can do quality work peacefully. Does not resort to compromising quality in a wrong way even when there is hurry. Gives more value than just doing testing. Ideates for helping business, other ICT and processes. Because the tester is in role with broad visibility to activities, she can share her views and experiences.

3) To justify the answer, tell a little about views about the tester’s future environment? (For example the ways the organisation works, the tools in use etc.). Limit as necessary – tell for example in what domain you see in your mind the tester you described.

I think testing should be able to be done physically from anywhere. If e.g. one is not able to get her holiday organised and rest of the family wants to go to Lapland or to Spain, she should be able to go along with the family and do the work from there.

____

The vision is based on the assumption that we are moving to the year 2025 in accordance to the idea of “need to get more with less”.

There will not be much coding anymore in 2025; the code comes from bots based on dictation and the architect then modifies it.

The tester is in the field, where she finds work, I pull out of the hat cybersecurity and sports gear that talk with each other.

____

I believe that testing services in companies will be handled in two possible ways – focused development of the company's internal testing competence or buying the testing competences from outside.

The targeted competence applies in my mind the companies that have a continuous, similar testing needs (pharmaceutical industry) and the use of outside expertise the project-based testing needs (the introduction of a new IT system).

____

Sure, testing is a little bit domain-specific, but on the other hand very universal, so I think the principles of quality assurance work regardless of the operational [business] environment.

____

Perhaps I explained that already.

____

Companies have more outsourced testing, which is often distributed abroad. Work in home country is increasingly monitoring, guidance and expert work. Organisation has a very small dedicated testing staff.

____

Testing will be done independently of time and place as separate modules, as all services are ultimately in the cloud or the equivalent.

____

A thoroughly agile organization, up to the CEOs. In any domain. Very disciplined action, maybe most closely resembles Lean.

____

A competent tester works where she is needed. She likes to solve problems. She is social. She knows what happens in technology.

____

I think that a tester can be seen in any domain where software testing is done. The interfaces/connections of computer systems to other systems increasing (at least not decreasing). Better integration between systems and their implementations will be needed than today.

____

Here, too, I would see a two-tier situation, on the other hand, there are a global huge companies where the test consultants, etc. will do the testing tasks, for any software project, anywhere in the world. The work is done either locally or virtually remotely from anywhere during those times when the customer needs it. On the other hand, maybe there is more room and perhaps more offerings for small testing expert companies or one-person projects – especially when one can show successes of past projects and multi-domain competences. One domain is not enough; there needs to be several of those. Sometimes funding of “own works” comes to mind; first one gets a sponsor for own competence and then offers the competence to the customers who need it. Web application services, both in private and public sector, can happen that way. Possibly crowdsourcing is on the increase; [there are] "user tests", after which one reacts to the found issues.

____

I am afraid that the above issues [Q1,2] are not seen in the eyes and thoughts of upper management. I am afraid that the environment gets so fragmented and into a multi-vendor environment that there will be horribly lot of handovers and at the same time one should control many systems and many technologies and the chaos of many processes and people. That will break many people.

4) For background, tell a little about yourself. How many years of experience from this area (testing, quality, product development) are your views based on?

____

Have been in the ICT sector for 24 years and in testing for 13 years. My roles have been tester, tes [answer clipped]

____

I have done testing and related things for 8 years and before that, now and then, 5 years.

~10 years as a tester.

I have been working in the field of testing for 3 years: two years as a tester and one year a test manager.

Around 15 years experience of quality assurance and testing tasks in various roles in project deliveries (tester, test manager, project manager) and also in business responsibility roles (team manager, business leader), both in Finland and [in an Asia country] (lived there for [years]).

Currently I am in charge of a quality assurance organisation which has in Finland around [more than hundred] people and globally around [several hundred] people.

R&D of tele systems from 1990, systems tester, test manager, project manager, line manager, test expert, senior specialist.

---

14 years experience, as tester 10 years, testing responsible 2,5 years and now test manager 1,5 years.

---

14 years of testing and in the background 15 years of coding.

---

20 years with testing. Many clients from consultant’s perspective; project and development tasks.

I have more than 15 years’ experience on the field of testing. Tester, test manager.

More than 30 years experience in IT field, software design, testing, customer support. Bachelor of Science. Age > 50. Unemployed.

————

S/w dev 15 years, as developer 3 years & in testing tasks 5 years (process, test manager, tester) & at times both

————

10 years in testing and test management, automating and QC [HP Quality Center] admin and in process development

————

5) Other thoughts? Some other views or comments?

————

The current trend seems to be outsourcing and, with it, working in a multicultural environment seems to be inevitable. Because of this, cultural differences, in particular working culture, should in the future paid considerably more attention to than is currently the case. A Finn is committed to her work, while to a foreigner, her salary is more important than the work itself, and with it, she does not commit to the work in the same way as the Finns. This should be told to the management that is considering outsourcing of some company function.

————

There are certainly many different kinds of testers; [the one I described] was a description of the "average".

————

I am ready to respond to the e.g. surveys. But I do not want my name to be made public.

————

Would the future hit be robotics – it is said to replace doers but also managers. Will a society’s turning point begin after which only a minority works and the rest are freelancing and having civil wage? Or the game industry – will there be a new games and applications for the devices every day from the same companies and rate of consumption accelerates. As such, the situation in the future is positive Data processing is needed and there will be more and more software: in cars, in refrigerators and in wearable technologies. Perhaps one issue is the management of all information in the systems and how to assure interoperability – there should be work for testing there. Maybe even the official testing laboratories, which would test key systems maybe in the name of cybersecurity ;) The testing would be needed everywhere, but it is in the open who wants to invest in it and at what level.

Automation could be heavily increased.

It would be beneficial to considerably increase usage of aids/trainees (students on hour work). Experts and superiors could delegate parts of their work and concentrate on their special expertise.

APPENDIX 3: Collected change-competence snippets

ID	Change-competence snippet	Change caused by -> enables	Competence implications (re: quality and testing)	Links with
	Global environment			
1	Digitalisation	Advancement of technology -> opportunities, changes in products and systems, changes in cultures and societies	Business understanding #O #U Customer-centredness #O #U #A Understanding new products and systems #O #U Working under insecurity and change #O #A Critical thinking and presenting critique #A Experiment design skills #A Evaluation of product concepts #A Doing proof of concept tests for technology #A Doing critical technology assessments #A Prototyping skills #A UX and usability testing #A Understanding permission, security, privacy #O #U Data analysis #U #A Understanding modern word and its new practices and thinking #O Understanding complex systems #U Managing change with information #A Understanding overall contexts #U System and system of systems thinking and testing #U #A Information systems and integration competences #O #U #A Risk thinking #U Risk analysis skills #A Understanding information security risks #O #U Business and product concept level testing #A Architecture evaluation #A Understanding about ethics #O #U Doing ethical assessment #A	-> Pervasive communication -> Changing Finland -> Information security and privacy -> Experimentation culture -> From products to services -> The startup phenomenon -> Machine industry turning into software industry -> Networked communication -> Experimentation culture -> Business understanding for all -> Industrial Internet -> Big Data -> Innovation in product development -> Rethinking the goals of testing and quality assurance

