# Agile practices when developing safety systems

**Thor Myklebust[a], Narve Lyngby[a], and Tor Stålhane[c]**
[a] SINTEF Digital, Trondheim, Norway
[c] NTNU, Trondheim, Norway

**Abstract:** In the recent years, there has been an increasing use of agile practices when developing safety-critical software. We have evaluated 50 of the most relevant agile practices and described necessary add-ons and adaptions to ensure that important international safety standards like IEC 61508 are satisfied. The evidence for the safety system are presented in an Agile Safety Case. For some of the customized practices we also have empirical information. In addition, we have included the practices into SafeScrum - an expansion of Scrum for the development of safety-critical software.

**Keywords:** Agile practices, SafeScrum, Scrum, Safety case, IEC 61508.

## 1. INTRODUCTION

In the recent years there has been an increasing use of agile practices when developing safety-critical software (SCSW) in order to reduce time to market, reduce cost, improve quality and enable frequent updates of the software. An agile approach together with the agile practises facilitates the development of only the necessary documentation [3] like e.g., an Agile safety case [1]. This is also an important part of a DevOps [20] approach for safety systems.
Companies introducing agile methods like Scrum, should also include relevant Agile practices to get the full benefit of an agile approach. However, it is also possible to introduce several agile practices while still having a traditional approach.

A practice in software development is considered to be a working activity that can be repeated. Some practices are old e.g., incremental SW development. Other practices are from the traditional software development approach but have become more important when having an agile approach like e.g., version control due to more frequent releases

**Table 1: Alphabetical overview of practices**

| | | | |
|---|---|---|---|
| 1. Acceptance testing | 13. Definition of ready | 28. Relative estimation | 41. Team-based estimation |
| 2. Automated build | 14. Epic | 29. Safety story | 42. Team communication |
| 3. Automated tests | 15. 40 hour week | 30. Security story | 43. Test driven requirement |
| 4. Burn down chart | 16. Frequent releases | 31. Simple design | 44. Test first development |
| 5. Backlog | 17. Hazard story | 32. Shippable code | 45. The wall |
| 6. Backlog refinement | 18. Incremental | 33. Short iterations | 46. Timebox |
| 7. Backlog splitting | 19. Information radiators | 34. Sprint | 47. Unit testing |
| 8. Coding standard | 20. Integration | 35. Sprint planning | 48. Usability testing |
| 9. Collective code ownership | 21. Iteration | 36. Sprint retrospective | 49. Velocity |
| 10. Continuous deployment | 22. Open office space | 37. Sprint review | 50. Version control |
| 11. Daily scrum | 23. On-site customer | 38. Stepwise integration | |
| 12. Definition of done | 24. Prioritized work list | 39. Story | |
| | 25. Quick design session | 40. Taskboard | |
| | 26. Refactoring | | |
| | 27. Release plan | | |

When developing a safety product, one should pick the relevant techniques for that project, while other practices may be relevant for other projects, even at the same company. For safety-critical systems, however, agile development practices do normally not fit as described by the agile community but have to be adapted and extended to accommodate safety aspects.

## 2. RESEARCH AND INNOVATION METHODS INCLUDING RESEARCH QUESTIONS

We have analysed agile practices commonly used in software development projects. The practices have been analysed trough review of the practices, the practices have been discussed as part of reformulation towards safety and discussed with safety and agile experts. Reformulation has been performed having general safety expertise in mind but also taking into considerations the requirements presented in international standards such as IEC 61508 series (generic), EN 5012X series (railway) and ISO 26262 series (automotive).

The research questions are listed below:
RQ1: What are the most adopted agile practices?
RQ2: Which practices are suitable both for traditional and Agile processes?
RQ3: Which practices are suitable when developing SCSW?
RQ4: Which practices are suitable in the first safety phases?
RQ5: Which practices are relevant for each of the three main challenges (requirement management, bad management and bad communication) when developing SCSW?

