Quality assurance (QA) in a software project

Matti Vuori

17.11.2015

Contents 1/3

Contents 2/3

Contents 3/3

Introduction 1/2

- This slide set is about quality assurance: The practices used for assuring the quality or the product.
- Main idea: When you do something, have a system for checking that it is good enough. In "QA" that system should usually be someone else or at least "external" to the core production process.
- The practices for QA are project practices and software development practices.
- QA is a subset of quality management, which looks at the whole operation around quality.

Introduction 2/2

- In the Agile culture, some people hate QA and think it is an enemy of innovation and agile development. That is a misconception.
- The truth is that QA is a safety net and enabler of innovation.

Product quality? 1/2

- Quality is about producing value to someone.
  - Customer, manufacturer.
- It is about doing the right product.
  - The product that is needed, instead of something that just came up.
- …And making it properly.
  - So it really does what it is supposed to do.
- …In a way that matches the contract, requirements and designs.
  - Code does what design says.
- …and the way is ok for the society
  - Meeting safety standards.

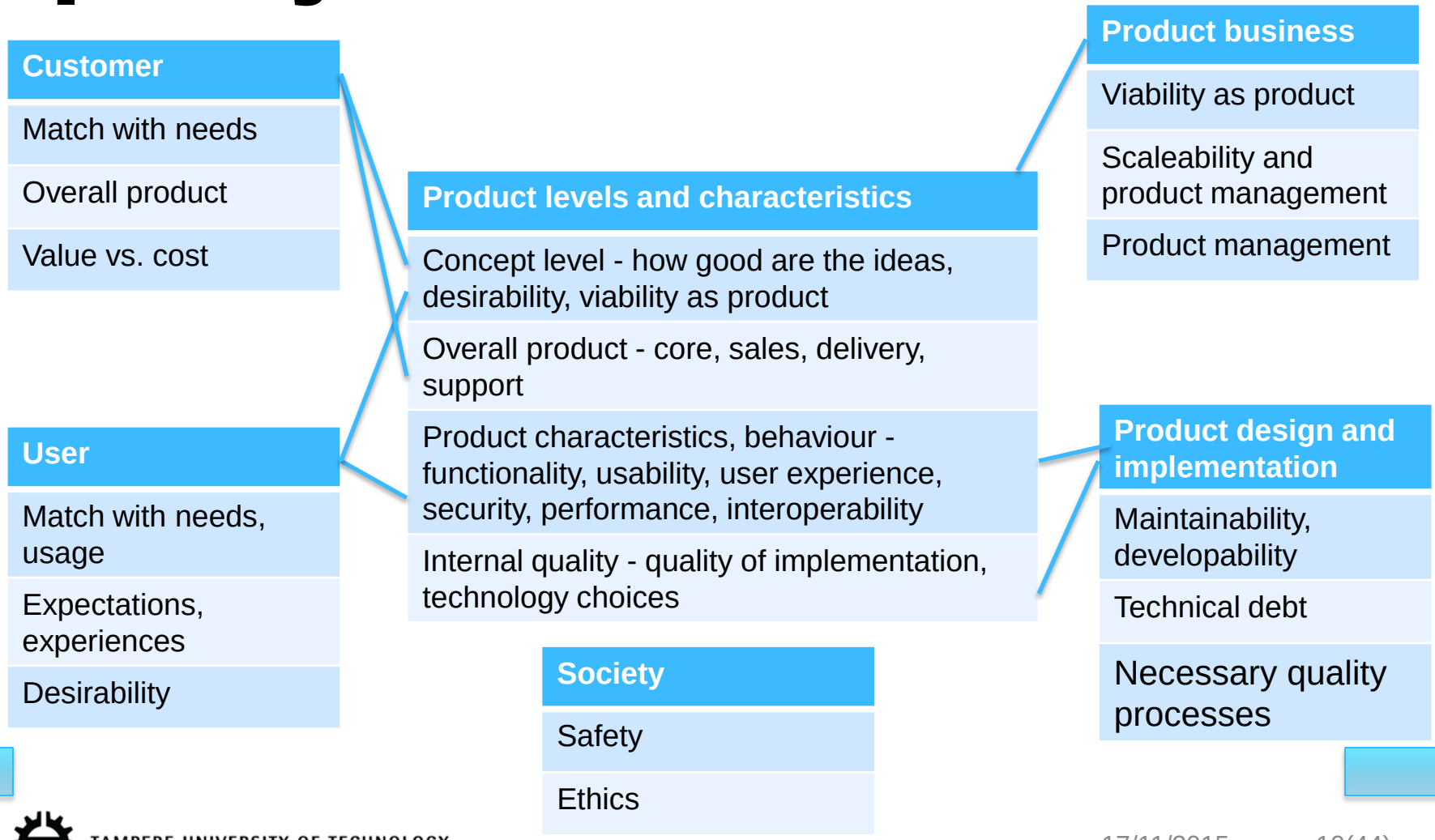
Product quality? 2/2

- Quality is always relative.
  - How good are the competitors?
  - What are the customers' and users' expectations?
  - The domain's culture?

Who defines product quality?

- Multiple viewpoints:
  - Customer and user define what is good and where compromises can be made.
  - Manufacturer has great interest in desirability, brand support of product and technical quality – developability, maintainability, and product risks.
  - Government agencies or accredited assessors define what is needed regarding safety or market entry on a regulated domain.
- But the development team should not define quality, just do it! (But it should develop the requirements.)
- Customer & user-centredness is usually key to good business.

The big picture of product quality



Quality factors vary at different layers of product – examples

Layer	Customer and user viewpoint	Manufacturer viewpoint
Product concept	Match to needs, values and desires	Clarity Technical feasibility Business viability – market, competition 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 to competition Developability Meeting of standards Lifespan expectancy
Technical product (software system)	Reliability Compatibility	Developability Maintainability

S/W quality model standard – ISO 25010 – for checklist 1/3

Charasteristic	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

S/W quality model standard – ISO 25010 – for checklist 2/3

Charasteristic	Sub-characteristics
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
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

S/W quality model standard – ISO 25010 – for checklist 3/3

Charasteristic	Sub-characteristics
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

Internal and external quality

- External quality is how the product "look" to the users: bugs, performance, usability…
- Internal quality is about how the product has been built: maintainability of code etc.
- On the long run, internal quality matters a lot. Neglecting it produces technical debt and rotting of the product.