ID	Change-competence snippet	Change caused by -> enables	Competence implications (re: quality and testing)	Links with
2	Responding to change	Changing world -> new ideas, practices	Understanding modern word and its new practices and thinking #O Understanding domains, contexts and situations #O #U Understanding changing nature of quality #O #U Understanding new products and systems #O #U Working under insecurity and change #O #A Understanding complex systems #U Managing change with information #A Collaboration skills #U #A Right timing of actions #A	-> Responding to change -> Living with contradictions -> Agility and flexibility -> Need for new types of workers -> The changing requirements of technical software systems -> Fast product development
3	Living with contradictions	Modern complex world view -> making good decisions	Handling contradictions #U Understanding overall contexts #U Broad flexible competence #O #U #A Multi-skilledness #O #U #A Working under insecurity and change #O Creativity #A Understanding changing nature of quality #O #U	<- Relation to change <- Changing Finland -> Agility and flexibility -> Flexibility over maturity
4	Pervasive communication	ICT technology -> social media, embedded communication	Using social media and web in getting information and sharing information #O #U #A Reputation management #A Communication skills #U #A Cultural skills (national, occupation, domains) #O #U #A Understanding permission, security, privacy #O #U	<- Information security and privacy
5	Information security and privacy	All information online, connected systems and devices	Risk thinking #U Understanding information security risks #O #U Risk analysis skills #A Product risk analysis #A Customer's risk analysis #A Security assessment and testing #A System and system of systems thinking and testing #U #A Understanding about ethics #O #U Doing ethical assessment #A	-> Pervasive communication -> Industrial Internet -> Cloud testing

ID	Change-competence snippet	Change caused by -> enables	Competence implications (re: quality and testing)	Links with
6	Emphasis on real competence	Companies rely on competences -> competences into use -> better business	Understanding about competence #U Broad flexible competence #O #U #A Multi-skilledness #O #U #A Collaboration skills #U #A Team skills #A Work and process design #A (to tempt competent people) Competence development focused on business needs #U #A	-> Responding to change -> Agility and flexibility -> Need for new types of workers -> Quest for multi-skilledness -> Finnish style challenged -> Changing engineering education
	Changing national working life			
7	Changing Finland	Political changes, economy changes -> New opportunities	National competence infrastructure development #A Improving education for quality and testing #U #A	<- Responding to change -> New external operating environment -> Smaller companies -> Need for new types of workers -> Changing engineering education -> Effective work in small, smart companies -> The startup phenomenon -> Finnish style challenged
8	Changing working life	Changing society, new generation of population -> new ways of working, using people’s competences fully, rich work	Broad flexible competence #O #U #A Multi-skilledness #O #U #A Collaboration skills #U #A Social skills #A Team skills #A Role finding #A Creativity #A Business understanding #O #U Active, self-steered working for quality #A Understanding innovation #U Cultural skills (national, occupation, domains) #O #U #A Communication skills #U #A	<- Smaller companies -> Relation to change -> Finnish style challenged -> Need for new types of workers -> Quest for multi-skilledness -> Changing engineering education -> Emphasis on real competence -> Better workplaces

ID	Change-competence snippet	Change caused by -> enables	Competence implications (re: quality and testing)	Links with
9	Need for new types of workers	Changing society, economic systems -> opportunities for broad competences	Creativity #A Understanding overall contexts #U Business understanding #O #U Active, self-steered working for quality #A Broad flexible competence #O #U #A Multi-skilledness #O #U #A Personal competence development #A Creativity #A Risk thinking #U Role finding #A Social skills #A Collaboration skills #U #A Communication skills #U #A	<- Changing working life -> Finnish style challenged -> Changing engineering education -> Quest for multi-skilledness -> Better workplaces -> Living with contradictions
10	Changing engineering education	Need for innovation and product development skills -> new businesses	National competence infrastructure development #A Understanding innovation #U Assessment and testing of innovations and product concepts #A Creativity #A Understanding overall contexts #U Business understanding #O #U Experiment design skills #A Prototyping skills #A	<- Changing Finland <- Changing working life <- Quest for multi-skilledness <- Innovation in product development <- New technology products <- Experimentation culture
11	Cultural competences emphasised	Raise of abstraction level, more communication, global networking -> shared competences into use	Cultural skills (national, occupation, domains) #O #U #A Cultural adaptation #A Communication skills #U #A	<- Changing Finland <- New external operating environment -> Finnish style challenged -> Need for new types of workers -> Networked communication
	Changes in the structure of the economy			
12	Smaller companies	Normalisation of society -> more dynamic economy	Broad flexible competence #O #U #A Multi-skilledness #O #U #A Adaptability and flexibility #U #A Independent problem solving capability #A Competences usable in various process models and contexts #A Personal competence development #A Forming and promoting practices #A Active, self-steered working for quality #A	-> Changing Finland -> Need for new types of workers -> Finnish style challenged -> Quest for multi-skilledness -> Emphasis on real competence -> The startup phenomenon

ID	Change-competence snippet	Change caused by -> enables	Competence implications (re: quality and testing)	Links with
13	New external operating environment	Global economy -> opportunities	Risk thinking #U Cultural skills (national, occupation, domains) #O #U #A Networking skills #A System and system of systems thinking and testing #U #A Platform-agnostic skills #A Comparison testing #A	-> Relation to change -> Cultural competences emphasised -> Modern risk management -> Emphasis on real competence -> Networked communication
14	From products to services	Managed products, raise of abstraction level, Industrial Internet and IoT -> more added value	Understanding overall contexts #U Business understanding #O #U Customer-centredness #O #U #A System and system of systems thinking and testing #U #A IoT-related competences #A Information systems and integration competences #O #U #A Risk thinking #U Risk, safety and reliability analysis #A Security assessment and testing #A Competence development focused on business needs #U #A	<- Digitalisation -> Platform economy and API economy -> Business understanding for all -> Modern risk management -> Machine industry turning into software industry
15	Platform economy and API economy	Managed products, raise of abstraction level, need for added value while limited internal resource -> more added value, growth	Understanding overall contexts #U Business understanding #O #U Customer-centredness #O #U #A System and system of systems thinking and testing #U #A Information systems and integration competences #O #U #A Architecture evaluation #U #A Risk thinking #U Understanding permission, security, privacy #O #U Security assessment and testing #A UX and usability testing #O #U #A Quality advocacy #O #U #A	<- Digitalisation <- From products to services <- Industrial Internet <- Big Data -> Small inexpensive apps -> Business understanding for all -> Modern risk management -> Machine industry turning into software industry

ID	Change-competence snippet	Change caused by -> enables	Competence implications (re: quality and testing)	Links with
16	The startup phenomenon	Changing society, need for innovation -> new companies, new products, new economy	Focusing and prioritising actions at each startup phase #U #A Working under insecurity and change #O #U #A Business understanding #O #U Multi-skilledness #A Personal competence development #A Process development #O #U #A (when changing into growth mode) Understanding of possibilities and alternatives in testing #U Team skills #A Creativity #A Right timing of actions #A Making compromises in quality #O #U #A Comparison testing #A UX testing for feature development #O #U #A	-> Effective work in small, smart companies -> Finnish style challenged -> Agility and flexibility -> Need for new types of workers -> Quest for multi-skilledness -> Business understanding for all -> New thinking on defect costs during application lifecycle -> Rethinking the goals of testing and quality assurance
17	The rise of the game industry	Global open mobile device software market -> new companies, room for innovation	Broad flexible competence #O #U #A Multi-skilledness #O #U #A Team skills #A Multi-skilledness #A Role finding #A Understanding users #U Understanding quality and its practices #U Open-minded quality thinking #O #U UX and usability testing #A Collaboration skills #U #A Understanding permission, security, privacy #O #U Security assessment and testing #A Forming and promoting practices #A Working under insecurity and change #O Business understanding #O #U Configuration testing #A Making compromises in quality #O #U #A	<- The startup phenomenon -> Finnish style challenged -> Need for new types of workers -> Rethinking the goals of testing and quality assurance -> Information security and privacy -> Cultural competences emphasised -> Small inexpensive apps -> New technology products -> Need for personal understanding of quality
	Changes in some businesses			