## 3. AGILE, SAFESCRUM AND THE AGILE SAFETY CASE

There exists more than 70 named agile practices. Several of these practices cannot be used as-is when developing SCSW as they do not include parts that are mandatory for safety requirements. We have evaluated 50 of the most relevant practices, shown in Table 1, and described necessary add-ons and adaptions in chapter 4.3 to ensure that important international standards such as IEC 61508, EN 50128 and ISO 26262 can be satisfied.
There are, unfortunately, few papers published that discusses the important question of which are the most used agile practices. We consider those practices in general as most interesting as more information and experience exists for these practices.
We have looked at the combination of two views – (1) the most used practices and (2) how they can be combined with challenges and resulting failures when developing software. The IT Professional Facilitator [16] has published a 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 several agile practices:

- Practices related to Communication – how is the information flow organized in the project; the number one priority for any successful project.
- Practices related to Planning – what to do when.
- Practices related to Development – how to do it.
- Practices related to Management

All practices, from concept to the resulting agile safety case [1] have been evaluated. In the figure below we have shown several practices together with the SafeScrum approach.

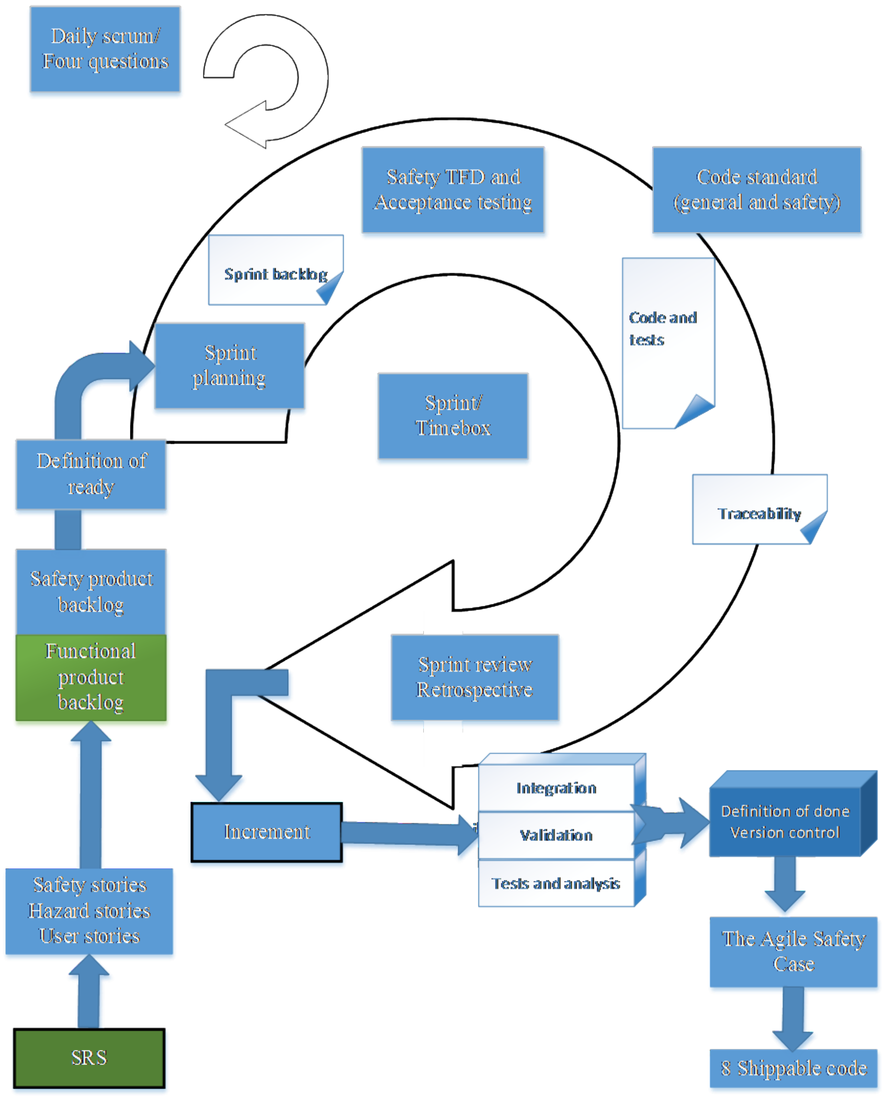

Daily scrum/ Four questions
Safety TFD and Acceptance testing
Code standard (general and safety)
Sprint backlog
Code and tests
Sprint planning
Sprint/ Timebox
Definition of ready
Traceability
Safety product backlog
Functional product backlog
Sprint review Retrospective
Integration
Validation
Tests and analysis
Increment
Definition of done Version control
Safety stories Hazard stories User stories
The Agile Safety Case
SRS
8 Shippable code

