Use of Agile Practices when developing Safety-Critical Software

T. Myklebust. SINTEF ICT, Trondheim, Norway

T. Stålhane. NTNU, IDI, Trondheim, Norway

G. K. Hanssen. SINTEF ICT, Trondheim, Norway

Keywords: Agile Practices, SafeScrum, Safety-critical SW

Abstract

Objectives: During the last 10 years there has been an increasing use of agile development methods and practices when developing safety-critical software, in order to shorten the time to market, to reduce costs, to improve quality and to have more frequent releases. Several of the agile practices cannot be used as-is when developing Safety-Critical Software (SCSW).

There are many agile practices but we are searching for those agile practices that can be used to obtain agility when developing SCSW. We have evaluated several practices and suggested how to add necessary safety aspects to these practices. In addition we have evaluated how to adapt the practices to development of SCSW when using SafeScrum, an adaptation of the Scrum agile software development methodology, ensuring that safety standards like IEC 61508 (part 3) are satisfied.

Methods: In this paper we have analysed agile practices commonly used in software development projects. The acquired information is used to suggest how to include add-ons to the practices and how to adapt the processes to the development of SCSW. We have also performed a literature study and checked:

- What are the most adopted agile practices?
- Which methods are suitable when developing SCSW?

Results: The descriptions of the agile practices were assessed and consolidated.

Several add-ons and adaptations are suggested for agile practices. Three new extended agile practices have been suggested; the "Backlog Splitting", the "STDD" (Safety TDD) and "Four questions".

The paper starts by presenting and clarifying relevant terms and definitions, as these may differ between the agile community and the safety community. A short introduction to SafeScrum is presented together with some of the agile practices. The main part of the paper structures and describes the relevant agile practices together with suggested add-on's and adaptions.

Conclusions: There exist more than 50 named agile practices. Several of these practices cannot be used as-is when developing SCSW since they do not meet mandatory safety requirements. We have evaluated 10 of the most relevant practices and described necessary add-ons and adaptions to ensure that important international standards like IEC 61508 are satisfied. The practices have been described as part of the SafeScrum method.

1 Introduction

In the last years there has been an increasing use of agile practices when developing safety-critical software in order to reduce time to market, reduce costs and to improve quality.

Companies introducing agile methods like Scrum, also have to include relevant Agile practices to get the full benefit of an agile approach.

A practice in software development is considered to be a working activity and it is required that the activity can be repeated. In this paper we have limited the practices to Agile practices but some of them may be used also when doing a waterfall/V-modell approach. Agile practice is named Agile Techniques in [VersionOne].

For safety critical systems, however, some agile development practices does not fit as-is, but has to be adapted and/or extended by safety aspects.

Acknowledgements: This work was partially funded by the Norwegian Research Council under grant #228431 (the SUSS project) and SINTEF SEP project Safe Software.

2 Background

In the subchapters below we have described the agile approach and agile practices in general and how some of them are used as part of SafeScrum [Stålhane et. al 2011].

There has been little experience published on the use of agile practices for use together with IEC 61508:2010 and for safety critical software in general.

2.1 Agile practices

There exist more than 50 named agile practices. Several of these practices cannot be used as-is when developing SCSW as they do not include adapted parts that are mandatory for safety requirements. We have evaluated 10 of the most relevant practices and described necessary add-ons and adaptions in chapter 3, to ensure that important international standards such as IEC 61508 and EN 50128 are satisfied.

There are unfortunately few papers published that discusses the important question of which are the most used agile practices. The majority of information in this area is thus taken from blogs. This is not surprising – blogs are mostly two to four years ahead of scientific papers when it comes to the cutting edge for most aspects of software engineering.

Some of the practices that are collected under the heading “Agile practices” are new – e.g., the daily scrum – while some of them are old – e.g. incremental development [Larman and Basili]. In addition, some of them are old but are used in a new way – e.g. post-mortem analysis [Birk et al.], now called retrospectives. A survey performed by [ProjectSmart] shows that 26% of all software development companies use one or more agile development methods. We will discuss two aspects of agile practices – (1) which practices are often used [VersionOne] and (2) which have been selected most often in the development of safety-critical software [Kelly].

