The application of Safe Scrum to IEC 61508 certifiable software

Tor Stålhane[a*], Thor Myklebust[b], Geir Hanssen[b]
[a]NTNU, Trondheim, Norway
[b]SINTEF ICT, Trondheim, Norway

Abstract: *In order to develop a better understanding of the application of Scrum to IEC61508 certifiable software, we assessed the standard to see how Scrum could conform. Each section of part 3 of the standard was assigned one of the categories: (1) "OK" – no modification needed to Scrum or IEC61508, (2) "?" – need to be discussed, (3) "Not OK" – need adaptation of Scrum*

Based on our assessment we proposed the Safe Scrum where the main idea is separation of concerns. Everything that is not part of the software development process is kept outside Scrum and will thus not be influenced by our choice of paradigm. In Safe Scrum, all requirements are split into safety critical requirements and other requirements and inserted into separate product backlogs.

We then did a new assessment where we took the Safe Scrum into consideration. We found 15 issues where we need changes in order to make the process acceptable to Scrum and the safety assessors: how to structure development, plan for validating safety, create, review, select, design and ensure safety, write requirements for module testing, and test and evaluate the outputs from the safety lifecycle

The first part of our model consists of the IEC61508 steps of developing the environment description and the SSRS phases 1-4. These initial steps result in the initial requirements of the system that is to be developed and is the key input to the second part of the model – the Safe Scrum process. Using an iterative and incremental approach means that the project can be continuously re-planned based on recent product experience. Between the iterations, experience can be used to re-prioritize the product backlogs. This makes the process flexible.

When the sprints are completed, a final RAMS validation will be done. Since most of the system has been incrementally validated during the sprints, we expect the final RAMS validation to be less extensive than when using other development paradigms. This will also help us to reduce the time and cost needed for certification.

Keywords: IEC 61508, Scrum, agile, safety-critical, software

1. INTRODUCTION

Agile development methods such as Scrum are used in more and more areas of software development. For safety critical systems, however, safety engineers feel that agile development does not fit, the main reason being that these projects require a strict plan which makes later changes costly. Still, safety critical projects suffer from many of the same problems that mare other software development projects, such as the need to change plans and requirements, being too late and having a solid budget overrun.

Our contribution to solving this problem is an assessment of how we can adapt Scrum so that it can be used without losing the benefits that we get from both of these two concepts. The assessment is performed by three experts in the areas software development, certification and agile development respectively.

The remainder of this paper is organized as follows: section 2 describes related work; section 3 discusses agile development both as-is and in a safe version – called Safe Scrum – and why adopting an agile approach might be a good idea. In addition we show how Safe Scrum allows us to separate the software process from the rest of the IEC 61508 process. Section 4 describes the two iterations we used to assess the main challenges when adapting agile development to IEC 61508 while section 5 describes the necessary adaptations. Section 6 discusses threats to validity of our conclusion, while section 7 ends the paper with a conclusion and a short description of possible further works.

2. RELATED WORK

There has been little experience published on the use of agile development both for use together with IEC 61508 and for safety critical software in general. A search on the internet show that the majority of hits when using the search terms “agile” or “Scrum” and “IEC 61508” or “safety critical” has led to hits on blogs, courses and discussion fora. Searches with only “Scrum” and “safety critical” gave only 25 hits and no papers at all. This is as expected since we deal with emerging technology. All that is published is assessment and analysis of how agile methodology will fit a certification scheme for the development of safety-critical software – little practical experience is published. In addition, all work done on IEC 61508 is related to versions published before 2010.

L.R. Thorsen [1] has studied the use of XP when developing safety critical software according to IEC 61508. The main challenges identified where in planning (three out of 12 XP practices) and design (three out of seven XP practices). Coding (seven XP practices) and testing (six XP practices) did not pose any problem.

M. Muller [8] has presented a paper on functional safety, automotive SPICE and agile methodology. His main points were that some agile methods, such as incremental development, rough overall release planning and the development and clarification of requirements over the development process has “proved their worth in automotive for years” while he will not recommend a pure agile approach.