**Figure 1: Upfront activities and Sprint together with Agile practices and finally issuing of the Agile Safety case**

## 4. AGILE PRACTICES WHEN DEVELOPING SAFETY SYSTEMS

### 4.1 The most adopted agile practices

The most adopted agile practices have been based on interviews with SW engineers, [11], [12], [15] and [18]. They are shown in the Table below.

**Table 2: The most adopted agile practices**

| The most adopted agile practices | |
|---|---|
| Backlog | Sprint review |
| Backlog splitting and prioritized backlog | Refactoring and simple design |
| Daily scrum | Iteration planning |
| Short iterations | Release planning |
| Acceptance testing | Stories (User stories, Safety stories and Hazard stories) |
| Incremental development | |

## 4.2 Practices suitable both for traditional and Agile processes

We have evaluated which practices can be used both when having a traditional and agile approach. The evaluation has been based on the author(s) experience with companies using these practices also when having a traditional development process.

**Table 3: Practices that can be used both in traditional and agile projects (alphabetical order)**

| Practice that can be used both in traditional and agile projects | |
|---|---|
| Acceptance testing | Prioritized work list |
| Automated tests | Refactoring and simple design |
| Backlog (Backlog splitting, Backlog refinement and backlog grooming | Stories (User stories, Safety stories and Hazard stories) |
| Daily scrum/Four questions | Test first development |
| Definition of done and definition of ready (see [22]) | Usability testing |
| Epic | Version control |

## 4.3 Practices suitable when developing SCSW

The practices described below were established as a combination of

- the most frequently used practices, based on interviews with SW engineers, [11], [12], [15] and [18].
- our evaluation of how the practices fits into development of safety-critical software
- practices that can be used both when having a traditional and agile approach (but not all the practices listed below, see Table 2 above)

A selection of important agile practices and how they can be adapted when developing safety-critical software are presented below. Information in parenthesis include information whether the different practices are relevant for development, planning, communication and management.

1. **<u>Acceptance testing (development practice)</u>**

Acceptance testing has some similarities with TFD (Test First development). Required acceptance tests should be planned for and developed as early as possible. TFD is a good practice and parts of the acceptance tests should be planned together with TFD planning. Associating a test with every piece of functionality is brilliant according to [9].
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 satisfies the safety tests required by e.g. EN 50128 or IEC 61508-3. In the survey [12] the practice "acceptance tests" is ranked as number 20. Automating the tests requires a lot of work.
**Safety adaptations**
Acceptance testing is of crucial importance for SCSW and several tests are required as part of the Verification and Validation plan. Regression testing is especially important when using incremental SW development. This topic should be improved in the next edition of both EN 50128 and IEC 61508-3. Some of the acceptance tests will be performed by the independent testers in the Alongside engineering team during a sprint. The final tests are run after the end of the last Sprint if e.g., coding have been performed as part of the last Sprint.

2. **<u>Backlog splitting and backlog refinement (planning activities)</u>**

Prioritized work list is closely linked to this practice and [12] has ranked backlog as number 2 of the practices.
**Safety adaptations**
Backlog splitting is an important part of SafeScrum [4]. 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. Backlog refinement is also known as story time and backlog grooming. The Product Backlog needs to be refined based on current knowledge. The Team,

the Scrum Master, the Product Owner and the RAMS (Reliability Availability, Maintainability and Safety) Manager should participate in the backlog refinement meeting. Several of the safety requirements are mainly taken care of by e.g. the RAMS team. If in doubt regarding safety requirements related to legislation or standards, the assessor should be consulted. The backlog refinement meetings will improve the understanding of the requirements and as a result ensure that requirements are implemented correctly. In most cases, the backlog refinement process will not require SRS changes. If there are reasons to believe that the SRS should be changed, a CIA (change impact analysis) should be initiated.

### 3. **Prioritized work list** (planning activities)

Prioritizing the backlog items may be performed by using Index cards or similar.