The [VersionOne] report shows that the five most used agile practices are the daily stand-up/scrum (83%), a prioritized backlog (82%), short iterations (79%), retrospectives (74%) and iteration planning (69%), while for instance only 24% use pair programming. The selected five most important practices

reported are the same in 2014 and 2015. This survey had 3880 respondents. Two-thirds of the survey respondents said they worked in software organizations with more than 100 people. About 50% of the respondents worked for the software industry, financial services and professional services. But also safety related industries took part, 3% from healthcare and 2% from transportation. 56% of the respondents are from North America and 26% from Europe.

T. Kelly has performed a survey in the UK. Among the questions used in the survey one is of special interest here: "Which of the following practices of agile development can contribute to safety-critical systems development and assurance?" They had a total of 69 respondents, thus all differences larger than 0.12 are significant at the 5% level. The five most popular results were as follows: (1) simple design – 67%, (2) continuous integration – 61%, (3) release planning – 56%, (4) pair programming – 50% and (5) small and short releases – 44%. Except for "short releases", the two top-five sets (VersionOne and Kelly's survey) are disjoint.

The large differences between the survey results from VersionOne and Kelly can, at least partly, be explained by different populations. VersionOne looks at the whole software community, while Kelly just looks at those developing safety-critical software. The top five practices from VersionOne are mostly related to administrative matters, while three of Kelly's top five items are related to development – e.g. simple design.

Thus, it seems that what is important agile practices will depend on your area of interest. In reality, we have to solve both administrative and software development concerns, thus both areas are important. However, the [Chaos report] identifies that the two main reasons for projects to go wrong are bad management and bad communication. The Chaos report has received a certain amount of flak from academia – see e.g. [Eveleens and Verhoef]. Their main objection is the definition of two of the project categories – success and challenged – based on adherence to time and budget. We have, however, only used the category impaired, which is not criticized. In addition, the IT Professional Facilitator [IT Professional Facilitator] has also published their top ten list of cause for failure. They are all related to management and communication. However, we cannot just focus on avoiding failure. We also need to develop a product and we thus add development practices. This leads us to recommend projects to focus on the following ten agile practices (the numbers in brackets refer to Table 1 below):

- Communication – how is the information flow organized in the project; the number one priority for any successful project. Here we include the daily scrum (4), sprint review (9) and retrospectives (10)
- Planning – what to do when. Here we include incremental planning which include a release plan (5, 6, 7) and a prioritized back-log (2, 3).
- Development – how to do it. Here we include a simple design (see Kelly's survey), short iterations and frequent releases and stepwise integrations (7, 8), which are needed due to the planning items included. In order to care of the safety aspects, we also need to include safety validation and verification here (1, 2).

2.2 SafeScrum and Agile practices

SafeScrum is a variant of the well-known and extensively used Scrum development model [Schwaber and Beedle] with some additional elements added to be able to fulfill the process requirements from the IEC 61508 standard. Scrum and SafeScrum can be seen as a set of fundamental agile principles

and practices, composed to organize an agile software development project. A Scrum process is organized as a series of sprints, which are time boxed iterations, typically 3 or 4 weeks. Each sprint contains all typical development activities like detailed requirements specification, development, testing, and integration. Thus, a Scrum development project can be understood as a series of sprints where each is a mini waterfall project. The fundamental idea is that requirements can be refined and re-prioritized between each sprint, based on increasingly growing knowledge of the system under development and the problem it is intended to solve. This enables agile and flexible management of requirements. Requirements are organized into a product backlog, which is defined at the start of a development project. Each sprint starts with a sprint planning meeting where top-prioritized requirements (also called issues) are added this to the sprint backlog, which then guides the work in the upcoming sprint. A product owner is responsible of making priorities in collaboration with the development team and the Scrum master, which is responsible of making sure that the Scrum process works effectively. After one or more sprints the process is supposed to deliver a working piece of the final solution, meaning that development is both iterative and incremental. Each sprint ends with a sprint review meeting where results are evaluated, and if needed, a sprint retrospective where the scrum process itself is evaluated and potentially improved. Each working day within a sprint starts with a short stand-up meeting where the team members each explains results from the previous working day, plans for the day and potential problems that needs attention. The Scrum master is responsible of solving problems hindering the work.

