Agile development of safety-critical software – while meetings standards' requirements

Matti Vuori, Tampere University of Technology

Contents 1/2

Contents 2/2

A study in Ohjelmaturva

- Identification of agile principles, process and activity elements and practices that are key issues from the viewpoint of developing safety-critical software systems.
- Analysis of the identified elements: risks of applying of them, how they should be supplemented, modified or avoided.
- Mapping of essential development tasks , quality and safety assurance into a generic agile development framework.
- Synthesis of key issues for guidance in process tailoring and development in companies.
- IEC 61508(-3) as context.
- The safety-criticality of the context of the study was moderate. We assessed mostly the safety integrity levels SIL levels 1, 2 and 3.

Tendency to be agile

- There is a tendency for companies to transform their software and product development practice into more incremental form.
  - Agile project management models.
  - Agile software lifecycle models.
  - Practices from Agile culture.
- The main value of agile processes comes from controlled increments and releases, which they produce more often than previous models.
- Controlled releases should especially help in getting feedback from the customer and to manage project risks.
- Familiar in other domain, now also in safety-critical development.
- However, the question still is, how it can be done so that any project features – safety, of course, being critical – are not compromised.

Benefits of Agile according to Larman (2004)

- Iterative development is lower risk; the waterfall is higher risk.
- Early risk mitigation and discovery.
- Accommodates and provokes early change; consistent with new product development.
- Manageable complexity.
- Confidence and satisfaction from early, repeated success.
- Early partial product.
- Relevant process tracking; better predictability.
- Higher quality; less defects.
- Final product better matches true client desires.
- Early and regular process improvement.
- Communication and engagement required.
- IKIWISI required [IKIWISI = I'll Know It When I See It].

Benefits in different product development settings

- Customer benefits in the development of tailored systems:
  - Getting an early release and understanding of the system by using it.
  - Getting regular releases at a sensible pace so that the new system can be learned and all necessary adaptations can be made in time.
  - Ability to make changes to plans during the process in a flexible way.
- Benefits in mass-market product development:
  - Manufacturer’s desire to control risks.
  - Ability to be fast in responding to competition and emerging market needs.

Positive conclusions for agile safety-critical development 1/2

- Customers need time to understand all safety percussions and early releases allow for that.
- Communicating design and safety information is easier, as agile emphasises verbal communication, not just written documentation.
- If a new technology should prove to be unreliable, the situation can be detected early and changes be made.
- Risk analysis has a gradually evolving scope (the concept), which should improve its quality.
- Safety assessments made in small increments can be more focused, delivering better results.

Positive conclusions for agile safety-critical development 2/2

- A rhythmic process of integration – at all levels, from code to customer production systems – should reduce many kinds of problems.
- Even if the increments are not targeted for production, their behaviour can be well tested, assessed and understood in simulation.
  - Any corrective measures can be based on practical observations from executing the system, not just analysis.
- Agile can provide a good learning experience for all involved, if applied properly.

Different goals => different approaches to Agile

- Goals influence how companies approach the development process.
- Some companies may start the agile process with a very vague idea of a concept.
- Some may see it as a way to make software production more controlled
- Others simply aim to get a row of productive solutions, new releases to customers.
- => All these require different processes and activities.

- Release orientation and release readiness are seen as a key element of agile approaches.

Risks of new processes

- Safety-critical development is all about managing risks.
- Just as in taking new components into use, new ways of action need to be carefully analysed.
  - Suitability.
  - Risks.
  - What the change influences.
  - Business risks, safety risks, quality risks.
  - Etc...
- Boehm and Turner present a rule that is particularly well suited to safety-critical development:
  - *Is it riskier for me to apply (more of) this process component or to refrain from applying it?*

Risk analysis of the transformation

- A risk analysis should be made for the process change in order to identify potential pitfalls.
- A transformation of project practices is very much a cultural change.
  - Pure processes are easy to change, but in software development it is a question of how people work together.
- Elements of software process – practices, processes, techniques and tools – can be thought to form a toolbox of items that we are free to choose from.
  - There many moments when we need to make a decision: shall we use that particular practice or not?

Problems in Agile culture according to Kruchten (2011) 1/2

1. Commercial interests censoring failure.
2. Pretending agile is not a business.
3. Failure to dampen negative behaviour.
4. Context and Contextual applicability of practices.
5. Context gets in the way of dogma.
6. Hypocrisy.
7. Politic.
8. Anarchism.
9. Elitism.
10. Agile Alliance.