- But on the short term, one must make the external quality good – "at any cost"…

Quality assurance 1/2

- Quality really can't be "assured". But we need to do our best that we
  - Understand what is required from the product and project.
  - Validate that the understanding is correct.
  - Make good designs and verify that they are based on the requirements and that we design the right thing.
  - Do good implementation and check that it is good on all aspects.
  - On delivery time, check that everything is in order.
  - And make any maintenance properly.
- Product quality results from quality of our thinking and actions (including processes)!

Quality assurance 2/2

- QA is:
  - Reflection of team's & own work. Are we doing things well?
  - Ensuring strictness of practices – when we have found good practices, they should be followed, because otherwise there will be problems.
  - Producing information for decision making.
  - Producing trust and proof for customers and official parties that we have done things like they should be done according to agreements and standards.
  - Risk management.
  - Just normal professional work.

Quality must be made

- Quality is made to the product by good design and implementation.
- Testing and quality assurance only checks how things are, but assuring makes no sense if you don't try to put quality in place.
- Same people that like to make quality also like to "assure it". Both are characteristics of mature professionals.
- This all depends on quality culture of the company.

QA priorities during the project lifecycle

Sprint 0	Sprints 1-N	Last, finalising sprint
Good concept Technology choices Project plan Review of starting point Identifying requirements	Meeting requirements prioritised Good design Solid, maintainable code Technical debt Testing	Overall product Overall quality Meeting all requirements Maturity – bugs Deployment practices Maintainability

About practices

- QA is built on things that we do – actions, practices.
- Practices should be chosen by business value: how much we can assure and improve business by doing some reviews or testing?
- Practices are influenced by culture too – domains have habits and that improves collaboration.
- Practices are sometimes required by a voluntary standard (official or de facto) or a mandatory standard, such as the safety standard IEC 61508 (see Testing of safety critical software – some principles https://noppa.oulu.fi/noppa/kurssi/811601s/luennot/811601S_lecture_11__vuori.pdf)

Common practices

- Reviews and inspections: Sitting down together and checking whether a plan, specification or implementation is correct.
- Risk analysis: To produce information about what might happen.
- Prototypes: Used for validating that ideas make sense.
- Testing during development: Mostly verifying that implementation is done correctly.
- Usability and UX testing: Validating the user interface. Can validate the product, or verify matching plans.
- Design analysis: Expert analysis for UI, architecture and so on.
- Safety and reliability analysis. Mostly validating the product. Reliability analysis also produces info for design and testing.
- Acceptance testing by customer: Validating the product from customer's perspective.