Two students at Lund University [7] have done two case studies in Swedish industry on the clash between agile and plan-driven development. Even though their study did not focus explicitly on the development of safety critical software, their results are still relevant for our work. Their conclusion is that the main problems are planning, risk and documentation. Risk handling implies focus on control and prediction – two components that are not found in agile development. To quote the thesis: “...the strongest, most important factor to handle in order to fuse agile with plan-driven or adapt agile processes for safety-critical development seem to be about managing risk in some way”.

An article from S. Blair and R. Watt [9] discusses the architectural part of agile development – in particular the problems related to the Big Design Up Front versus an agile approach where the architecture is developed along with the software. Their main argument is that in the end all the design documents will be delivered: “By the end of the project, the auditors get the paperwork he or she needs to feel comfortable but the paperwork was developed progressively”. The authors ignore, however, that a large part of what “the auditors need in order to feel comfortable” stems from the process and not from the product or the accompanying documents.

Z.R. Stephenson, J.A. McDermid and A.G. Ward [5] have studied the application of agile software development in safety-critical health systems. Their main conclusions are that even though there are problems with introducing agile methods in this area, none of the identified issues will prevent the use of agile methods in safety-critical development. Their main issues are that we need to include the certifying authority into the process and that we need to develop a “... working, simulation and validation environment...”

A. Garg [6] has analysed all published agile practices in order to identify what need to be added if the method shall be applied in the safety-critical domain. He identified the following issues: (1) extension of agile to co-design of hardware and software, (2) reliability and safety impact analysis, (3) preliminary architecture design, (4) rules for refactoring, (5) configuration control ad (6) traceability.

X. Ge, R.F. Paige and J.A. McDermid [2] has published a work on iterative and agile development of safety critical software where they claim that there has not been a successful, documented deployment of existing agile methods on a real industrial safety-critical project. The process they propose “...precisely capture the notion of sufficient up-front design...” They also outline how to use safety patterns and a “modular notion of safety argument to enable the iterative development of safety argument...”

P. Gardner at the consultancy company ATENA [3] has produced a summary of issues that will rise when trying to apply agile methods to aviation (DO-178B) and rail (EN 50128) software. According to the authors,

the main problems identified are related to documents – e.g. architectural design, low-level design and test cases – and their contents plus traceability.

R. Morsicato and B. Shoemaker [10] have written a tutorial on agile development used in systems to be certified by the FDA. The authors focus on testing under the slogan "Why test only when we are done?" They claim that the most important aspect of agile development is the introduction of continuous testing which in turn will lead to that "validation becomes integral to development". In addition, they claim that agile development will improve hazard identification as they registered that in their work they discovered most hazards as the system evolved.

R.A. Chisholm [4] has done a study of the application of agile methods on software that will be certified according to the aviation standard DO-178B. He identified the following important issues: planning, requirements analysis, refactoring of code that is already validated and / or certified. On the positive side he focused on the continuous testing as an improvement over traditional software development.

Wils et al [19] have done a rather thorough assessment of agile principles for Barco - a major Belgian avionics equipment supplier. The authors look into how agile development will fit in with the avionics standard DO-178B. Two of their main conclusions are that an agile development process will add transparency and feedback to the project and enable the company to treat documents like source code and apply continuous integration, enabling shorter iterations.

None of the papers we have reviewed claim that agile development is incompatible with good practices for developing safety critical software. The most commonly mentioned problems are design – e.g. architectural design and low-level design – and planning plus rules for refactoring and traceability. On the positive side, the most commonly mentioned positive ideas are the continuous testing and flexibility. We should also note that several companies are already using agile development in safety critical systems, e.g. Avinor and Kugler Maag CIE [8] and several others are looking into it – e.g. ABB.

3. AGILE DEVELOPMENT

3.1 Separation of concerns

In order to be able to use an agile approach without overloading the concept with large amounts of "non-agile" activities, we have split the software development from the rest of the IEC 61508 development process as shown in the diagram below. Only the software development itself is handled by the Safe Scrum process - the rest is kept outside Scrum. Thus, the software developers develop software in Safe Scrum while high level planning, systems design and decisions concerning safety – e.g. new safety requirements – are done outside the Safe Scrum process. Note that changes of high-level plans may be fed back from the RAMS validation to the Safe Scrum after each sprint.