ID	Change-competence snippet	Change caused by -> enables	Competence implications (re: quality and testing)	Links with
18	Finnish style challenged	Stereotypical Finnish working style not sufficient in all conditions -> versatility in working	Reflection on working styles #U #A Handling contradictions #U Cultural skills (national, occupation, domains) #O #U #A Cultural adaptation #A Broad flexible competence #O #U #A Multi-skilledness #O #U #A Collaboration skills #U #A Social skills #A Team skills #A Role finding #A	<- Changing working life -> Cultural competences emphasised -> Need for new types of workers
19	Competences focused on business type	Different businesses need different competences, business innovation -> effectiveness, good business	Business understanding #O #U Understanding about competence #U Competences usable in various process models and contexts #A Broad flexible competence #O #U #A Multi-skilledness #O #U #A Cultural skills (national, occupation, domains) #O #U #A Competence development focused on business needs #U #A Understanding of possibilities and alternatives in testing #U Understanding overall contexts #U Risk thinking #U Personal competence development #O #U #A	-> Business understanding for all -> Quest for multi-skilledness -> Explosion of important quality attributes -> Cultural competences emphasised
20	Machine industry turning into software industry	Software controlled machines, production control, integrated information systems -> more added value into products	Changing company-level competence profile #O #U #A Generic software quality and testing competences #O #U #A Process development #U #A Information systems and integration competences #A Business understanding #O #U Understanding overall contexts #U System and system of systems thinking and testing #U #A Understanding information security risks #O #U Security assessment and testing #A Data analysis #U #A	<- From products to services <- Platform economy and API economy -> Business understanding for all -> Industrial Internet
	Working style in companies			

ID	Change-competence snippet	Change caused by -> enables	Competence implications (re: quality and testing)	Links with
21	Effective work in small, smart companies	Changing world -> speed, effectiveness, innovation	Process development #U #A Managing change with information #A Quality advocacy #A Active, self-steered working for quality #A Collaboration skills #U #A Team skills #A Networking skills #A Role finding #A	-> Need for new types of workers -> Finnish style challenged -> The startup phenomenon
22	Testers in development teams	New social systems in companies -> more integration, fast feedback	Social skills #A Team skills #A Role finding #A Quality advocacy #A	<- Changing working life
23	Networked communication	Networked, dynamic industry -> external social systems, information flow	Social skills #A Networking skills #A Communication skills #U #A Understanding information security risks #O #U Understanding domains, contexts and situations #O #U	<- Pervasive communication -> Information security and privacy
24	Experimentation culture	Need for innovation -> validated ideas, concepts	Evaluation of product concepts #A Critical thinking and presenting critique #A Prototyping skills #A Hardware-related skills #U #A Experiment design skills #A Doing proof of concept tests for technology #A Using exploratory testing for understanding the behaviour of technology #A UX and usability testing #A Understanding permission, security, privacy #O #U Data analysis #U #A Creativity #A Cultural adaptation #A Changing company-level competence profile #O #U #A	<- Innovation in product development <- Fast product development -> Need for personal understanding of quality -> Flexibility over maturity -> Changing engineering education

ID	Change-competence snippet	Change caused by -> enables	Competence implications (re: quality and testing)	Links with
25	Agility and flexibility	Dynamic environment -> business change to new domains (existing and emerging)	Understanding of domains and cultures #U Domain-agnostic competences #A Cultural skills (national, occupation, domains) #O #U #A Adaptability and flexibility #A Understanding of possibilities and alternatives in testing #U Broad flexible competence #O #U #A Multi-skilledness #O #U #A	<- Relation to change <- Changing Finland <- Flexibility over maturity <- Need for new types of workers <- Testers in development teams -> Agile software development -> Lean -> Need for personal understanding of quality
26	Faster decision making	Dynamic environment, fast business -> rapid reaction, fast action	Business understanding #O #U Communication skills #U #A Experiment design skills #A Prototyping skills #A Business and product concept level testing #A Quality advocacy #A Risk thinking #U Critical thinking and presenting critique #A Comparison testing #A Right timing of actions #A Dependability #O #A	<- Relation to change <- Flexibility over maturity <- Agile software development <- Lean -> Need for new types of workers -> Need for personal understanding of quality
27	Flexibility over maturity	Dynamic environment	Understanding about competence #U Adaptability and flexibility #U #A Broad flexible competence #O #U #A Multi-skilledness #O #U #A Understanding overall contexts #U	<- Agility and flexibility <- Emphasis on real competence
	Relations to competence in companies			
28	Quest for multi-skilledness	Effectiveness, smaller companies -> collaboration, dynamic organisation	Broad flexible competence #O #U #A Multi-skilledness #O #U #A Personal competence development #O #U #A	<- The startup phenomenon <- Need for new types of workers <- Business understanding for all <- New technology products <- Testing of intelligent systems -> Changing engineering education

ID	Change-competence snippet	Change caused by -> enables	Competence implications (re: quality and testing)	Links with
29	Business understanding for all	Testing needs to support business -> better testing, better information -> better business	Business understanding #O #U Understanding of domains and cultures #U Business understanding #U Understanding customers #U Understanding users #U	<- Quest for multi-skilledness <- Smaller companies <- Changing Finland <- Testing in every process <- Innovation in product development -> New thinking on defect costs during application lifecycle
30	Scaling of competences	Changing businesses, domains, growing companies -> successful lifespan for companies	Scaling personal toolbox #A Platform-agnostic skills #A Understanding domains, contexts and situations #U Scaling resource management practices #A Risk thinking #U Process development #O #U #A Business understanding #O #U	<- The startup phenomenon <- From products to services <- Machine industry turning into software industry <- The changing requirements of technical software systems
31	Testing in every process	Moving from engineering to product development -> better business, lower risk level	Business understanding #O #U Business and product concept level testing #A System and system of systems thinking and testing #U #A Process development #U #A (for integrating testing and experimentation into business activities)	-> Business understanding for all
32	Integrated QA	Varying contexts, team-independence, ISO 9001 experiences -> independent thinking, selecting most suitable practices for context	Understanding quality and its practices #U Understanding domains, contexts and situations #O #U Process development #U #A Quality advocacy #A Active, self-steered working for quality #A	<- Flexibility over maturity <- Agility and flexibility <- Need for personal understanding of quality <- Rethinking the goals of testing and quality assurance <- Testers in development teams
	Changes in product technology			