**Safety adaptations**

RAMS manager (a common role in railway signalling projects) or e.g., the Safety manager should be involved in the prioritization. This topic is an important part of SafeScrum [4].

### 4. **Daily Scrum meeting (DSM) including four questions** (communication activities)

Findings by [8] show that DSMs may not necessarily have to be held daily. In addition, the focus of the meetings should be on discussing and solving problems, and planning for the future, rather than reporting what has been done. This corresponds to the experience by one of the authors. VersionOne 2016 [12] has daily stand-up ranked as number 1.

**Safety adaptations**

Paasivaara et al. [10] examined agile practices in global software development and found that DSMs helped to reveal problems early, which is important when developing SCSW. Four questions was presented at ISSC in 2016 [18]. The questions are the three 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 related impact of the completed work

Adding question 4 to the list is important for organizations that develop both safety and non-safety products. It is relevant both for the work performed yesterday and the work to be performed today. If the answer to the last question – question 4 – is positive, we need some additional process activities. First, we need to close the daily stand up meeting. Those who have the necessary competence stay for the safety meeting to discuss and resolve the safety issues. If this proves difficult, we should involve a safety expert or, if this also fails, we should involve the assessor. Sometimes the answer can be "I do not know" or "I'm not sure". This should be followed up by e.g., the RAMS manager together with a person from the Sprint team.

There is no need to record any minutes of meeting, the value of the Daily Standup is to keep everybody informed and quickly highlight any problems. The Scrum Master is responsible for taking actions if there are any problems identified.

### 5. **Sprint planning meeting** (planning activities)

The sprint planning meetings are attended by the product owner, Scrum Master and the Scrum team. According to [7], sprint planning is a critical meeting, probably the most important event in Scrum. VersionOne 2016 [12] has iteration planning ranked as number 5.

**Safety adaptations**

Sprint planning, together with high-level plans are the main part of the planning activity [6].

### 6. **Timebox** (Development and management activities)

The sprints should have a fixed length ranging from two weeks to a month. Scrum meetings, which last about 10-15 minutes each day, are also formal.

Time-boxing is a brilliant practice according to [9]: "*Time-boxing every iteration – not accepting any delays, even if some functionality has not been implemented – is an excellent discipline, forcing team members and customer representatives to plan carefully and realistically, and bringing stability to the project*".

**Safety adaptations**

Time boxing is an important part of SafeScrum. It is important for the management to ensure that the work is time boxed and not to change the planned Sprint activities due to e.g., other projects.

7. **Incremental development including iteration, stepwise integration and short iterations** (development activities)

Development of software is normally incremental, e.g., product releases and maintenance releases. Research at The Standish Group indicates that shorter timeframes with delivery of software components early and often, will increase the success rate. Shorter timeframes result in an iterative process consisting of design, prototype, develop, test, and deploy of small elements [13].

According to [9] iteration is brilliant: *Short iterations are perhaps the most visible influence of the 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 2016 [12] has short iterations ranked as number 3.

**Safety adaptations**

A stepwise integration as part of sprints is an integrated part of the SafeScrum approach. Often there are a few iteration sprints before integration (increment) 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 must be involved. Incremental development is an important part of SafeScrum [4].

8. **Shippable code** (development activities)

This is also called "delivering working software" by [9]: *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.*

**Safety adaptations**

Several opponents to Agile development have claimed that this is difficult when developing SCSW, but our studies and experience have shown that this is possible (http://safescrum.no/). But the code is not "shippable as often as when developing traditional software.

9. **Sprint review** (communication activities)

At the end of each sprint, a sprint review meeting is held. VersionOne 2016 [12] has iteration reviews ranked as number 10.

**Safety adaptations**

Sprint review is an important part of SafeScrum. This means that at the end of each sprint, the team has produced a coded, tested and reviewable piece of software. The RAMS manager may be involved in the sprint review. According to [15] systematic (design) errors are introduced whenever there is a misalignment between the original intent of a requirement and its implementation. Potentially hazardous emergent behaviours may result from well-intended, but in hindsight, flawed design decisions made when addressing or satisfying requirements that, unfortunately, have unintended hazardous side effects. They can also be a result of implementation (process execution) errors during the software development process – e.g., modelling errors, coding errors, and tool-use errors. It is necessary to ensure that an assurance effort has been targeted at attempting to reveal both of these sources of errors.