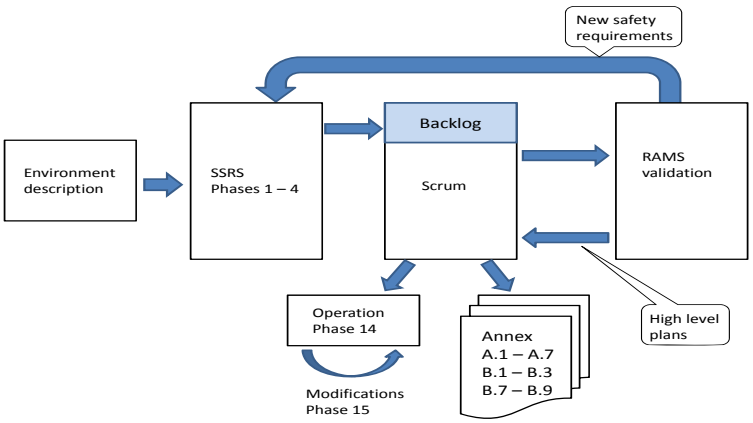

Figure 1: Scrum's role in safety critical software development

Being conformant to IEC 61508 will add a set of requirements on the Scrum software development process in that they either will have to use the techniques described in the standard's annexes or write explanations on why they use other techniques – e.g. explain why it is just as good as the ones recommended by the standard. Many companies that develop safety critical software have already defined a standard set of development activities based on the required SIL level and their chosen tools and methods. In this case, the Scrum team will use these tools and methods.

3.2 Agile development – as is

Agile software development is a way of organizing the development process, emphasizing direct and frequent communication, frequent deliveries of working software increments, short iterations, active customer engagement throughout the whole development life cycle and change responsiveness rather than change avoidance. This can be seen as a contrast to waterfall-like models, which emphasize thorough and detailed planning, and design upfront and consecutive plan conformance. Several agile methods are in use whereof *extreme programming* [16] and *Scrum* [17] are the most commonly used. Figure 2 explains the basic concepts of an agile development model.

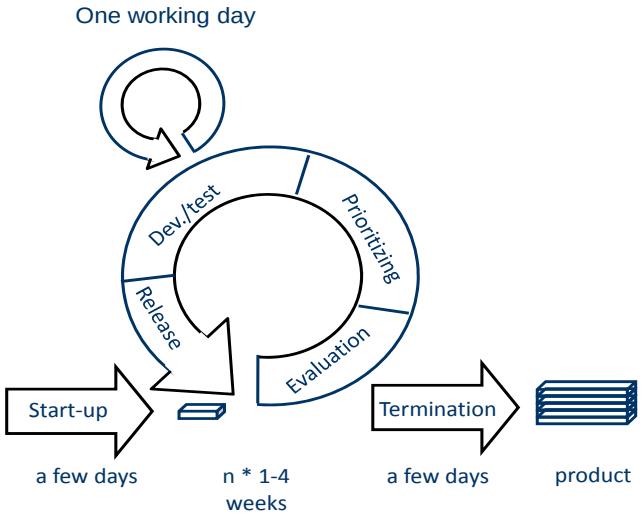

Figure 2: The basic agile software development model

The main constructs of this model are (based on Scrum):

- Initial planning is short and results in a prioritized list of requirements for the system called *the product backlog*. Developers also develop *estimates* per item.
- Development is organized as a series for *sprints* (iterations) that lasts a few weeks. Typically, developers will apply the principles of *test-driven development* [13] where automated tests are developed *before* the code.
- Each sprint starts with a *sprint planning meeting* where the top items from the product backlog is moved over to the *sprint backlog* – adding up to the amount of resources available for the period. These requirements will be implemented in the following sprint.
- Each working day starts with a *scrum***,** which is a short meeting where each member of the development team (1) explains what she/he did the previous work day, (2) any impediments or problems that need to be solved and (3) planned work for the work day.
- Each sprint *release* an *increment* which is a running or demonstrable part of the final system.
- The increment is *demonstrated* for the customer(s), which will decide which backlog items that have been resolved and which that need further work. Based on the results from the demonstration the next sprint is planned. The product backlog is revised by the customer and is potentially changed / reprioritized. This initiates the sprint-planning meeting for the next sprint.
- When all product backlog items are resolved and / or all available resources are spent the final product are released. Final tests can be run to ensure completeness.