Based on a thorough investigation of the requirements in part 3 of the IEC 61508 standard (which defines the software part of the total system) [Stålhane et al. 2011], we have proposed a set of extensions and adjustments to make Scrum applicable to development of safety critical software. Firstly, there are two product backlogs, one for functional requirements and one for safety requirements. Functional requirements may change frequently whilst safety requirements normally are stable and even reusable between projects and products. Relationships between these two set of requirements are maintained to keep track of which safety requirements that are affected by which functional requirements. Secondly, SafeScrum needs to be a traceable process. All decisions and changes throughout development must be documented, stored and made available to the assessor to prove conformance to the standard. The same goes for code reviews where all remarks and how they were resolved needs to be kept track of. Thirdly, each sprint encompasses a validation of reliability, availability, maintainability and safety (RAMS) of the present system are validated. As part of the sprint review of each sprint, the product backlog may be updated. In cases where a change is considered to affect the safety of the system, a change impact analysis (CIA) [Myklebust et al. 2014 and Stålhane et al. 2014] is done – this also needs to be documented. Here the two backlogs come in handy as a mean to identify how a change related to a functional requirement potentially influences a safety requirement. Besides these extensions, common agile practices are important, like test-driven development (important to establish high test coverage), regular work iterations, daily stand-ups, and stepwise integration.

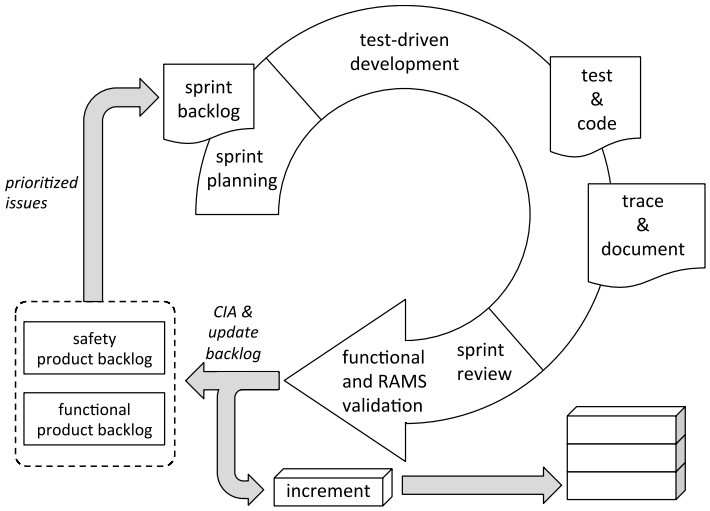

Figure 1 The SafeScrum model

The series of sprints replaces the 'coding' part and the 'evaluation' part of the V-model. This means that documentation is produced continuously and *as a part of* development and not as a finalizing phase as it is described by the V-model. A simple way of putting it is that SafeScrum replaces the bottom and right side of the V-model. This is fundamentally important for *software* development and enables a project to become more flexible with respect to changes and still be able to provide the needed documentation and traceability to the assessor.

In addition to the basic agile practices, which constitutes SafeScrum, other practices may also be taken into use to complement and strengthen development. The rest of this paper discusses how they may be applied to development of SCSW.

3 Evaluation of agile practices and safety

In the subchapters below we have evaluated the most important agile practices for SCSW and described adaptations of three agile practices.

3.1 Popular practices relevant for SCSW

The top ten practices was established as a combination of the most frequently used practices and our evaluation of how the practices fits into development of safety critical software. The four Scrum practices, sometimes also named ceremonies (4, 5, 9 and 10 below) are included together with six more practices.