ID	Change-competence snippet	Change caused by -> enables	Competence implications (re: quality and testing)	Links with
33	Industrial Internet	Internet technologies, low cost of communication technology -> added value by intelligence, monitoring, maintenance	System and system of systems thinking and testing #U #A UX and usability testing #A Risk, safety and reliability analysis #A IoT-related competences #O #U #A Hardware-related skills #U #A Understanding information security risks #O #U Information systems and integration competences #O #U #A Security assessment and testing #A Data analysis #U #A Architecture evaluation #A Doing critical technology assessments #A	<- Big Data -> Platform economy and API economy -> Multi-device systems with new interaction styles -> Testing of intelligent systems -> New technology products -> Information security and privacy
34	Big Data	Connectivity, sensors -> monitoring, prediction, testing	Instrumenting of systems #A Efficient and secure data collection #A Information systems and integration competences #O #U #A Data analysis #U #A Using analysis and reporting tools #A Understanding information security risks #O #U Security assessment and testing #A Architecture evaluation #A	-> Industrial Internet -> Platform economy and API economy -> Information security and privacy
35	Cloud testing	Cloud -> dynamic test environments, low investment, new testing opportunities	Understanding cloud systems, their possibilities and problems #U Managing of test environments in the cloud #A Deployment and automation skills #A Understanding information security risks #O #U Learning new testing tools #A	-> Virtualisation -> From products to services -> Fast product development -> Information security and privacy
36	Virtualisation	Virtualisation technology, computer capabilities -> fast deployment of environment, hardware and OS-agnosticism	Understanding virtualisation #U Virtual environment design and implementation #A Virtual environment deployment skills #A	-> Cloud testing -> Fast product development

ID	Change-competence snippet	Change caused by -> enables	Competence implications (re: quality and testing)	Links with
37	Multi-device systems with new interaction styles	Device interactions, IoT -> collaborative device systems	System and system of systems thinking and testing #U UX and usability testing #A Testing of complex interactions #A IoT-related competences #O #U #A Experiment design skills #A Doing proof of concept tests for technology #A Security assessment and testing #A Risk, safety and reliability analysis #A Using exploratory testing for understanding the behaviour of technology #A Hardware-related skills #U #A Configuration testing #A Installation testing #A Architecture evaluation #A Configuration management #A	<- Industrial Internet -> Information security and privacy
	Changes in product requirements			
38	Explosion of important quality attributes	Quality attributes expand -> need to assess for total quality -> better products	Understanding changing nature of quality #O #U Understanding of product, product culture, businesses and their needs #U Open-minded quality thinking #O #U Customer-centredness #O #U #A Quality advocacy #A	<- Innovation in product development <- The changing requirements of technical software systems <- Need for personal understanding of quality -> Business understanding for all -> Competences focused on business type
39	The changing requirements of technical software systems	Complexity, risks, nature of systems -> better, focused testing	Understanding of product, product culture, businesses and their needs #U Understanding changing nature of quality #O #U Understanding systems' requirements #U Understanding technical systems #U Understanding complex systems #U Risk-based testing #A Robustness testing #A Open-minded quality thinking #O #U Cost-benefit thinking in selecting quality practices #O	-> Modern risk management -> Integrated QA

ID	Change-competence snippet	Change caused by -> enables	Competence implications (re: quality and testing)	Links with
40	Small inexpensive apps	App culture, app stores, heavy competitions, low cost -> business possibilities	Risk thinking #U Business understanding #O #U Making compromises in quality #O #U #A Cost-benefit thinking in selecting quality practices #U Business and product concept level testing #A	-> Business understanding for all -> Agility and flexibility -> Modern risk management
41	New technology products	New technology -> new concepts, disruptive products, new business	Understanding technology #U Evaluation of product concepts #A Understanding overall product lifecycles #U Critical thinking and presenting critique #A Doing critical technology assessments #A Hardware-related skills #U #A Understanding what qualities are the most critical at each phase of the company's lifecycle phase and the product development phase #U Risk, safety and reliability analysis #A Using exploratory testing for understanding the behaviour of technology #A Experiment design skills #A	<- Innovation in product development <- Fast product development <- The startup phenomenon -> Testing of intelligent systems -> Experimentation culture
42	Testing of intelligent systems	AI, robotics -> human-like robot, intelligent software systems	Broad flexible competence #O #U #A Understanding new products and systems #U Evaluation of product concepts #A Understanding complex systems #U Testing of complex interactions #A Security assessment and testing #A Hardware-related skills #U #A Safety management #U #A (on safety-critical domains) Risk, safety and reliability analysis #A Understanding innovation #U Open-minded quality thinking #O #U Understanding users #U UX and usability testing #A Team skills #U	-> Information security and privacy -> Need for new types of workers -> Modern risk management <- New technology products <- Multi-device systems with new interaction styles
	Software development process changes			

ID	Change-competence snippet	Change caused by -> enables	Competence implications (re: quality and testing)	Links with
43	Innovation in product development	Need for new products, disruption -> new business	Understanding innovation #U Understanding of product, product culture, businesses and their needs #U Understanding overall product lifecycles #U Understanding the product development paradigm #U Evaluation of product concepts #A Critical thinking and presenting critique #A Prototyping skills #A Doing proof of concept tests for technology #A Experiment design skills #A Understanding of needs of development #O #U Doing experiments with users #A UX testing for feature development #O #U #A Working under insecurity and change #O Team skills #A Understanding about the company's business **#**U Understanding customers #U Understanding about ethics #O #U Doing ethical assessment #A	<- Changing Finland -> Industrial Internet -> Changing engineering education -> Experimentation culture -> Fast product development -> Rethinking the goals of testing and quality assurance
44	Relation to change	Dynamic environment -> change offers opportunities	Adaptability and flexibility #A Right timing of actions #A Short work-in-progress lists #A Understanding software engineering #U Reflection on working styles #U #A Rigour in collaborative keeping the platform robust and tolerating change #A	<- Agility and flexibility -> Agile software development -> Lean
45	Timing and rhythm	Agility, speed, efficiency -> flow, efficiency	Sense of rhythm #U Right timing of actions #O #U #A	-> Designing new development lifecycles -> The next steps in software development lifecycles -> Agility and flexibility -> Agile software development -> Lean

ID	Change-competence snippet	Change caused by -> enables	Competence implications (re: quality and testing)	Links with
46	Towards continuous delivery	Reactivity, speed of deployment, deployment risk control -> capability used for various business benefits	Discipline #O #A Deployment and automation skills #A Configuration management #A Rigour in collaborative keeping the platform robust and tolerating change #A Process development #A (integrating manual testing into the workflow) Supporting deployment decisions with assessment and test information #A	-> Lean -> Fast product development <- Timing and rhythm
47	Fast product development	Rapid market entry, reactivity -> timing for actions, customer satisfaction	Understanding the product development paradigm #U Quality advocacy #A Risk thinking #U Customer-centredness #O #U #A Process development #A (solid testing and rigour in doing it) UX testing for feature development #O #U #A Comparison testing #A Configuration management #A Rigour in collaborative keeping the platform robust and tolerating change #A Dependability #O #A Independent problem solving capability #A Changing company-level competence profile #O #U #A	<- Relation to change -> Innovation in product development -> Towards continuous delivery -> Lean -> Business understanding for all -> Experimentation culture <- Timing and rhythm
48	Modern risk management	Incremental development -> better understanding of risk by learning	Business understanding #U Understanding the customer's business and needs #O #U Risk thinking #U Product risk analysis #A Customer’s risk analysis #A Safety management #U #A (on safety-critical domains) Understanding information security risks #O #U Security assessment and testing #A Cost-benefit thinking in selecting quality practices #O #U #A Dependability #O #A	<- Agility and flexibility <- Business understanding for all <- Explosion of important quality attributes <- Information security and privacy <- Pervasive communication
	Evolving lifecycle models			