We have observed that the main problem with introducing an agile development process in the development of safety critical systems is not to prove that it will work but that it is difficult to get the product certified. Our focus will thus be on certification and not on software development per se. You will also need to write a new set of procedures and train the developers in a new way of developing software but this problem will not be discussed here.

3.3 Safe Scrum

The proposed variant of Scrum, which we call "Safe Scrum", is motivated by the need to make it possible to use methods that are flexible with respect to planning, documentation and specification while still being acceptable to IEC 61508, as well as making Scrum a practically useful approach for developing safety critical systems. The rest of this section explains the components and concepts of this combined approach.

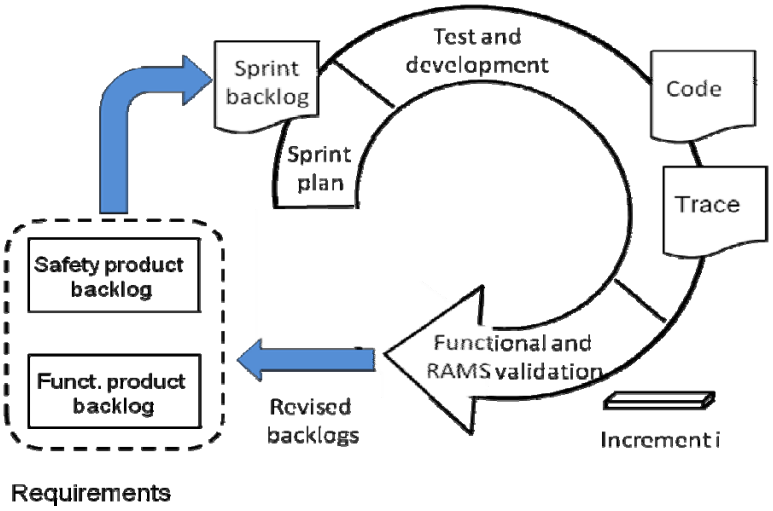

Figure 3: The Safe Scrum model

Our model has three main parts. The first part consists of the IEC 61508 steps needed for developing the environment description and then the SSRS phases 1-4 (concept, overall scope definitions, hazard and risk analysis and overall safety requirements) – see figure 1. These initial steps result in the initial requirements of the system that is to be developed and is the key input to the second part of the model, which is the Scrum process. The requirements are documented as *product backlogs.* A product backlog is a list of all functional and safety related system requirements, prioritized by the customer. We have observed that the safety requirements are quite stable, while the functional requirements can change considerably over time. Development with a high probability of changes to requirements will favour an agile approach.

Usually, each *backlog item* also indicates the estimated amount of resources needed to complete the item – for instance the number of developer work hours. These estimates can be developed using simple group-based techniques like 'planning poker', which is a popularized version of wideband-Delphi [12].

All risk and safety analyses on the system level are done outside the Safe Scrum process, including the analysis needed to decide the SIL level. Software is considered during the initial risk analysis and all later analysis – on per iteration. Just as for testing, safety analysis also improves when it is done iteratively and for small increments – see [10].

Due to the focus on safety requirements, we propose to use two product backlogs, one *functional product backlog*, which is typical for Scrum projects, and one *safety product backlog*, which is used to handle safety requirements. 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 core of the Scrum process is the repeated *iterations*, which are called sprints in the Scrum terminology. Each iteration is a mini waterfall project or a mini V-model, and consists of planning, development, testing, and verification. For the development of safety critical systems, traceability between system/code and backlog items, both functional requirements and safety requirements, is needed. The documentation and maintenance of trace information is introduced as a separate activity in each sprint – see fig. 3. In order to be performed in an efficient manner, traceability requires the use of a supporting tool. There exist several process-support tools that can manage this type of traceability in addition to many other process support functions. One out of many examples is Rally software - see rallydev.com.

An iteration starts with the selection of the top prioritized items from the product backlog. In the case of Safe Scrum, items in the functional product backlog may refer to items in the safety product backlog. The staffing of the *development 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. The selected items constitute the *sprint backlog*, which ideally should not be changed during the sprint. The development phase of the sprint is based on developers selecting items from the sprint backlog, and producing code to address the items.

