Safety standards and Scrum – A synopsis of three standards

Tor Stålhane – IDI / NTNU, Thor Myklebust and Geir Kjetil Hanssen – SINTEF ICT

1. Introduction

There are a large number of standards used to develop safety critical software. They all have one thing in common – being compliant with the standard, the software will be safe for its intended use. One common trait for all these standards, however, is that they are developed for plan driven, waterfall or V-model development processes. This organization of software projects has one large weakness – they are not god at handling changes to requirements and customer needs. On the other hand, Scrum and other agile development models are not well adapted to the development of safety critical software. For this reason we have started to study possible adaptations to Scrum so that it can be used in the development of safety critical systems.

If we want this adaptation to be of any interest, it is important to consider a wide set of application domains. For this reason, we have chosen three standards for our work – IEC 61508 [1], which is used a lot in the area of factory automation and which is also the starting point of other important standards for the development of safety critical software, e.g., ISO 26262 [2] for the automotive domain, IEC 60880 [3], which is the main standard for development of software of category A in nuclear control systems and EN 50128 [4] which is the main standard for software development in the railway domain.

We have analysed the three above-mentioned standards to identify where the stumbling blocks are if we want to both be compliant to the standard and to use Scrum [5] – or in our case, SafeScrum. By doing this for the three chosen standards, we hope to achieve three things (1) to understand the general questions and issues that may prevent the use of agile development for safety critical software, (2) to be able to extract some general rules which can help people to adapt standards to agile development and (3) by showing where the problems are, to start a process to update the relevant standards to allow agile software development.

2. SafeScrum – a short intro

We have observed that the safety requirements are quite stable, while the functional requirements can change considerably over time. The most important sources of changes for safety requirements are changes in relevant standards, which happen only seldom, and the discovery of new hazards during RAMS (Reliability, Availability, Maintenance and Safety) validation. This is taken care of in Safe Scrum with the possibility for revising the backlog after RAMS validation – see figure 1 below.

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 [6]. See also J. Gremming's article [14].

All the risk and safety analyses on the system level are done outside the Safe Scrum process, including the analysis needed to decide the safety level. Software is considered during the initial risk analysis and all the later analysis – on per iteration. Just as for testing, safety analysis also improves when it is done iteratively and for small increments – see [7].

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.

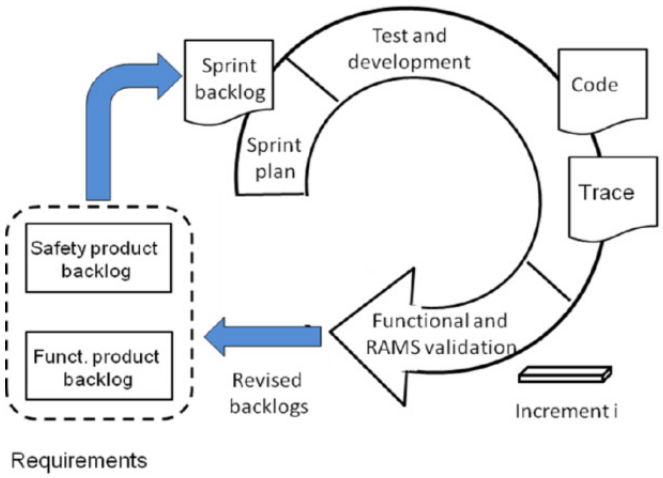

Figure 1: Safe Scrum process

3. The selected standards

3.1 IEC 61508, part 3

We are only dealing with IEC 61508 – part 3. How to decide the risk level and necessary SIL level are decided by other parts of IEC 61508 and is not a part of the software development process. In practice, the standard is mainly used for SIL2, but certification of SIL3 products is expected to increase in the near future.

For each SIL level, there is a set of recommended or highly recommended methods or techniques that must be used in order to stay compliant to the standard. The standard has defined a development process which should be instantiated based on the required SIL level.