ID	Change-competence snippet	Change caused by -> enables	Competence implications (re: quality and testing)	Links with
49	Designing new development lifecycles	Rethinking software development lifecycle models -> opportunity to design tailored methods for particular needs	Process development #U #A (integrating testing into any new method)	-> Working in various development lifecycles -> The next steps in software development lifecycles -> Agile software development -> Lean
50	Working in various development lifecycles	Different situations require different lifecycles, job market dynamism -> adaptability, effectiveness, opportunities	Understanding software engineering #U Understanding product/system development #O #U Understanding the product development paradigm #O #U Understanding of needs of development #U Competences usable in various process models and contexts #A Versatile method/practice toolbox #A Understanding the relevant lifecycles #U	-> Designing new development lifecycles -> The next steps in software development lifecycles -> Agile software development -> Lean
51	The next steps in software development lifecycles	Lifecycles always evolve -> new practices fulfil perceived needs	Understanding the product development paradigm #U Process development #U (needs for tailoring, addition, re-planning of processes) Competences usable in various process models and contexts #A Understanding overall product lifecycles #U	-> Designing new development lifecycles -> The next steps in software development lifecycles -> Agile software development -> Lean -> Fast product development
52	Agile software development	Deplorability, changeability, learning -> less risks	Understanding the product development paradigm #U Exploratory testing for feature development #A UX testing for feature development #A Right timing of actions #A Sense of rhythm #U Communication skills #U #A Team skills #A Active, self-steered working for quality #A Configuration management #A Short work-in-progress lists #O #U #A Rigour in collaborative keeping the platform robust and tolerating change #A Role finding #A	-> Designing new development lifecycles -> The next steps in software development lifecycles -> Lean <- Agility and flexibility <- Timing and rhythm

ID	Change-competence snippet	Change caused by -> enables	Competence implications (re: quality and testing)	Links with
53	Lean	Need for practices in agile development, flow control -> value flow, focus	Understanding the product development paradigm #U Process development #U #A (integrating testing into the flow) Short work-in-progress lists #O #U #A Helping developers in development queue #A Deployment and automation skills #A Configuration management #A	<- Agility and flexibility <- Agile software development -> Designing new development lifecycles -> The next steps in software development lifecycles
	Changes in testing thinking			
54	Rethinking the goals of testing and quality assurance	Better effectiveness and support for business -> better business	Test planning for purpose #U #A Business understanding #O #U Understanding product/system development #O #U Understanding of needs of development #U Understanding software engineering #U Versatile method/practice toolbox #A Collaboration skills #U #A Communication skills #U #A Right timing of actions #A	-> Business understanding for all -> Designing new development lifecycles
55	Need for personal understanding of quality	Less reliance of formal requirements -> opportunity to find essential characteristics	Open-minded quality thinking #O #U Understanding of product, product culture, businesses and their needs #U Understanding technical systems #U Using social media and web in getting information and sharing information #O #U #A Personal competence development #O #U #A	<- Explosion of important quality attributes <- Changing working life <- Agility and flexibility <- Business understanding for all <- New thinking on defect costs during application lifecycle
56	New thinking on defect costs during application lifecycle	Update / deployment mechanisms -> opportunities to rethink and prioritise processes	Understanding overall product lifecycles #U Understanding customers #U Business understanding #O #U Understanding software engineering #U Understanding deployment #U Deployment and automation skills #A	<- Fast product development <- Towards continuous delivery <- Small inexpensive apps -> Business understanding for all
	Testing arrangements			

ID	Change-competence snippet	Change caused by -> enables	Competence implications (re: quality and testing)	Links with
57	Testers changing context more often	Business and national dynamics -> competence transfer, new ideas	Understanding overall contexts #U Understanding domains, contexts and situations #O #U Business understanding #O #U Active, self-steered working for quality #A Broad flexible competence #O #U #A Domain-agnostic competences #A Versatile method/practice toolbox #A Multi-skilledness #O #U #A Role finding #A Reflection on working styles #U #A Cultural adaptation #A Social skills #A Collaboration skills #U #A Communication skills #U #A Using social media and web in getting information and sharing information #O #U #A	<- Changing working life <- Smaller companies <- The startup phenomenon -> Quest for multi-skilledness -> Changing engineering education -> Need for personal understanding of quality
58	Better workplaces	Modern worked need well designed work and workplaces -> well-being, effectiveness, quality	Work, process and practice design #A	<- Changing working life <- Need for new types of workers -> Gamification for engagement
59	Gamification for engagement	Characteristics of games improve work -> well-being, effectiveness, quality	Testing tool UX design #A Work, process and practice design #O #U #A Understanding about ethics #O #U Doing ethical assessment #A	<- Better workplaces <- Changing working life <- Need for new types of workers
60	Subcontractor competences	More business-critical collaboration -> competences promote effectiveness, efficiency, mutual development	Service design #A Customer-centredness #O #U #A Understanding the customer's business and needs #U Entrepreneurial competencies #A Reputation management #A Organisational competence development #U #A Competence development focused on business needs #U #A Competences usable in various process models and contexts #A Understanding of possibilities and alternatives in testing #U Platform-agnostic skills #A Collaboration skills #U #A Communication skills #U #A Dependability #A	-> Business understanding for all -> Fast product development -> Networked communication -> Testing service competences

ID	Change-competence snippet	Change caused by -> enables	Competence implications (re: quality and testing)	Links with
61	Testing service competences	Outsourcing essential -> reshoring more probable when service offerings develop	Understanding the customer's business and needs #O #U #A Service skills at all levels of the organisation #A Platform-agnostic skills #A Entrepreneurial competencies #A Understanding overall product lifecycles #U Customer-centredness #O #U #A Independent problem solving capability #A Special testing services #A (security, performance, UX) Versatile method/practice toolbox #A Understanding of possibilities and alternatives in testing #U Competences usable in various process models and contexts #A Competence development focused on business needs #U #A Discipline #O #A	<- Subcontractor competences <- Explosion of important quality attributes -> Business understanding for all -> Fast product development -> Networked communication
62	Crowd testing	Opportunities for employment, micro-services -> flexibility for some cases	Entrepreneurial competencies #A Reputation management #A Personal competence development #A Understanding about ethics #O #U Doing ethical assessment #A	<- Subcontractor competences <- Testing service competences -> Fast product development -> Networked communication