Problems in agile culture according to Kruchten (2011) 2/2

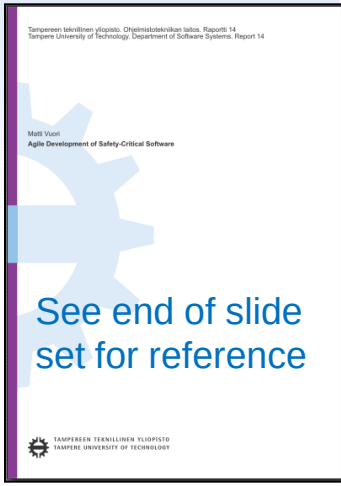
11. Certification (the “zombie elephant”).
12. Abdicating responsibility for product success (to others, e.g., product owners).
13. Business value.
14. Managers and management are bad.
15. Culture.
16. Role of architecture and design.
17. Self-organising team.
18. Scaling naïveté (e.g., scrum of scrums).
19. Technical debt.
20. Effective ways of discovering info without writing source code.

No generic "safety-critical system" under development

- Type of product and system.
  - From medical devices to machines to nuclear power plants.
- The role of software in the system.
  - Is it mainly a software-based system or is software still only in a restricted role and the product is perceived as a mechanical device, for example.
- The size and scope of the system.
  - Clearly small personal devices require a very differ-ent approach to large plant level systems.
- The risk level of the system.
  - Factory machines have a very different risk level than nu-clear power plants.

What we need to do in agile way?

- The process in IEC 61508-3 has this kind of properties:
  - Emphasis not up-front or end phases, but both and all in-between!
  - Requirement specification based on the overall system.
  - Quality of architecture.
  - Quality of plans and implementation.
  - Quality of verification – testing etc.
  - Safety assessment whenever designs change.
  - Validation in relation to what really is needed – in the context of the overall system.
  - Solid documenting with not gaps.
  - Very strong configuration management and tracing of things.
  - Competence and roles of process participants.



See end of slide set for reference

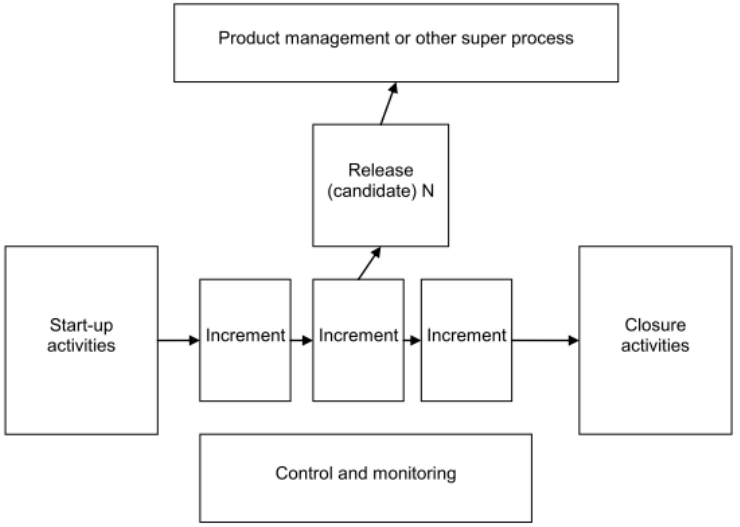
What is ”forgotten” in agile?... Hybrid approaches are needed

- By Coplien & Bjørnvig (2010):
  - Architecture.
  - Handling dependencies between requirements.
  - Foundations for usability.
  - Documentation.
  - Common sense, thinking and caring.
- Analytical approach.
  - Safety-critical development is very analytical by nature.
  - Must not be lost in transition.
  - Principles of ”Lean” are used to add some missing elements.
- Agile development is never fully agile.
  - Hybrid methods (see Kennaley (2010)) .
  - Each SDLC model is based on history.
  - => Hybrid models allow for choosing essential elements.

Anatomy of agile development

- Principles (including values).
- Project model.
- Software development lifecycle.
- Software engineering methods, techniques and practices.

- ...And beside that the whole global Agile culture...



Lean

- Lean is an approach that is often associated in the context of agile development.
- Lean was originally called “Lean production”, but today “Lean” represents only a vision of good efficient way of action.
- It has been applied mostly in manufacturing industries, but lately it has been coming into the software development world to complement agile development (see for example Poppendieck (2007)).
- Therefore, we included analysis of Lean principles in our study.
- Yet, Lean actually has very different principles than Agile.
- More about this in the report (Vuori, 2011).

Guidance for going agile

- Based on the study we offer the following guidance to companies who aim to develop their processes into more agile.

Prerequisites 1/2

- Shared sense of a need to change.
  - Agile is not a goal as such, but a means to meet some business or other goals.
  - For any change in an organisation, there should be a clear, shared sense of urgency to change processes and a feeling that the current way of action is not sufficient anymore.
- Solid process understanding and vision.
  - There should be an understanding of software and product development processes – agile and otherwise.
  - The aim is not to be agile as such, but get the best results, meet goals.
  - A nice read: Barry Boehm's and Richard Turner's book "Balancing Agility and Discipline – A Guide for the Perplexed".