Chapter 1.1 assumes that the safety requirements are already defined before we start working on the functional software requirements. However, section 1.1 –part d has an important note:

"Specifying the software safety functions and software Systematic Capability is an iterative procedure".

Just as most other safety standards, IEC 61508 uses the V-model for software development. However, the development organization is given a large degree of freedom. To quote sections 7.1.2.1 and 7.1.22:

" A safety lifecycle for the development of software shall be selected and specified during safety planning in accordance with IEC 61508-1, 6", and further "Any software lifecycle model may be used provided all the objectives and requirements of this clause are met."

3.2 IEC 60880

As the standard IEC 60880 only applies to level A systems, it has no specific requirements related to risk assessment and identification. Just as IEC 61508 it has a table of recommended practices for this level – see appendix B in the standard. IEC 60880 also explicitly allows iterative development, which brings us a step closer to agile development. A. Cockburn defines iterative development as follows:

"By iterative development, I specifically wish to mean a rework scheduling strategy in which time is set aside to revise and improve parts of the system. It does not presuppose incremental development, but works very well with it. …the difference is that an increment may actually ship, whereas an iteration is examined for modification." Iterative development helps you improve your product. Each time around the process you get to change and improve the product itself (and maybe some of your work habits).

It should be noted that IEC 60880 says nothing about important activities like system requirements specification and (overall) system specification but is only concerned with software development, thus including software requirements specification. In addition it handles the software aspects of integration, validation, installation and modification.

When it comes to development method, the standard states that

"The approach to software development should be based on the traditional "V" model as this approach has been reflected and promulgated in other standards …, but allowing necessary adjustments recognizing that some phases of the development can be done automatically by tools and that software development may be iterative".

3.3 EN 50128

This standard is, in practice, mainly used for SIL4 products and systems. Just as IEC 61508 and IEC 60880, EN 50128 explicitly allows iterative development, which brings us a step closer to agile development.

5.3.2.2 The lifecycle model shall take into account the possibility of iterations in and between phases.

Just as most other safety standards, EN 50128 uses the V-model for software development. However, the development organization is given a large degree of freedom. To quote the introduction:

"This European Standard does not mandate the use of a particular software development lifecycle. However, illustrative lifecycle and documentation sets are given...."

4. The adaptation approach

4.1 Method

We present a method for Scrum adopted to address the needs of development of safety system. This method was used in [8] and [9], it is simple and our experience so far is that it is highly efficient.

1. Collect a team containing a software expert, a domain expert and an assessor for the standard(s) under consideration
2. Identify all requirements in the standard related to software development
3. Go through all the requirements, asking the question "Will this requirement be fulfilled if we use Scrum?" This delegates each requirement to one of the following categories:
    a. Is fulfilled also if we use Scrum as is
    b. Is partly fulfilled if we use Scrum as is. Will need adding extra activities to the Scrum process
    c. Cannot be met if we use Scrum as is.
4. To sort out the problems we need to explore two options:
    a. Changes to Scrum – e.g.
        i. add-on to cater to the traceability requirement
        ii. interface to FAT
    b. Alternative interpretations of requirements in the applicable standards – e.g. what should be accepted as proof of compliance for an activity?

This approach leaves us with two challenges – (1) have we identified all relevant requirements and (2) different assessors have different opinions of what should count as proof of compliance. Especially challenge (2) is problematic since it has no final solution. One possible way out of this is to involve the assessor from day one and ask questions such as "If we use approach X here, will this be accepted?"

4.2 Results

4.2.1 IEC 61508

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

Necessary changes and adaptations:

- Traceability is achieved 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.
- We can have a separate documentation team which works in close connection with the Scrum team and participate in each sprint review and sprint planning meeting. We also need to clarify some problems, e.g. will
  - a print-out of a backlog be good enough documentation for a requirement?
  - code with comments or Javadoc be accepted as documentation?
- The safety validation plan is moved outside the Scrum process.
- High-level design is done before the Scrum starts and will result in a design document. Adjustments can be done as part of the preparation for each sprint and must be documented – e.g. by a white-board snapshot.
- An important question is what the assessor will accept as evidence, e.g. as documentation of decisions and decision acceptance. Such decisions can be documented in meeting minutes.
- The requirements for software testing can be met by using TDD (Test Driven Development) and documentation of the results. Tools like FitNess will be useful.
- System design is done before the Scrum process starts. During development in the Scrum process all modules and their interfaces are defined before coding starts.