10. **Retrospective** (communication activities)

This is performed at the end of one or more sprints to let the sprint team reflect on how they are doing and to find ways to improve. Copy from [7]: *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 2016 [12] has retrospectives ranked as number 4.

**Safety adaptations**

Retrospectives is an important part of SafeScrum. The RAMS or e.g., the safety manager may be involved in the retrospective.

**11. <u>Automated build</u>** <u>(development activities)</u>
In software development, build refers to the process that converts files and other assets under the developers' responsibility into a software product in its final or testable form.
**Safety adaptations**
As a part of a sprint stepwise integration is an integrated part of the SafeScrum approach. Often there are a few iteration sprints involving builds before an integration are performed into a real testable system.

**12. <u>Automated tests</u>** <u>(development activities)</u>
Working in Sprints does present some challenges - especially if your team is building a product from scratch. Sprint after sprint you have to add new features and safety related software. Having an automated testing framework, which takes care of both system and integration tests, is helpful.
**Safety adaptations**
Some of the tests should be the responsibility of the independent test team.

**13. <u>Burn down chart</u>** <u>(communication activities)</u>
The Sprint team should have a burn-down chart This is closely linked to monitoring the velocity. Some burn-down chart can be auto-generated.
**Safety adaptations**
One could preferably add special burn down chart for the safety requirements and for the Alongside engineering team.

**14. <u>Collective code ownership</u>** <u>(development activities)</u>
Collective code ownership is the explicit convention that all team members is not only allowed, but has a positive duty, to make changes to any code file when necessary. This is hype according to [9]: *The policy governing who is permitted to change various parts of the code is a delicate decision for each project. It depends on the nature of the team and many other considerations. It is pointless to describe a universal solution.*
**Safety adaptations**
It is certainly a delicate decision also when developing safety-critical software.

**15. <u>Continuous deployment</u>** <u>(development activities)</u>
Continuous deployment is an extension of continuous integration, aiming at minimizing lead time, the time elapsed between development writing one new line of code and this new code being used by users. This is an important part of DevOps.
**Safety adaptations**
A stepwise deployment after sprints is an integrated part of the SafeScrum approach. Often there are a few iterative sprints before a deployment. Continuous deployment have become more important due to security issues, and the familiar quotations "If it is not secure it is not safe" (unknown heritage). Security issues should be solved quickly. Chinese security researchers have hacked Tesla cars (turning on the brakes remotely) for the second year in a row (USA Today July 27, 2017). The Chinese researchers informed Tesla of their discovery. Tesla patched the vulnerabilities within two weeks.

**16. <u>40 hour week</u>** <u>(development activities)</u>
The idea behind the 40-hour week is that team members should only work the hours that they can sustain quality.
**Safety adaptations**
Since the idea behind the 40-hour week is that team members should work the hours that they can sustain quality, it is even more important when they also have to include safety.

**17. Open office space** (communication activities)

Open office space is a solution for managing a company's workplace using open space and not offices. Personal experience of the authors is that it is funny in the short run but certainly not in the long run.

**Safety adaptations**

Information and communication between the Sprint team and Alongside engineering team can be facilitated. But it is also important that the independent roles are not working too closely with the other teams.

**18. On-site customer** (communication activities)

Extreme programming (XP) wish to always have the customer available, not only to help the development team, but to be a part of it as well.

**Safety adaptations**

Information and communication between stakeholders like customer, ISA and e.g. NoBo is important.

**19. Information radiators** (communication activities)

This is about visualizing project information at relevant places. The main benefit of the practice is to promote responsibility among the team members. Another benefit is that information radiators tend to inspire conversation when outsiders visit, which can yield useful ideas.

**Safety adaptations**

One could evaluate to have information radiators for both the Sprint team and the Alongside engineering team. In addition, the project may benefit if the groups can see each other's information radiators.

**20. Refactoring** (development activities)

Refactoring could be considered as the agile answer to Big upfront design. Refactoring consists of improving the internal structure of an existing source code, while preserving its behavior.