APPENDIX 4: Competences referenced in change-competence snippets

Competence	Number of references	In change-competence snippets
Business understanding	20	1 Digitalisation, 8 Changing working life, 9 Need for new types of workers, 10 Changing engineering education, 14 From products to services, 15 Platform economy and API economy, 16 The startup phenomenon, 17 The rise of the game industry, 19 Competences focused on business type, 20 Machine industry turning into software industry, 26 Faster decision making, 29 Business understanding for all, 30 Scaling of competences, 31 Testing in every process, 40 Small inexpensive apps, 48 Modern risk management, 54 Rethinking the goals of testing and quality assurance, 56 New thinking on defect costs during application lifecycle, 57 Testers changing context more often
Multi-skilledness	14	3 Living with contradictions, 6 Emphasis on real competence, 8 Changing working life, 9 Need for new types of workers, 12 Smaller companies, 16 The startup phenomenon, 17 The rise of the game industry, 18 Finnish style challenged, 19 Competences focused on business type, 25 Agility and flexibility, 27 Flexibility over maturity, 28 Quest for multi-skilledness, 57 Testers changing context more often
Broad flexible competence	13	3 Living with contradictions, 6 Emphasis on real competence, 8 Changing working life, 9 Need for new types of workers, 12 Smaller companies, 17 The rise of the game industry, 18 Finnish style challenged, 19 Competences focused on business type, 25 Agility and flexibility, 27 Flexibility over maturity, 28 Quest for multi-skilledness, 42 Testing of intelligent systems, 57 Testers changing context more often
Risk thinking	12	1 Digitalisation, 5 Information security and privacy, 9 Need for new types of workers, 13 New external operating environment, 14 From products to services, 15 Platform economy and API economy, 19 Competences focused on business type, 26 Faster decision making, 30 Scaling of competences, 40 Small inexpensive apps, 47 Fast product development, 48 Modern risk management
Process development	11	16 The startup phenomenon, 20 Machine industry turning into software industry, 21 Effective work in small, smart companies, 30 Scaling of competences, 31 Testing in every process, 32 Integrated QA, 46 Towards continuous delivery, 47 Fast product development, 49 Designing new development lifecycles, 51 The next steps in software development lifecycles, 53 Lean
Understanding overall contexts	10	1 Digitalisation, 3 Living with contradictions, 9 Need for new types of workers, 10 Changing engineering education, 14 From products to services, 15 Platform economy and API economy, 19 Competences focused on business type, 20 Machine industry turning into software industry, 27 Flexibility over maturity, 57 Testers changing context more often
Collaboration skills	10	2 Responding to change, 6 Emphasis on real competence, 8 Changing working life, 9 Need for new types of workers, 17 The rise of the game industry, 18 Finnish style challenged, 21 Effective work in small, smart companies, 54 Rethinking the goals of testing and quality assurance, 57 Testers changing context more often, 60 Subcontractor competences

Competence	Number of references	In change-competence snippets
Communication skills	10	4 Pervasive communication, 8 Changing working life, 9 Need for new types of workers, 11 Cultural competences emphasised, 23 Networked communication, 26 Faster decision making, 52 Agile software development, 54 Rethinking the goals of testing and quality assurance, 57 Testers changing context more often, 60 Subcontractor competences
Security assessment and testing	10	5 Information security and privacy, 14 From products to services, 15 Platform economy and API economy, 17 The rise of the game industry, 20 Machine industry turning into software industry, 33 Industrial Internet, 34 Big Data, 37 Multi-device systems with new interaction styles, 42 Testing of intelligent systems, 48 Modern risk management
Team skills	10	6 Emphasis on real competence, 8 Changing working life, 16 The startup phenomenon, 17 The rise of the game industry, 18 Finnish style challenged, 21 Effective work in small, smart companies, 22 Testers in development teams, 42 Testing of intelligent systems, 43 Innovation in product development, 52 Agile software development
System and system of systems thinking and testing	9	1 Digitalisation, 5 Information security and privacy, 13 New external operating environment, 14 From products to services, 15 Platform economy and API economy, 20 Machine industry turning into software industry, 31 Testing in every process, 33 Industrial Internet, 37 Multi-device systems with new interaction styles
Understanding information security risks	8	1 Digitalisation, 5 Information security and privacy, 20 Machine industry turning into software industry, 23 Networked communication, 33 Industrial Internet, 34 Big Data, 35 Cloud testing, 48 Modern risk management
Role finding	8	8 Changing working life, 9 Need for new types of workers, 17 The rise of the game industry, 18 Finnish style challenged, 21 Effective work in small, smart companies, 22 Testers in development teams, 52 Agile software development, 57 Testers changing context more often
Customer-centredness	7	1 Digitalisation, 14 From products to services, 15 Platform economy and API economy, 38 Explosion of important quality attributes, 47 Fast product development, 60 Subcontractor competences, 61 Testing service competences
Experiment design skills	7	1 Digitalisation, 10 Changing engineering education, 24 Experimentation culture, 26 Faster decision making, 37 Multi-device systems with new interaction styles, 41 New technology products, 43 Innovation in product development
UX and usability testing	7	1 Digitalisation, 15 Platform economy and API economy, 17 The rise of the game industry, 24 Experimentation culture, 33 Industrial Internet, 37 Multi-device systems with new interaction styles, 42 Testing of intelligent systems
Right timing of actions	7	2 Responding to change, 16 The startup phenomenon, 26 Faster decision making, 44 Relation to change, 45 Timing and rhythm, 52 Agile software development, 54 Rethinking the goals of testing and quality assurance
Creativity	7	3 Living with contradictions, 8 Changing working life, 9 Need for new types of workers, 10 Changing engineering education, 16 The startup phenomenon, 24 Experimentation culture
Cultural skills	7	4 Pervasive communication, 8 Changing working life, 11 Cultural competences emphasised, 13 New external operating environment, 18 Finnish style challenged, 19 Competences focused on business type, 25 Agility and flexibility

Competence	Number of references	In change-competence snippets
Active, self-steered working for quality	7	8 Changing working life, 9 Need for new types of workers, 12 Smaller companies, 21 Effective work in small, smart companies, 32 Integrated QA, 52 Agile software development, 57 Testers changing context more often
Personal competence development	7	9 Need for new types of workers, 12 Smaller companies, 16 The startup phenomenon, 19 Competences focused on business type, 28 Quest for multi-skilledness, 55 Need for personal understanding of quality, 62 Crowd testing
Quality advocacy	7	15 Platform economy and API economy, 21 Effective work in small, smart companies, 22 Testers in development teams, 26 Faster decision making, 32 Integrated QA, 38 Explosion of important quality attributes, 47 Fast product development
Working under insecurity and change	6	1 Digitalisation, 2 Responding to change, 3 Living with contradictions, 16 The startup phenomenon, 17 The rise of the game industry, 43 Innovation in product development
Information systems and integration competences	6	1 Digitalisation, 14 From products to services, 15 Platform economy and API economy, 20 Machine industry turning into software industry, 33 Industrial Internet, 34 Big Data
Social skills	6	8 Changing working life, 9 Need for new types of workers, 18 Finnish style challenged, 22 Testers in development teams, 23 Networked communication, 57 Testers changing context more often
Competences usable in various process models and contexts	6	12 Smaller companies, 19 Competences focused on business type, 50 Working in various development lifecycles, 51 The next steps in software development lifecycles, 60 Subcontractor competences, 61 Testing service competences
Understanding the product development paradigm	6	43 Innovation in product development, 47 Fast product development, 50 Working in various development lifecycles, 51 The next steps in software development lifecycles, 52 Agile software development, 53 Lean
Critical thinking and presenting critique	5	1 Digitalisation, 24 Experimentation culture, 26 Faster decision making, 41 New technology products, 43 Innovation in product development
Evaluation of product concepts	5	1 Digitalisation, 24 Experimentation culture, 41 New technology products, 42 Testing of intelligent systems, 43 Innovation in product development
Prototyping skills	5	1 Digitalisation, 10 Changing engineering education, 24 Experimentation culture, 26 Faster decision making, 43 Innovation in product development
Understanding permission, security, privacy	5	1 Digitalisation, 4 Pervasive communication, 15 Platform economy and API economy, 17 The rise of the game industry, 24 Experimentation culture
Data analysis	5	1 Digitalisation, 20 Machine industry turning into software industry, 24 Experimentation culture, 33 Industrial Internet, 34 Big Data
Architecture evaluation	5	1 Digitalisation, 15 Platform economy and API economy, 33 Industrial Internet, 34 Big Data, 37 Multi-device systems with new interaction styles
Understanding about ethics	5	1 Digitalisation, 5 Information security and privacy, 43 Innovation in product development, 59 Gamification for engagement, 62 Crowd testing
Doing ethical assessment	5	1 Digitalisation, 5 Information security and privacy, 43 Innovation in product development, 59 Gamification for engagement, 62 Crowd testing
Understanding domains, contexts and situations	5	2 Responding to change, 23 Networked communication, 30 Scaling of competences, 32 Integrated QA, 57 Testers changing context more often
Competence development focused on business needs	5	6 Emphasis on real competence, 14 From products to services, 19 Competences focused on business type, 60 Subcontractor competences, 61 Testing service competences