Prerequisites 2/2

- Professionalism in action.
  - Before transformation the organisation needs to be able to fulfil, for example, all the requirements of IEC 61508 [2010] or other applicable standards with no problems. That provides a baseline.
- Experts and collaboration.
  - Expertise on agile is needed in the transformation, to guide the rest of the people in the transformation process.
  - Process people or consultants are needed who can lead the transformation.
  - Agile consultants do not necessarily understand all the process requirements of safety-critical development and the leading principles of safety and risk management.
  - Even world class agile experts may have a limited understanding of testing and verification and validation!

Ten generic guidelines for process development 1/3

1. Respect your own engineering skills and experiences in the process development.
2. Do not follow fashion.
   - Seek proven practices and analyse their suitability in your environment and culture.
3. Understand that not everything needs to be agile and that “more agile” will not mean “better”.
4. Remember that the more safety-critical the system and its development are, the more control is required for the process and that will usually mean less agility.
5. No single paradigm is sufficient.
   - Being just agile or just plan-driven is not enough.
   - Agility in some areas may need less agility in others, for specific process needs and for balance.

Ten generic guidelines for process development 2/3

6. Much of any company's current activity is tacit in nature and not formulated or documented.
   - Thus, you might be blind to your current success factors.
   - External consultants are often needed to see the essentials in your activities.
7. Safety-critical development is all about managing risks.
   - In the same way, you need to manage the risks of making process changes.
8. Find the style that is best for you, is the message in all development.
   - If you act like everyone else, how could you be the best in your branch?

Ten generic guidelines for process development 3/3

8. Process developments are never just mechanistic process issues.
   - Agility needs to fit into the corporate culture, which is a hard thing to change.
10. Co-operation and collaboration between all stakeholders is one key issue in Agile. The same applies to process development.

Develop basic elements that provide value for any process model

- More thorough unit testing and code quality assurance.
  - So that there is a solid base for changing the product.
- Continuous integration so that there is always a working product at hand, to assess how it behaves and to learn from it.
- Test automation.
  - Automating as many as possible of the tasks that are required for efficient verification and validation, yet supporting many test paradigms (among others exploratory testing) for diversity and test effectiveness.
- Agile process automation.
  - Automated reporting tools and information systems that can easily be tailored to changing situations in a project.
- Automated tracing.
  - When a requirement or design or implementation is being planned to change, or has changed, tools are needed to see immediately what it affects and what also needs to be addressed.

Make rules clear

- Who owns the products, the product business and safety?
- Who owns the development process?
- What are the principles of product development?
  - Is it customer-led or innovation-led?
  - Both may require different processes and participation.
- How do we weight the stakeholders' opinions.
  - Is product development a democratic process or is some party clearly leading it?
- How open do we really want to be towards the customers?
- How much do we really wish to engage subcontractors?

Communication and collaboration

- More self-reflection is needed in the process.
  - All participants need to look into the product and process frequently to see what in their way of action needs to be improved. This calls for frequent lessons learned / retrospect meetings in the process.
- Team building and improved leading of teams.
  - Teams need to able to work as teams and learning that, and learning to lead teams in more collaborative ways, takes a time.
- Improved communications tools.
  - Anything that helps people to communicate better, yet provides a peaceful environment for those tasks and for persons that will benefit from it.
- Improved communication between distributed sites (including any subcontractor sites).
  - No party should be left in a secondary role in communication.

Make time-consuming phases efficient

- Creation of external processes outside the incremental main development process.
- Clear understanding of at what times the validation related tasks need to be carried out.
- Making a very clear distinction between safety-critical and other requirements, and SIL levels
  - So that the elements requiring validation or re-validation can be identified exactly.
- Efficient information systems so that any paperwork or routines or finding and checking of development records do not take time.
- Flexible arrangements for validation.
  - If external validators are used, the situation can be reassessed – could the validation be carried out internally?

Provide control

- Split requirements into smaller items, aiming for similar size to aid in planning and estimation.
- Improve meetings – make them more frequent, get all people to participate; develop meeting cultures.
- Create dashboards and information systems that visualize the project's status to every-one.

Split the process

- A chain of events from planning of what to do, to evaluation of what has been achieved.
- Aiming for a demonstrative whole at the end of each increment.
- True planning in the beginning of an increment.
- Utilising real product development practices in defining requirements.
  - That is, the input should be something that provides value: new use case, user story or similar, but NOT a technical specification or a change request.
  - Change requests belong in the domain of maintenance, not in the domain of new product development lifecycle.