**Safety adaptations**

It is also important to spend time on this practice when developing safety-critical software.

**21. Simple design** (development activities)

Simple design bases its software design strategy on the following principles:
- design is an ongoing activity, which includes refactoring and heuristics such as YAGNI ("You Aren't Gonna Need It"). Design quality is evaluated based on the rules of code simplicity and design elements such as "design patterns", plugin-based architectures or similar. Design decisions should be deferred until the "last responsible moment". It si important to collect as much information as possible on the benefits of the chosen option before taking the costs.

**Safety adaptations**

Safety-critical systems includes both functional and safety-critical requirements. That makes them complex. As a result, it is even more important to have simple design in mind. Regarding "last responsible moment", this should be discussed with the safety experts.

**22. Integration** (development activities)

Integration and the associated regression tests are an important factor in the success of modern software projects.

**Safety adaptations**

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 to create a testable system. Guidelines related to regression testing will be improved in the next edition of IEC 61508-3

**23. Quick design sessions** (development and communication activities)

Quick design sessions address the need for strategic decisions, while refactoring takes care of local design issues. If design decisions that have far-reaching consequences arises, two or more developers meet for a quick design session at e.g. the whiteboard.

**Safety adaptations**

Participants could be both from e.g., the Sprint team and the Alongside engineering team.

24. **Epic** (development activities)

An epic is a large user story that cannot be delivered as defined within a single iteration or is large enough that it can be split into smaller user stories.

**Safety adaptations**

An epic including safety aspects may help the software engineers to have a better understanding of larger parts of the safety system.

**25. Stories, user story, safety story** (development activities)

The term "user stories" were introduced by K. Beck in the XP-1999 conference and published in Computer [2]. A user story is a short, simple description of a feature, told from the problem owner's perspective.

**Safety adaptations**

Safety stories was introduced by [21] and further elaborated in [17].

The result of a safety analysis is usually either a requirement change, an architectural change or the introduction of a barrier – e.g., a watchdog. The safety story should refer to the analysis that introduced it. This is achieved by linking the safety story to the system, subsystem or function FMEA or FMEDA – e.g.:

**To keep** <function> **safe, the system must** <achieve or avoid something>.

**26. Team communication** (communication activities)

See Table 5 below, listing the different practices relevant due ensure good communication (listed at the bad communication row).

**Safety adaptations**

Meeting between e.g. the Sprint team and the Alongside engineering team.

**27. TFD** (development activities)

Test first development refers to programming activities in which three activities are tightly connected (1) coding, (2) testing (in the form of writing unit tests) and (3) design (in the form of e.g. refactoring). According to [9] TFD practice is brilliant as "*Associating a test with every piece of functionality*".

**Safety adaptations**

At least some of the tests have to be developed by independent testers.

We need a meeting between the Sprint team and the independent tester team to decide which test should be performed as part of the Sprint and which should be performed by the independent test team.

**28. Unit testing** (development activities)

This is defined in IEEE 24765:2010, section 3.3222. Unit test (1) testing of individual routines and modules by the developer or an independent tester. (2) a test of individual programs or modules in order to ensure that there are no analysis or programming errors. In [7], related to TDD (Test Driven Development): "*make sure any complex or business-critical code is covered by unit tests*" and related to code branching "*Be strict about keeping the mainline (trunk) in a consistent state. This means, at the very least, that everything should compile and all unit tests should pass*". The agile community may consider a unit test as a low/mid-level code-near test consisting of a set of assertions testing the interface of the code unit. Unit tests are typically managed by a unit test framework. Unit test only test low level code.

**Safety adaptations**

It is important to know that "unit testing" may have different meaning for a SW engineer and a safety expert. We have found that some of the assessors used the term unit as synonymous with "functional unit" while most software developers use the term for a "chunk of code" or to function or method.

**29. Usability testing** (development and communication activities)

Usability testing is not only an Agile practice, but has attracted much attention.

Usability testing is a long-established, empirical and exploratory technique to answer questions such as "how would an end user respond to our software under realistic conditions?"

**Safety adaptations**

A safety expert should preferably be involved in the evaluations of hos users behave related to the safety functions.