Competence	Number of references	In change-competence snippets
Risk, safety and reliability analysis	5	14 From products to services, 33 Industrial Internet, 37 Multi-device systems with new interaction styles, 41 New technology products, 42 Testing of intelligent systems
Understanding of possibilities and alternatives in testing	5	16 The startup phenomenon, 19 Competences focused on business type, 25 Agility and flexibility, 60 Subcontractor competences, 61 Testing service competences
Open-minded quality thinking	5	17 The rise of the game industry, 38 Explosion of important quality attributes, 39 The changing requirements of technical software systems, 42 Testing of intelligent systems, 55 Need for personal understanding of quality
Hardware-related skills	5	24 Experimentation culture, 33 Industrial Internet, 37 Multi-device systems with new interaction styles, 41 New technology products, 42 Testing of intelligent systems
Configuration management	5	37 Multi-device systems with new interaction styles, 46 Towards continuous delivery, 47 Fast product development, 52 Agile software development, 53 Lean
Understanding overall product lifecycles	5	41 New technology products, 43 Innovation in product development, 51 The next steps in software development lifecycles, 56 New thinking on defect costs during application lifecycle, 61 Testing service competences
Doing proof of concept tests for technology	4	1 Digitalisation, 24 Experimentation culture, 37 Multi-device systems with new interaction styles, 43 Innovation in product development
Understanding complex systems	4	1 Digitalisation, 2 Responding to change, 39 The changing requirements of technical software systems, 42 Testing of intelligent systems
Business and product concept level testing	4	1 Digitalisation, 26 Faster decision making, 31 Testing in every process, 40 Small inexpensive apps
Understanding changing nature of quality	4	2 Responding to change, 3 Living with contradictions, 38 Explosion of important quality attributes, 39 The changing requirements of technical software systems
Understanding innovation	4	8 Changing working life, 10 Changing engineering education, 42 Testing of intelligent systems, 43 Innovation in product development
Cultural adaptation	4	11 Cultural competences emphasised, 18 Finnish style challenged, 24 Experimentation culture, 57 Testers changing context more often
Adaptability and flexibility	4	12 Smaller companies, 25 Agility and flexibility, 27 Flexibility over maturity, 44 Relation to change
Platform-agnostic skills	4	13 New external operating environment, 30 Scaling of competences, 60 Subcontractor competences, 61 Testing service competences
Comparison testing	4	13 New external operating environment, 16 The startup phenomenon, 26 Faster decision making, 47 Fast product development
UX testing for feature development	4	16 The startup phenomenon, 43 Innovation in product development, 47 Fast product development, 52 Agile software development
Dependability	4	26 Faster decision making, 47 Fast product development, 48 Modern risk management, 60 Subcontractor competences
Deployment and automation skills	4	35 Cloud testing, 46 Towards continuous delivery, 53 Lean, 56 New thinking on defect costs during application lifecycle
Understanding of product, product culture, businesses and their needs	4	38 Explosion of important quality attributes, 39 The changing requirements of technical software systems, 43 Innovation in product development, 55 Need for personal understanding of quality

Competence	Number of references	In change-competence snippets
Understanding software engineering	4	44 Relation to change, 50 Working in various development lifecycles, 54 Rethinking the goals of testing and quality assurance, 56 New thinking on defect costs during application lifecycle
Rigour in collaborative keeping the platform robust and tolerating change	4	44 Relation to change, 46 Towards continuous delivery, 47 Fast product development, 52 Agile software development
Versatile method/practice toolbox	4	50 Working in various development lifecycles, 54 Rethinking the goals of testing and quality assurance, 57 Testers changing context more often, 61 Testing service competences
Understanding new products and systems	3	1 Digitalisation, 2 Responding to change, 42 Testing of intelligent systems
Doing critical technology assessments	3	1 Digitalisation, 33 Industrial Internet, 41 New technology products
Managing change with information	3	1 Digitalisation, 2 Responding to change, 21 Effective work in small, smart companies
Using social media and web in getting information and sharing information	3	4 Pervasive communication, 55 Need for personal understanding of quality, 57 Testers changing context more often
Reputation management	3	4 Pervasive communication, 60 Subcontractor competences, 62 Crowd testing
Understanding about competence	3	6 Emphasis on real competence, 19 Competences focused on business type, 27 Flexibility over maturity
Independent problem solving capability	3	12 Smaller companies, 47 Fast product development, 61 Testing service competences
Networking skills	3	13 New external operating environment, 21 Effective work in small, smart companies, 23 Networked communication
IoT-related competences	3	14 From products to services, 33 Industrial Internet, 37 Multi-device systems with new interaction styles
Making compromises in quality	3	16 The startup phenomenon, 17 The rise of the game industry, 40 Small inexpensive apps
Understanding users	3	17 The rise of the game industry, 29 Business understanding for all, 42 Testing of intelligent systems
Reflection on working styles	3	18 Finnish style challenged, 44 Relation to change, 57 Testers changing context more often
Changing company-level competence profile	3	20 Machine industry turning into software industry, 24 Experimentation culture, 47 Fast product development
Using exploratory testing for understanding the behaviour of technology	3	24 Experimentation culture, 37 Multi-device systems with new interaction styles, 41 New technology products
Understanding customers	3	29 Business understanding for all, 43 Innovation in product development, 56 New thinking on defect costs during application lifecycle
Cost-benefit thinking in selecting quality practices	3	39 The changing requirements of technical software systems, 40 Small inexpensive apps, 48 Modern risk management
Understanding of needs of development	3	43 Innovation in product development, 50 Working in various development lifecycles, 54 Rethinking the goals of testing and quality assurance
Short work-in-progress lists	3	44 Relation to change, 52 Agile software development, 53 Lean