Increase number of releases

- Practice the making of release plans that are rougher and based on value and risk, not just technology and functions.
- The increment-based lifecycle can be exercised during implementing – select features to implement and test during an increment and in the second phase, transfer more design work inside the increments.
  - At the same time, the practices of architecture design need to be developed so that it can also evolve sufficiently.
- Development of release processes and practices so that all steps leading to the customer's utilisation of the product are as efficient as possible, yet do not compromise safety in any way.
- When the processes are in good shape, releases can be added in a controllable manner if there is benefit gained from that.

Add any other practices that bring value

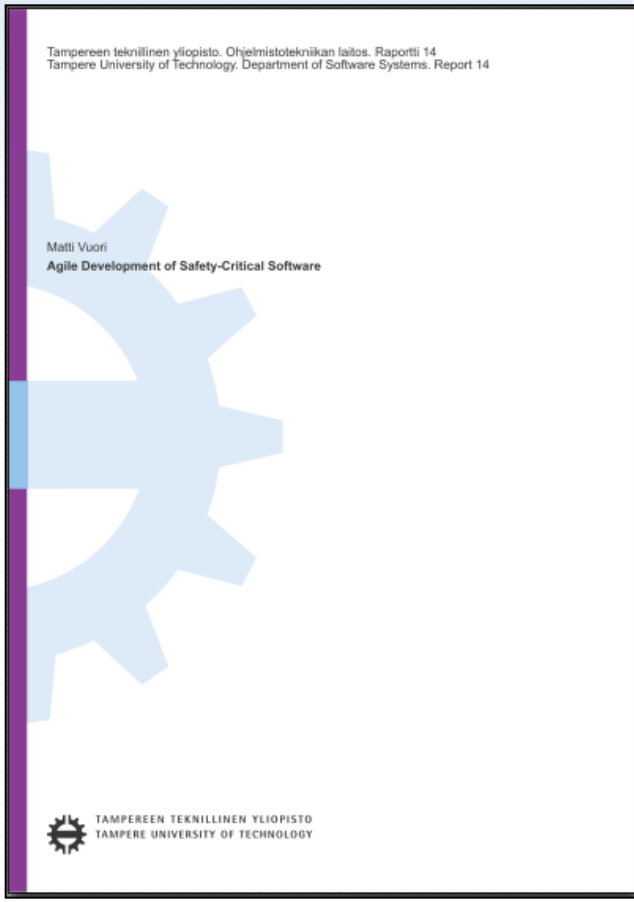
- There are and will be many good practices in the agile culture.
- The idea is not to implement all of them, but only those that really bring value.
  - For example, while pair programming – an important old agile practice – seems to have lost its appeal somewhat lately, it could be a tool in transferring practical knowledge and experience from older developers to new developer generation.
- Any mature software development shall not restrict its views to one paradigm only.
- The current agile culture did not just emerge, but is a result of process evolution, combining ideas and practices into a new whole.
- So, for new process development ideas one should look into many relevant areas, not just agility.
- Yet, in agile, things need to be kept simple.

Continuous improvement

- Keep improving the process and meeting any problems one at a time.
- The agile processes applied need to be developed further, just like any processes.
- Processes are only optimal in a given context and learning changes that – not to mention any external influences from technology, changing standards and so on.
- Continuous improvement should have an important role in any organisation.

Read the report for more info

- *The full report, containing much more information:*
  Vuori, M. 2011. Agile Development of Safety-Critical Software. Tampere University of Technology. Department of Software Systems. Report 14. 95 p. Available at: http://urn.fi/URN:NBN:fi:tty-2011061414702.



References 1/2

- Vuori, M. 2011. Agile Development of Safety-Critical Software. Tampere University of Technology. Department of Software Systems. Report 14. 95 p. Available at: http://urn.fi/URN:NBN:fi:tty-2011061414702.
- Wikipedia. Agile software development. Available at: http://en.wikipedia.org/wiki/Agile_software_development
- Larman, C.. Agile and Iterative Development. A Manager's Guide. Addison-Wesley. 342 p.
- Kruchten, P. 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/

References 2/2

- Kennaley, M. 2010. SDLC 3.0. Beyond a Tacit Understanding of Agile. Towards the next generation of Software Engineering. Fourth Medium Press. 280 p.
- Poppendieck, M. & T. 2007. Implementing Lean Software Development: From Concept to Cash. Addison-Wesley. 276 p.
- Boehm, B. & Turner, R. 2003. Balancing Agility and Discipline – A Guide for the Perplexed. Addison–Wesley. 266 p.
- Coplien, J.O. & Bjørnvig, G. 2010. Lean Architecture: for Agile Software Development. Wiley. 376 p.