An important practice in many Scrum projects is *test-driven development*, where the test of the code – usually some kind of unit-test [13] – is defined *before* the code itself is developed. Initial, this test is simple, but as the code grows, the test is extended to continuously cover the new code. The benefits of test-driven development are that the developer needs to consider the testing of the code before implementation, it enables regression testing, and it provides documentation of the code.

A sprint should always produce an *increment*, which is a piece of the final system, for example executable code. The sprint ends by demonstrating and validating the outcome to assess whether it meets the items in the sprint backlog. Some items may be found to be completed and can be checked out while others may need further refinement in a later sprint and goes back into the backlog. To make Scrum conform to IEC 61508, we propose that the final validation in each iteration is done both as a validation of the functional requirements and as a RAMS validation, to address specific safety issues. If appropriate, the independent safety validator may take part in this validation for each sprint. He should also take part in the retrospective after each sprint to help the team to keep safety consideration in focus. If we discover deviation from the relevant standards or confusions, the assessor should be involved as quickly as possible. Running such an iterative and incremental approach means that the development project can be continuously *re-planned* based on the most recent experience with the growing product. This principle is related to the well-known principle of the Deming/Shewhart cycle [18]. Between the iterations, it is the duty of the customer or product owner to use the most recent experience to re-prioritize the product backlogs.

In addition to the re-planning mentioned above, applying the RAMS validation process to each increment will also give risk and hazard analyses a gradually evolving scope. This will improve the quality of these analyses. Even if these increments cannot be installed at the customer's site, they can still be tested and run as part of a system simulation In addition, safety analysis performed on small increments can be more focused and thus give better results [20]

As the final step, when all the sprints are completed, a final RAMS validation will be done. Given that most of the developed system has been incrementally validated during the sprints, we expect the final RAMS validation to be less extensive than when using other development paradigms. This will also help us to reduce the time and cost needed for certification.

3.4 The benefits of Safe Scrum

To summarize the key benefits that comes from this combination of a safety-oriented approach and a process model for agile software development are that the process enables

- Continuous feedback both to the customer, the development team and the independent test team.
- Re-planning, based on the most recent understanding of the requirements and the system under development.
- Mapping of functional and safety requirements.
- Code-requirements traceability.

- Coordination of work and responsibilities between the three key roles; the development team, the customer and the assessor.
- Test-driven development of safety critical systems.

All of these points will help us to get a more visible process and thus better control over the development process, which again will help us to deliver on time and within budget.

4 IEC 61508 REQUIREMNTS ASSESSMENT

4.1 The IEC 61508 assessment – first iterations

The first iteration of the assessment process was done by going through each section of part 3 – Software Requirements - of the standard, asking "Will we fulfil this requirement if we use the Scrum process?" This was done by three experts: an expert on safety and software engineering, a certified safety assessor and an expert on software engineering and agile development.

As a result of this process, each section of part 3 of the standard – a total of 372, including bullet points – was assigned to one of the categories:
- "OK" – this section can be met without modification to Scrum or IEC 61508.
- "?" – this section needs to be discussed further
- "Not OK" – this section will require adaptation for IEC 61508 or Scrum.

After this first iteration we had 49 sections in the categories "?" and "Not OK". As a second part of the first iteration, we had an in-depth discussion and analysis of all sections in the "?" category. As a result of this, all of these sections were assigned to one of the categories "OK" or "Not OK".

4.2 The IEC 61508 assessment – second iteration

In the second iteration, with a more thorough analysis, we took the Safe Scrum model and the splitting of responsibilities shown on figures 1 and 3 into account. The main effect of this was to move several "Not OK"-activities out of the Scrum process. E.g. most of the planning work is done outside Scrum and only the low-level planning activities are retained inside the Scrum process. Many companies already have done such a split in that they have an overall plan for the whole project and a separate software development plan.

We ended up with 15 issues where we need adaptations and flexibility in order to make the process acceptable to both the Scrum team and to the safety assessors. These adaptations are as shown below:
- 7.1 – How to structure the development of the software. 2 out of 9 requirements
- 7.3 – How to develop a plan for validating the software safety. 2 out of 25 requirements
- 7.4.2 – How to create, review, select, design and ensure the safety of the system. 9 out of 50 requirements
- 7.4.7 – Requirements for software module testing. 1 out of 4 requirements
- 7.9 – How to test and evaluate the outputs from a software safety lifecycle. 1 out of 95 requirements

Thus, we have to deal with 15 IEC 61508 requirements where Scrum and the assessor have to show some flexibility.