Common QA problems

- Focus only on functional quality at low level (code-centricity). Too much relying on test automation.
- No systematic assessment of usability and security.
- No metrics. At least trend of open bugs should be monitored.
- Insufficient testing at all levels. Regression problems.
- Definition of done does not consider testing at system level, integrated into the whole product.
- Lack of discipline in doing things well.
- Lack of roles (no testers in team) and competence.

QA is mostly integrated with development

- Most QA activities are done in the development team.
- Sometimes there are external activities:
  - Independent QA testing in safety critical domains.
  - QA done in a proper productising phase, after a team releases a version to be turned into a polished product.
  - Distribution channels may have their own testing processes.
- But the overall trend is toward doing QA mostly in the teams, and there it is not called QA, but testing, reviews and other practical things.

Characteristics of good QA

- Proactive:
  - Detect deficiencies NOW that could hit us later.
- Reactive and agile:
  - React to small problems and change plans according to the emerging needs.
- Based on quality culture:
  - Respect for culture and discipline.
- Professional:
  - There are good people who understand the quality practices.

Roles and responsibilities 1/2

- Outside team:
  - Company management just provides the opportunity, resources and tools to make quality.
  - Company has the legal responsibility.
  - Project planners need to include the necessary tasks into project.
  - Quality managers maintain the quality management system and monitor things company-wide.

Roles and responsibilities 2/2

- Inside team:
  - Project manager must see that the project manages quality.
  - Every developer has a responsibility to produce quality
  - Team needs someone with tester/quality role to do and look after these things.
  - Team must get outside help for expert tasks (such as load testing, usability testing).

Quality of requirements

- Understanding the requirements are a key.
- Project team must discuss requirements with customer to see that both understand what is wanted.
- Need to use demos – discussions is prone to errors.
- But that starts the development and creating real requirements.
- We need to check that standards, laws, environment and technical constraints are considered.
- Practices:
  - Reviews are a tool for quality assurance.
  - The customer must accept the developed requirements at the beginning and when they evolve (like new use cases in sprints). This is done besides quality, also for risk management: when there are problems, there needs to be documented understanding of things.

Quality of project plans

- For good quality, the development team should plan their project, not an external consultant who just hands over the plan.
- Practices for QA:
  - Review of plan inside team
  - Review of plan with customer.
- Essential:
  - Checking that everything is realistic, everything can and will happen.
  - Simulating the project flow.
  - A checklist helps a lot.

Reviews are beneficial

- Forming a shared understanding, learning.
- Provide rhythm to project.
  - Ending and starting of activities.
  - Looking back and ahead.
- Opportunity to look at risks and to see good things.
- Bring team together in preparation for next steps.
- Opportunity to stop and change approach.
- Use for plans, reports, product versions, decision points.
- Process, product and technical reviews.
- First in team, then in steering group or with customer.

Verification and validation

- Validation is checking that the product really meets its purpose – making the right product.
  - Is actually useful, usable or safe in real use.
  - Critical: 1) at the start – concept, requirements, validity of user stories, use cases, understanding of environment 2) at the end, when considering shipping, delivery, marketing, taking into use.
- Verification is checking that we have done as specified – making the product right.
  - Implementation matches the designs and specifications and so on.
  - Concentration on this at the middle of project / sprint.

Validating the key ideas early

- Key ideas need to be validated early:
  - New approaches, new concepts.
  - New everywhere or just in this context.
  - New technology.
- User interface:
  - Usability and UX analysis.
  - Assessment and testing with prototypes.
- Technology:
  - Proof of concept testing for technology.

Risk analysis

- Product risk analysis when the concept is defined and key characteristics knows:
  - What problems there might be for the customer, users in their activities. Safety risks. Security.
  - Technical risks.
  - Market acceptance risks.
  - Risks in product areas, quality factors.
  - Whole lifespan of product.
  - "What if…"
- Produces general understanding, improvements to plans. New requirements, things to test.
- Detailed analyses done later (like security, architecture).