Competence	Number of references	In change-competence snippets
Understanding the customer's business and needs	3	48 Modern risk management, 60 Subcontractor competences, 61 Testing service competences
Entrepreneurial competencies	3	60 Subcontractor competences, 61 Testing service competences, 62 Crowd testing
Understanding modern word and its new practices and thinking	2	1 Digitalisation, 2 Responding to change
Risk analysis skills	2	1 Digitalisation, 5 Information security and privacy
Handling contradictions	2	3 Living with contradictions, 18 Finnish style challenged
Product risk analysis	2	5 Information security and privacy, 48 Modern risk management
Customer's risk analysis	2	5 Information security and privacy, 48 Modern risk management
National competence infrastructure development	2	7 Changing Finland, 10 Changing engineering education
Forming and promoting practices	2	12 Smaller companies, 17 The rise of the game industry
Understanding quality and its practices	2	17 The rise of the game industry, 32 Integrated QA
Configuration testing	2	17 The rise of the game industry, 37 Multi-device systems with new interaction styles
Understanding of domains and cultures	2	25 Agility and flexibility, 29 Business understanding for all
Domain-agnostic competences	2	25 Agility and flexibility, 57 Testers changing context more often
Testing of complex interactions	2	37 Multi-device systems with new interaction styles, 42 Testing of intelligent systems
Understanding technical systems	2	39 The changing requirements of technical software systems, 55 Need for personal understanding of quality
Safety management	2	42 Testing of intelligent systems, 48 Modern risk management
Sense of rhythm	2	45 Timing and rhythm, 52 Agile software development
Discipline	2	46 Towards continuous delivery, 61 Testing service competences
Understanding product/system development	2	50 Working in various development lifecycles, 54 Rethinking the goals of testing and quality assurance
Work, process and practice design	2	58 Better workplaces, 59 Gamification for engagement
Work and process design	1	6 Emphasis on real competence
Improving education for quality and testing	1	7 Changing Finland
Assessment and testing of innovations and product concepts	1	10 Changing engineering education
Focusing and prioritising actions at each startup phase	1	16 The startup phenomenon
Generic software quality and testing competences	1	20 Machine industry turning into software industry
Scaling personal toolbox	1	30 Scaling of competences

Competence	Number of references	In change-competence snippets
Scaling resource management practices	1	30 Scaling of competences
Instrumenting of systems	1	34 Big Data
Efficient and secure data collection	1	34 Big Data
Using analysis and reporting tools	1	34 Big Data
Understanding cloud systems, their possibilities and problems	1	35 Cloud testing
Managing of test environments in the cloud	1	35 Cloud testing
Learning new testing tools	1	35 Cloud testing
Understanding virtualisation	1	36 Virtualisation
Virtual environment design and implementation	1	36 Virtualisation
Virtual environment deployment skills	1	36 Virtualisation
Installation testing	1	37 Multi-device systems with new interaction styles
Understanding systems' requirements	1	39 The changing requirements of technical software systems
Risk-based testing	1	39 The changing requirements of technical software systems
Robustness testing	1	39 The changing requirements of technical software systems
Understanding technology	1	41 New technology products
Understanding what qualities are the most critical at each phase of the company's lifecycle phase and the product development phase	1	41 New technology products
Doing experiments with users	1	43 Innovation in product development
Understanding about the company's business	1	43 Innovation in product development
Supporting deployment decisions with assessment and test information	1	46 Towards continuous delivery
Understanding the relevant lifecycles	1	50 Working in various development lifecycles
Exploratory testing for feature development	1	52 Agile software development
Helping developers in development queue	1	53 Lean
Test planning for purpose	1	54 Rethinking the goals of testing and quality assurance
Understanding deployment	1	56 New thinking on defect costs during application lifecycle
Testing tool UX design	1	59 Gamification for engagement
Service design	1	60 Subcontractor competences

Competence	Number of references	In change-competence snippets
Organisational competence development	1	60 Subcontractor competences
Service skills at all levels of the organisation	1	61 Testing service competences
Special testing services	1	61 Testing service competences

APPENDIX 5: Changes ranked by their effect on product development performance factors

ID is the number of the corresponding change-competence snippet.
Values for ranking: 3 high direct effect, 2 medium direct effect, 1 low effect or high possible indirect effect.
The table is mostly illustrative in nature and the rankings have low reliability.
Attribute All has weight of 3, others: 1.

ID	Change-competence snippet	All (infra-structure, culture) Weight = 3	Innova-tiveness	Effective ness	Agility	Speed	Quality and risk mana-gement	Weight-ed sum
15	Platform economy and API economy	3	3	3	1	2	3	21
17	The rise of the game industry	2	3		3	3	3	18
26	Faster decision making	2		3	3	3	3	18
1	Digitalisation	3	3				3	15
21	Effective work in small, smart companies		3	3	3	3	3	15
57	Testers changing context more often	2		2	2	2	3	15
18	Finnish style challenged		3	3	3	3		12
22	Testers in development teams			3	3	3	3	12
29	Business understanding for all		3	3	3		3	12
24	Experimentation culture		3		3	3	2	11
31	Testing in every process		3	3	1	1	3	11
35	Cloud testing			3	2	3	3	11
54	Rethinking the goals of testing and quality assurance		2	2	2	2	3	11
16	The startup phenomenon		3	1	3	3		10
27	Flexibility over maturity		2	1	3	1	3	10
36	Virtualisation		2	3	2	3		10
52	Agile software development			3	3	1	3	10
4	Pervasive communication	3						9
6	Emphasis on real competence	3						9
7	Changing Finland	3						9
8	Changing working life	3						9
9	Need for new types of workers	3						9
11	Cultural competences emphasised	3						9
12	Smaller companies	3						9
13	New external operating environment	3						9
14	From products to services		3		3	3		9
19	Competences focused on business type		3	3	3			9
20	Machine industry turning into software industry		3		3	3		9
23	Networked communication	3						9
25	Agility and flexibility			3	3	1	2	9
28	Quest for multi-skilledness	3						9
30	Scaling of competences	3						9
32	Integrated QA			3		3	3	9
39	The changing requirements of technical software systems			3	2	1	3	9
40	Small inexpensive apps		2	3	2	2		9
44	Relation to change			3	3	2	1	9
45	Timing and rhythm			3	3	2	1	9
49	Designing new development lifecycles	3						9
50	Working in various development lifecycles	3						9
51	The next steps in software development lifecycles	3						9
53	Lean			3	2	3	1	9
55	Need for personal understanding of quality		3	1	1	1	3	9
56	New thinking on defect costs during application lifecycle			2	2	2	3	9

58	Better workplaces	3						9
60	Subcontractor competences		1	3	1	1	3	9
47	Fast product development			2	2	3	1	8
48	Modern risk management			2	2	1	3	8
2	Responding to change		3		2		2	7
61	Testing service competences		2			2	3	7
10	Changing engineering education	2						6
42	Testing of intelligent systems		3				3	6
59	Gamification for engagement			3			3	6
38	Explosion of important quality attributes		2				3	5
3	Living with contradictions		3					3
5	Information security and privacy						3	3
33	Industrial Internet		3					3
34	Big Data		3					3
37	Multi-device systems with new interaction styles		3					3
41	New technology products		3					3
43	Innovation in product development		3					3
46	Towards continuous delivery					3		3
62	Crowd testing						3	3