5 ACHIEVED CONFORMANCE

The 15 issues identified in section 4.2 are mostly related to documentation and planning. We have the following four areas of concern:

Traceability – 1 item: Traceability is handled by having two Scrum backlogs – one for ordinary requirements and one for safety requirements plus a mapping between these two backlogs. In this way we will see which function that is used to implement which safety requirement. In addition, we have a separate activity in each development iteration that is used to develop and maintain traces.

Design – 4 items*:* There is a lot of difference between the agile purists' view and what is done in the real world. A case in point is the work done in the early phases of development. In a survey done by S.W. Ambler for the year 2009 with 280 respondents [15], he found that

- 79% of all agile projects do high-level initial requirements modelling
- 88% of all agile projects do some sort of initial modelling or have initial models supplied to them
- 70% of all agile projects do high-level initial architecture modelling
- 86% of all agile projects do some sort of initial modelling or have initial models supplied to them

There is no problem doing high level design at the start of an agile project and most agile projects do requirements and architecture modelling up front. In addition, there is nothing in an agile methodology that will prevent us from starting an agile, safety critical project with a solid high level architecture and a good understanding of the main requirements.

Planning – 3 items*:* The plans we needed to consider were the detailed development plan and the verification and validation plans. The detailed development plan will consist of a high level plan for the Scrum process and a detailed plan for each sprint. This is taken care of as shown in fig. 1.

We will also need a verification and validation plan adapted to incremental development. Thus, instead of *one* verification and validation plan, we will need a sequence and the plans for the early stages of development will need to include development of e.g. mock-ups and simulations.

Documentation and proof of conformance – 7 items: The main question here is what an assessor will accept as sufficient documentation, i.e. as proof of conformance. Will e.g. a print-out of a backlog item be accepted as documentation for a requirement? We have two areas of concern – the documentation of the final system and documentation for proof of conformance. All documentation needs that are not related to code development should be moved outside Scrum.

We need a separate documentation team which works in close connection with the Scrum team and participate in each sprint review and sprint planning meeting. From the developers, code with comments or using Javadoc should be accepted as documentation.

When it comes to proof of conformance we know from our work with ISO 9001 and Scrum [11] that ISO 9001 auditors are willing to accept white-board snapshots of the discussions plus a list of participants as proof of conformance. At least some IEC 61508 assessors have confirmed that they will do the same.

For proof of conformance for e.g. running a test suite, the following information could be inserted into a formal document: a snapshot of the whiteboard during test planning – what did we want to achieve, a print out of the test cases or test scripts – how will we achieve it, and a printout of the test log or test results – what have we achieved. This should be accepted as proof of conformance for a testing session.

6 THREATS TO VALIDITY

There are three threats to validity for our conclusion on achieving conformance in section 5: have we understood IEC 61508, have we touched all relevant IEC 61508 part 3 items and have we understood agile development in general and Scrum in particular?

6.1 Have we understood IEC 61508

One of the authors is a certified IEC 61508 assessor who has a long experience in certifying safety critical SIL 4 systems for rail signalling and SIL 2 off-shore products. One of the other authors has been working with introducing IEC 61508 both in an off-shore project and in a NordTest guideline [14]. He is also working with IEC 61508 in the area of industrial automation in the CESAR project. Thus, we have a good knowledge of the interpretation and practical use of IEC 61508 in several domains.

6.2 Have we touched all relevant items

By using the process described in section 4, we have systematically gone through the whole of IEC 61508, part 3. We are confident that all relevant IEC 61508 items are identified and assessed.

6.3 Have we understood agile development

Agile development, including Scrum, is not an exactly defined methodology and there exist a handful of agile methods [11]. Yet they are all based on the few common principles described in section 3.2. We have applied these common and fundamental principles to reduce potential bias from our own interpretations of what agile development is, yet a certain level of subjective interpretation is probably inevitable.

One of the authors has a long experience in using and teaching Scrum in industry. Thus, the necessary Scrum expertise has been available.