Table 1: Ten most important Agile practices when developing SCSW

Practice	Category	General comments	Safety adaptations
1. Acceptance testing and the related STDD	Development	Test first is a good practice. And associating a test with every piece of functionality is brilliant according to [Meyer]. Normally A customer acceptance test is used to verify that an application behaves in the way that a customer expects, while within the safety domain this can be a test to check whether the product or system satisfy the safety tests required by e.g. IEC 61508-3. [VersionOne] has automated acceptance tests ranked as number 20. Automating the tests requires a lot of work and. That acceptance testing is far more important when developing SC-SW is obvious as often a safety case is necessary, assessors and even authorities can be involved.	Acceptance testing is of crucial importance for SCSW and several tests are required as part of the • validation plan (IEC 61508-3 chapter 7.3), • Test specifications (e.g. IEC 61508-3 requirements 7.4.7.1, 7.4.8.2, 7.4.8.3, 7.5.5.2, 7.5.2.3 - 7.5.2.5, 7.9.2.10 and 7.9.2.11) that has to be developed and • the Tables in Annex A+B in IEC 61508-3 (including regression testing). Regression testing, that is especially important when using incremental SW development, should be improved in the next edition of IEC 61508-3. Regarding STDD, see chapter 3.2 below.
2. Backlog together with backlog splitting	Planning	Prioritized work list is also closely linked to this practice, see 4 below.	Important part of SafeScrum [Stålhane et al. 2011]. Regarding backlog splitting, see chapter 3.2 below.
3. Prioritized work list	Planning	Prioritizing the backlog items may be performed by using Index cards or similar. [VersionOne] has prioritized backlogs ranked as number 2.	RAMS or e.g. the Safety manager should be involved in the prioritization. Together the Agile Safety Plan [Myklebust et. al 2016], t High Level Safety Plans and the Sprint planning constitutes the main Agile plans.

Practice	Category	General comments	Safety adaptations
4. Daily scrum meeting (DSM) including four questions	Communication	Findings by [Stray et al.] shows that DSMs may not necessarily have to be held daily, and focus in the meetings should be on discussing and solving problems, and planning for the future rather than reporting what has been done. Furthermore, it is beneficial to be standing in the DSMs and to conduct the meetings by a task board. This empirical evidence corresponds to the experience by one of the authors. [VersionOne] has daily standup ranked as number 1.	[Paasivaara et al.] examined agile practices in global software development and found that DSMs helped reveal problems early which is very important when developing SCSW. Regarding four questions, see chapter 3.2 below.
5. Sprint planning meeting	Planning	The sprint planning meetings are attended by the product owner, ScrumMaster and the entire Scrum team. According to [Kniberg], sprint planning is a critical meeting, probably the most important event in Scrum. [VersionOne] has iteration planning ranked as number 5.	This together with high level plans are the main part of the planning activity [Myklebust et al. 2016].
6. Timebox	Development	The duration of the sprint is formal and takes place within a specified timeframe, which ranges from two weeks to a month. Scrum meetings, which last about 10-15 minutes each day, are also formal. Timebox is a brilliant practice according to [Meyer].	Important part of SafeScrum. See also Incremental development below.

Practice	Category	General comments	Safety adaptations
7. Incremental development including iteration and stepwise integration	Development	Development of software is in fact normally incremental, e.g. product releases and maintenance releases. Research at The Standish Group indicates that smaller time frames, with delivery of software components early and often, will increase the success rate. Shorter time frames result in an iterative process of design, prototype, develop, test, and deploy of small elements [Chaos1995]. According to [Meyer] iteration is brilliant: *Short iterations are perhaps the most visible influence of agile ideas, an influence that has already spread throughout the industry. Few competent teams today satisfy themselves with six-month objectives. The industry has understood that constant feedback is essential, with a checkpoint every few weeks.* [VersionOne] has short iterations ranked as number 3.	A stepwise integration as part of sprints is an integrated part of the SafeScrum approach. Often there are a few iteration sprints before an integration (increment) are performed into a testable system. The agile experts often mention "continuous integration" instead of "stepwise integration" but "continuous integration" is more difficult when developing SC-SW. E.g. Safety cases may have to be finalized and assessors may be involved.
8. Shippable code	Development	This is also named "delivering working software" by [Meyer] and is considered as brilliant: *The emphasis on delivering working software is another important contribution. We have seen that it can be detrimental if understood as excluding requirements, infrastructure and upfront work. But once a project has established a sound basis, the requirements to maintain a running version imposes a productive discipline on the team.*	Several opponents against Agile have claimed that this is not possible when developing SCSW, but our studies has shown that this is possible (http://safescrum.no/).