Quality of designs

- Practices:
  - Design reviews. Testability reviews often in engineering industry.
  - Design analysis.
  - Task based analysis of activity.
- Design analysis:
  - Modelling and simulation. Formal verification.
  - Testing of prototypes, mock-ups.
  - Architecture analysis.
  - Reliability analysis using for example FMEA or fault tree analysis.
  - Usability analysis and human error analysis.

Quality of implementation

- Practices:
  - Testing at various levels.
  - Code review.
  - Static code analysis: code checkers that find problems in it, complexity analysis.

Some important principles of testing 1/5

Approach

- Take testing seriously, but enjoy it. It can be fun. New information of defects is a positive thing.
- Testing is a normal part of mature software development.

Timing and time for it

- Start testing as early as possible and do it continuously. Find problems and bugs early and fast.
- If you leave it to the end, it is too late – there will be no money or time left for it and no time to correct problems.
- There should be fast feedback and constant "dialogue" between testing and development.
- There must be time for testing – only when something is tested, it is done.

Some important principles of testing 2/5

Basis

- The idea is to create quality related information.
- Quality is relative, but the customer "decides" what is important.
- Understand the use of the product and its risks. Who uses it? For what purpose? How? What is important? What brings value? Problems and variations?
- Don't think too much about product technology – it is just a tool, but not value to customer
- Prioritise testing. Put most effort to testing most important things.
- The mental models of the system are different for the developers and the users – the final acceptance testing should be done by the customer.

Some important principles of testing 3/5

How to do it

- There are no silver bullets in testing.
- Good testing uses many practices that complement each other. Testing at all levels (unit, integration, system, acceptance), many testing paradigms – including exploratory testing and test automation.
- The ways of doing testing must be selected based on the context, criticality of the project and the product's requirements.
- In a small team everyone must test. Some with code, some through UI.
- Tests must challenge the system and the developers, try to break it! Negative testing is important.

Some important principles of testing 4/5

Documentation: What should be done?

- Plans.
- Testing tasks integrated to development or separate tasks in backlog.
- Requirements: Scenarios, user stories, use cases, other…
- (Test basis: user studies, business analysis…)
- Product areas that should be tested: functions, architectural elements, technologies…
- Prioritisation: What features (etc.) are more important than others
- Pre-planned test cases, scripts.

Some important principles of testing 5/5

Documentation: Current status

- Team collaboration: how things are happening (wiki, kanban board, tracking tools, etc.)
- Status of testing – what areas have been tested, which are ok, which are not. -> Status and maturity of the system. Lists, mindmaps.

Documentation: What has been done? Quality records

- Test cases, scripts.
- Test reports – overall (course report!) UX, performance, security…
- Test logs, especially for exploratory testing.

Documentation: What was found?

- Bug reports.

Monitoring quality

- To make quality and its progress visible, metrics should be used during project.
  - Open bugs.
  - Trend of open bugs. Number going up? Coming down?
- Metrics help in making data-based decisions about for example releases.

Quality of productisation

- When a product is nearing delivery, mindset (and sometimes people) change.
- No new implementations and their testing, but maturing the product: bug fixes and testing.
- Important to stop feature development and makes sure the product is robust.
- Need to check the overall product that it is ok. All elements, (digital) packaging, documents, store…
- Test all delivery mechanisms and media that they work robustly.
- QA here:
  - Checking that this is done.
  - Monitoring maturity metrics (bugs etc.).
  - Formal review of status: can we ship?

Learning and continuous improvement

- We can learn in and after every project.
  - From what went well and where there were problems and how we handled them.
  - Finding the causes of problems, like deficiencies in specifications or implementations, bad communication and so on. Then we can improve those.
- Lessons learned session periodically and after projects.
- Sprint reviews.
- Need to pass the learnings to other teams, other projects.
- Changing company practices based on learnings.

Key QA practices on TIE-PROJ type projects 1/2

- Mindset for overall quality
  - Understanding that these are not toy projects.
  - Usability and security analysis and testing included.
- Review or user and product requirements and their changes.
  - Understanding the concept.
  - Understanding the tasks, keeping the product focused.
  - Limiting focus to what can be done well.
- Testing focused on system level, corresponding with customer requirements.
  - Manual testing more important than test automation.

Key QA practices on TIE-PROJ type projects 2/2

- Good solid code in complete version control.
  - Unit testing for critical code.
- Review for deliverability and maintainability.
  - For us, the project audit in January.
  - A check that everything essential can and will be done properly.
  - Guidelines for that are on the web site.
  - Start orienting towards it.