6.4 Our claims to validity

Based on the discussion in the sections 6.1 to 6.3 we claim that our conclusions regarding IEC 61508 part 3 and agile development in general and Scrum in particular will be valid for a wide range of companies and certifying bodies.

7 CONCLUSIONS AND FURTHER WORK

Elements of the agile development process is already used or considered for use in several important industrial domains such as automotive and air traffic management (used) and avionics and industrial automation (planned or valuated for use).

In our opinion, the solution suggested in chapter 3.3 – Safe Scrum – is better than e.g. the process used by Kugler Maag CIE [8] since we have kept more of the agile process concepts. This is important in order to reap the maximum benefits from using Scrum.

When the issues identified in section 4 and 5 are settled, it should be straight forward to use Scrum and still be IEC 61508 conformant. It is now important to get one or more companies to try it out in cooperation with the safety assessors to get a reality check of the concepts discussed above. This will allow us to identify possible problems and to make the adjustments necessary for industrial application.

With a little flexibility there are, in our opinion, no large obstacles for using agile development for safety-critical software. We should also look at the possibility to make agile development possible in areas such as ISO 26262 for automotive software and IEC / EN 62061 for control of machinery.

The two main challenges are the standards' requirements on detailed planning and requirements for proof of conformance. The Safe Scrum development process will be an important input to this work. This will lead to a process that is better adapted to handle changes that occur in any software development process and give us an incremental process – development, testing and verification – that again will lead to more efficient software development.

Acknowledgement

We gratefully acknowledge valuable input from Mika Katara, Technical University of Tampere, Olawanda Daramola, NTNU and Jan Endresen and Tormod Wien, ABB.

References

[1] L.R. Thorsen: Extreme Programming in safety-related systems. Master thesis, NTNU, Trondheim, Norway, June 17, 2002.
[2] X. Ge, et al. An Iterative Approach for development of Safety-Critical Software and Safety Arguments. Agile Conference 2010.
[3] P. Gardner: Agile Methods and Safety-Critical Software. Are they compatible? SILVER ATENA presentation
[4] R.A. Chisholm: Agile Software Development Methods and DO-178B Certification. Master thesis, Royal Military College of Canada, July, 2007

[5] Z.R. Stephenson et al.: Health modelling for agility in safety-critical systems development, The 1st Institution of Engineering and Technology International Conference on Systems Safety, 2006.
[6] A. Garg: Agile Software Development. DRDO Science Spectrum, March 2009
[7] B. Jacobsen and M. Norrgren: Agile vs. Plan-driven in safety critical development cases. A clash of principles? Master thesis, Lund University, January, 2008
[8] M. Muller: Functional Safety, Automotive SPICE and Agile Methodology. 8th Automotive Software Workshop, Italy, February 17, 2011
[9] S. Blair and R Watt: Responsibility-Driven Architecture. IEEE Software, March / April 2010.
[10] R. Morsicano and B. Shoemaker: Tutorial: Agile Methods in Regulated and Safety-Critical Environments. ShoeBar Associates
[11] T. Stålhane and G.K. Hanssen: The application of ISO 9001 to agile software development. PROFES, 2008
[12] B. Boehm: Software Engineering Economics. Prentice-Hall Inc., Englewood Cliffs, New Jersey, 1981, ISBN 0-13-822122-7
[13] L. Koskella: Test Driven, Manning, Greenwich, UK, 2008, ISBN 1-932394-85-0
[14] J. Herard et al. Guideline for the Validation of Functional Safety according to IEC 61508, NordTest Technical report 459, September 2000.
[15] S. Ambler, M. Vizdo.: August 2009 Agile Project Initiation Survey, www.agilemodelling.com/surveys/
[16] K. Beck and C. Andres: Extreme programming explained: embrace change, 2nd Edition. 2004, Boston: Addison-Wesley Professional.
[17] Schwaber, K., Beedle, M.: Agile Software Development with Scrum. 2001, New Jersey: Prentice Hall.
[18] Deming, W.E., “Out of the Crisis”. 2000, Cambridge: The MIT Press
[19] Wils, A. et al.: Agility in the avionics software world, XP'2006, 17-22 June 2006, Oulu, Finland
[20] Vuori, M.: Agile development of safety-critical software – while meeting standards’ requirements., Tampere University of technology, Finland