**30. <u>Velocity</u>** <u>(development activities)</u>

At the end of each iteration, the team adds up effort estimates associated with user stories that were completed during that iteration. This total is called velocity. According to [9] "*The notion of velocity and the associated artifact of task boards to provide visible, constantly updated evidence of progress or lack thereof are practical, directly useful techniques that can help every project*".

**Safety adaptations**

Delays in development when developing safety-critical software is very common so more focus on velocity could be helpful in some projects.

**31. <u>Version control</u>** <u>(development activities)</u>

Version control is more important when having an agile approach as it is expected that there will be more releases. IEEE 24765:2010 has defined version control as: establishment and maintenance of baselines and the identification and control of changes to baselines that make it possible to return to the previous baseline.

**Safety adaptations**

This topic is also strongly related to configuration management [5] and the chapter "Definition of system" in the Agile safety case [1].

## 4.4 Practices relevant for each of the three main challenges (requirement management, bad management and bad communication) when developing SCSW

The three main challenges and the corresponding 10 relevant practices (as a help to solve the challenges) are presented in the Table 5 below. We have found the main challenges by performing literature surveys (e.g. Chaos report [13]) and surveys for both the Aviation [14] and Railway domain [19].

**Table 5: Practices relevant for the three main challenges**

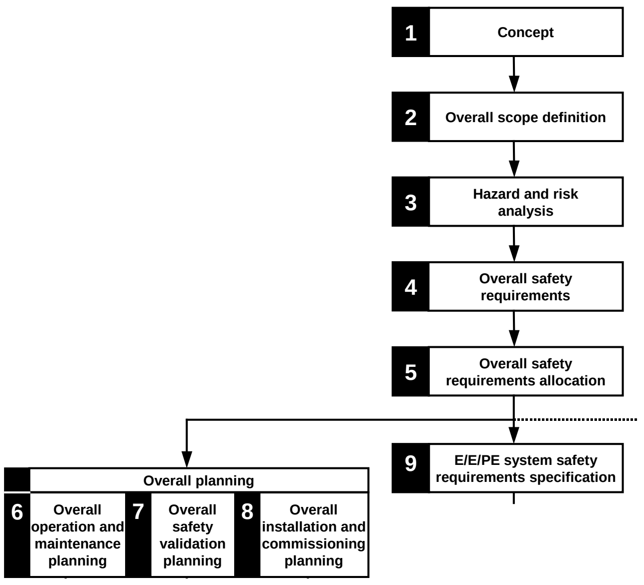
| Challenges | 10 relevant practices |
|---|---|
| 1. Requirement management<br>• Ambiguitos<br>• Frequent changes<br>• Insufficient<br>• Addition of new requirements<br>• Immature standards<br>• Different interpretation of standards | 1. Backlog<br>2. Backlog splitting<br>3. Backlog refinement meetings<br>4. Definition of done<br>5. Definition of ready<br>6. Hazard stories<br>7. Prioritize requirements/work list<br>8. Safety stories<br>9. Test driven requirement<br>10. User stories |
| 2. Bad management<br>• Lack of resources<br>• Lack of executive support<br>• Lack of planning<br>• Lack of IT management | 1. Backlog refinement meetings<br>2. Definition of done<br>3. Definition of ready<br>4. Frequnet releases<br>5. Iteration planning<br>6. Prioritize requirements/work list<br>7. Release planning<br>8. Sprint planning<br>9. Timebox<br>10. Test driven requirement |

| Challenges | 10 relevant practices |
|---|---|
| 3. Bad communication<br>• Incomplete requirements<br>• Lack of user involvement<br>• Did not need it any longer<br>• Unrealistic expectations | 1. Backlog refinement meetings<br>2. Daily scrum<br>3. Definition of done<br>4. Definition of ready<br>5. Prioritize requirements/work list<br>6. Retrospective<br>7. Sprint review/demo<br>8. Team-based estimation<br>9. Test driven requirement<br>10. The wall (similarities with scrum and Kanban boards) |

## 4.5 Practices suitable in the first safety phases

We have evaluated which practices that are relevant for the first safety phases as part of the work described above. There are several practices that are relevant. Upfront activities are also important when having an agile approach.