Practice	Category	General comments	Safety adaptations
9. Sprint review	Communication	At the end of each sprint, a sprint review meeting is held. [VersionOne] has iteration reviews ranked as number 10.	Important part of SafeScrum. In SafeScrum, after a planned number of sprints [Myklebust et al. 2016], the project is required to deliver a potentially reviewable product increment. This means that at the end of each incremental sprint, the team has produced a coded, tested and usable piece of software. The RAMS or e.g. the safety manager may be involved in the sprint review.
10. Retrospective	Communication	This is a dedicated period at the end of one or more sprints to deliberately reflect on how they are doing and to find ways to improve. Copy from [Kniberg]: *Sprint planning is a critical meeting, probably the most important event in Scrum (in my subjective opinion of course). A badly executed sprint planning meeting can mess up a whole sprint. Important? Yes. Most important event in Scrum? No! Retrospectives are waaay more important!* [VersionOne] has retrospectives ranked as number 4.	Important part of SafeScrum. The RAMS or e.g. the safety manager may be involved in the retrospective.

3.2 Extended agile safety practices

Three new extended agile practices have been suggested; the "Backlog Splitting", the "STDD" (Safety TDD) and "Four questions".

Backlog splitting was introduced as part of the introduction of SafeScrum [Stålhane et al.2011].
In SafeScrum, all requirements are split into safety critical requirements and other requirements and inserted into separate product backlogs. Alternatively, the safety requirements are tagged. Adding a second backlog is an extension of the original Scrum process and is needed to separate the frequently changed functional requirements from the more stable safety requirements. With two backlogs we can keep track of how each item in the functional product backlog relates to the items in the safety product backlog, i.e. which safety requirements that are affected by which functional requirements. This can be done by using simple cross-references in the two backlogs and can also be supported with an explanation of how the requirements are related if this is needed to fully understand a requirement. The staffing of the *Sprint team* and the duration of the sprint (30 days is common), together with the estimates of each item decides which items that can be selected for development. Sometimes also e.g. the Safety responsible or the RAMS responsible takes part in the selection of which items have to be prioritized.

The STDD: Safety TDD is a modified use of TDD when developing safety-critical software. The good parts with TDD is that you take small steps when writing software. Other benefits are that every piece of functionality has an associated test. Well-written unit tests also serve as excellent design documentation that is always, by definition, in synch with the production code.
When developing safety-critical software it is important to have in mind the abstractions and perspectives.
Abstraction is a key element of good software design. Abstraction helps encapsulate behavior, decouple software elements, having more self-contained modules and manage complexity.

In addition, abstraction makes the application extendable in a much easier way together with making refactoring easier. When developing with a higher level of abstraction, you communicate the behavior and less the implementation.

Safety perspective includes e.g. defensive programming.

Below is our STTD shown together with the TDD by [Beck]:

Table 2: TDD and Safety TDD

Pure Agile	Safety approach: STDD
1. Quickly add a test	1. Add a test if possible. Some tests may have to be performed by someone outside the Scrum team and analysis may also be required (see comments to Acceptance tests in Table 1). Some of these tests may have a link to acceptance tests
2. Run all tests and see the new one fail	2. Run most of the tests (selection based on e.g. impact analysis and resulting regression tests) and see the new ones fail
3. Make a little change	3. Make the relevant change
4. Run all tests and see them all succeed	4. Run most of the tests (based on e.g. impact analysis and resulting regression tests) and see them all succeed
5. Refactor to remove duplication	5. Refactor to remove duplication

The Four questions have been introduced as part of the daily scrum. The questions are the tree normally used as part of the daily scrum plus one safety question.

1. What work did You complete yesterday?
2. What have You planned for today
3. Are You facing any problems or issues
4. Any safety impact?

Adding the safety question 4 is especially important for organizations that develop both safety and non-safety products.

4 Conclusion

There exist more than 50 named agile practices. Several of these practices cannot be used as-is when developing SCSW as they do not include mandatory safety requirements. We have evaluated 10 of the most relevant practices and described necessary add-ons and safety adaptions to ensure that important international standards like IEC 61508 are satisfied. All these 10 practices may contribute to shorten the time to market, to reduce costs, to improve quality and to have more frequent releases.

Three new extended agile practices have been suggested; the "Backlog Splitting", the "STDD" (Safety TDD) and "Four questions".

References

1. T. Myklebust, T. Stålhane, G. K. Hanssen, T. Wien and B. Haugset. Scrum, documentation and the IEC 61508-3:2010 software standard. PSAM 12 Hawaii 2014
2. T. Myklebust, T. Stålhane, B. Haugset and G. Hanssen. Using a goal-based approach to improve the IEC 61508-3 software safety standard. Proceedings of the twenty-third safety-critical system symposium, Bristol, UK 3rd-5th February 2015
3. T. Stålhane, T. Myklebust and G. Hanssen. The application of Safe Scrum to IEC 61508 certifiable software. PSAM11/ESREL 2012. Helsinki June 2012.
4. T. Myklebust, T. Stålhane, G. K. Hanssen and B. Haugset. Change Impact Analysis as required by safety standards, what to do? PSAM 12 Hawaii 2014
5. T. Stålhane, V. Katta and T. Myklebust. Change Impact Analysis in Agile Development. EHPG Røros 2014

6. T. Stålhane, G. K. Hanssen, T. Myklebust and B. Haugset. Agile change impact analysis of safety critical software. SafeComp_Sassur, 2014
7. T. Myklebust, T. Stålhane and N. Lyngby. Application of an Agile Development Prosess for EN 50128/Railway conformant software. Esrel 2015
8. T. Myklebust, T. Stålhane and N. Lyngby. The Agile Safety Plan. PSAM13, 2016
9. K. Schwaber and M. Beedle. Agile software development with Scrum. Prentice Hall 2001
10. K. Beck. Test-Driven Development. By example. Addison-Wesley. 2003
11. H. Kniberg. Scrum and XP from the trenches. C4Media. Second edition. 2015
12. V. Stray, D. I. K. Sjøberg and T. Dybå.The daily stand-up meeting: A grounded theory study. Article in Journal of Systems and Software · January 2016
13. B. Meyer. Agile! The Good, the Hypa and the Ugly. Springer Ed.1, 2014
14. Paasivaara, M., Durasiewicz, S., Lassenius, C., 2008. Using Scrum in a globally distributed project: a case study. Softw. Process: Improv. Pract. 13, 527–544. doi:10.1002/spip.402.
15. Birk, A., T. Dingsøyr and T. Stålhane (2002). "Postmortem: Never Leave a Project without It." IEEE Software 19(3): 43 - 45.
16. VersionOne – 2015
17. Chaos1995. The Standish Group
18. Kelly, XP2015: http://dl.acm.org/citation.cfm?id=2894798&CFID=784458267&CFTOKEN=79745997
19. Project Smart 2014: www.projectsmart.co.uk/ last visited May 11, 2016
20. Eveleens and Verhoef. IEEE Software 27.1 (Jan/Feb 2010): 30-36.
21. IT Professional Facilitator. http://itprofessionalfacilitator.com/itpro/ last visited May 11, 2016
22. C. Larman and V. R. Basili. Iterative and Incremental Development: A Brief History. IEEE Computer Society 2003

BIOGRAPHIES

Thor Myklebust; Certification manager, Cand. Scient. Physics. System Safety
SINTEF ICT Norway. e-mail: thor.myklebust@sintef.no
Has experience in certification of products and systems since 1987. Myklebust has participated in several international committees since 1988. Member of safety (NEK/IEC 65), IEC 61508-3 committee, railway (NEK/CENELEC/TC 9) and NB-rail (notified bodies) since 2007. Chairman of NB-rail in the period October 2014 – October 2015. Founder of SafeScrum.

Tor Stålhane: PhD and Professor Emeritus
Department of Computer and Information Science, Norwegian University of Science and Technology (NTNU). e-mail: stalhane@idi.ntnu.no
He has a PhD in statistics from the Norwegian University of Science and Technology (NTNU). He has experience in software engineering topics such as software process improvement, reliability assessment, quality assurance and cost estimation. His current area of interest is safety analysis of software intensive systems. Founder of SafeScrum.

Geir Kjetil Hanssen: PhD and Senior Research Scientist
Software process improvement and knowledge management, SINTEF ICT Norway.
e-mail: geir.k.hanssen@sintef.no
He has a PhD in software engineering from the Norwegian University of Science and Technology (NTNU). His main areas of interest are software engineering methodologies, software process improvement, software ecosystems and software project management. Founder of SafeScrum.