**Table 4: Practices relevant in the first safety phases**

| Before the first sprint | The first nine safety phases according to IEC 61508 |
|---|---|
| 1. Iteration planning<br>2. Release planning<br>3. Backlog<br>4. Backlog splitting<br>5. Backlog refinement<br>6. Coding standard<br>7. Definition of ready<br>8. Epic<br>9. Prioritized work list<br>10. Relative estimation<br>11. Story<br>12. Story mapping<br>13. User story<br>14. Safety story<br>15. Hazard story<br>16. Security story<br>17. Taskboard<br>18. Team-based estimation<br>19. Test driven requirement<br>20. The wall |  1 Concept<br>2 Overall scope definition<br>3 Hazard and risk analysis<br>4 Overall safety requirements<br>5 Overall safety requirements allocation<br>9 E/E/PE system safety requirements specification<br>Overall planning<br>6 Overall operation and maintenance planning<br>7 Overall safety validation planning<br>8 Overall installation and commissioning planning |

## 5. CONCLUSION

There exist more than 70 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 50 relevant practices and described necessary safety adaptions for 31 pratices to ensure that important international software standards like the generic IEC 61508-3, EN 26262-6 for automotive and EN 50128 for railway are satisfied. All of these 50 practices may contribute to shorten the time to market, to reduce costs, to improve quality and to enable more frequent releases. We have seen that several agile practices can also be used when having a traditional SW development approach and that many of the agile practices can be used as part of up-front engineering (before e.g. the first Sprint).

**References**

[1] T. Myklebust And T. Stålhane. The Agile Safety Case. Springer February 2018.

[2] K. Beck: Embracing change with Extreme programming, 1999 IEEE Computer

[3] 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

[4] T. Stålhane, T. Myklebust and G. Hanssen. The application of Safe Scrum to IEC 61508 certifiable software. PSAM11/ESREL 2012. Helsinki June 2012.

[5] T. Myklebust, T. Stålhane and N. Lyngby. Application of an Agile Development Prosess for EN 50128/Railway conformant software. Esrel 2015

[6] T. Myklebust, T. Stålhane and N. Lyngby. The Agile Safety Plan. PSAM13, 2016

[7] H. Kniberg. Scrum and XP from the trenches. C4Media. Second edition. 2015

[8] 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

[9] B. Meyer. Agile! The Good, the Hypa and the Ugly. Springer Ed.1, 2014

[10] 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.

[11] VersionOne – 10th State of Agile 2015

[12] VersionOne – 11th State of Agile 2016

[13] Chaos1995. The Standish Group

[14] G. K. Hanssen, G. Wedzinga and M. Stuip. An Assessment of Avionics Software Development Practice: Justifications for an Agile Development Process. XP 2017

[15] Kelly, XP2015:
http://dl.acm.org/citation.cfm?id=2894798&CFID=784458267&CFTOKEN=79745997
Project Smart 2014: www.projectsmart.co.uk/ last visited May 11, 2016

[16] Eveleens and Verhoef. IEEE Software 27.1 (Jan/Feb 2010): 30-36. IT Professional Facilitator. http://itprofessionalfacilitator.com/itpro/ last visited May 11, 2016

[17] T. Stålhane and T. Myklebust, XP 2018. Hazard stories, HazId and Safety stories in SafeScrum

[18] T. Myklebust, T. Stålhane and G. K. Hanssen. Use of Agile Practices when developing Safety-Critical software. ISSC 2016-08, Orlando.

[19] T. Myklebust, G. K. Hanssen and N. Lyngby. A survey of the software and safety case development practice in the railway signalling sector. ESREL Portoroz Slovenia 2017

[20] T. Myklebust and T. Stålhane. The Agile Safety Case and DevOps for Railway signalling system. 20th Nordic seminar on Railway technology, Gothenborg Sweden 2018

[21] T. Myklebust and T. Stålhane. Safety stories – A New Concept in Agile Development. SafeComp 2016-09, Trondheim.

[22] JOINT SOFTWARE SYSTEMS SAFETY ENGINEERING HANDBOOK . DEVELOPED BY THE JOINT SOFTWARE SYSTEMS SAFETY ENGINEERING WORKGROUP. Original published December 1999. Version 1.0 Published August 27, 2010