The work on IEC 61508 resulted in the original SafeScrum model, shown in the diagram below.

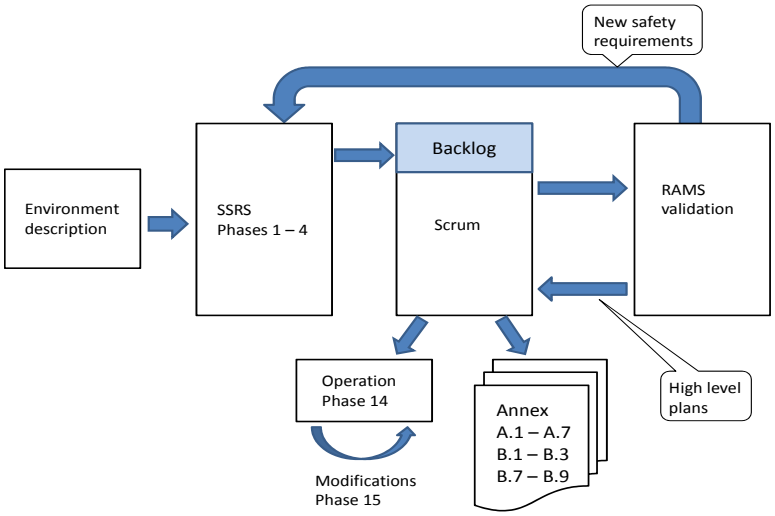

Figure 2: SafeScrum and IEC 61508

4.2.2 IEC 60880

According to Lahtinen et al [15], IEC 60880 corresponds to SIL 3 in IEC 61508. In addition, "IEC 60880 often has an emphasis on the required matters, or tasks, whereas IEC 61508 discusses the methods that can be used to meet these requirements."

We ended up with 6 issues where we need adaptations and flexibility in order to make the process acceptable. Note that the standard does not define the term "module". However, based on the term's usage in the standard, it is reasonable to interpret the term as synonymous with "functional module"

- 5.4 – Software project management
  2 out of 11 requirements
- 6.1 – Specification of software requirements.
  2 out of 15 requirements
- 7.1.2 – Implementation of new software in general-purpose languages
  1 out of 5 requirements
- 8.2.3 – Implementation verification
  1 out of 12 requirements

Necessary changes and adaptations:

- For the test descriptions, we need to add test specification, test report and code verification report
- We will consider the end of a sprint as the end of one or more phases. The sprint review will fulfil the requirement that each phase shall be systematically terminated by a review
- All requirements must be rigorously defined when they are (1) Inserted into the backlog, (2) Taken out of the sprint backlog and (3) Revised and reinserted into the backlog
- A formal description of the test inputs and results (test protocol) should be produced." This is an extension of the common way of doing testing in Scrum and we thus need to insert this into the develop-and-test part of the development process.
- As a minimum, the tests listed in appendix E, table E.4.2 and marked with "always" should be included for each module

4.2.3 EN 50128

We ended up with 20 issues where we need adaptations and flexibility in order to make the process acceptable. The term phase, as used in this standard, is complicated. In the general case, each sprint will contain several phase but for the concrete requirements in this standard we assume that a "phase" is the same as a "sprint. For the term "stage" we will use the definition from ISO 12207 [10]: "Period within the life cycle of an entity that relates to the state of its description or realization". Thus, a sprint can be considered as a stage. All work related to defining a QA plan is moved outside SafeScrum.

- 5.1.2. Organisation, roles and responsibilities – Requirements
  3 out of 14 requirements

- 5.3.2. Lifecycle Issues and documentation – requirements
  2 out of 14 requirements.
- 6.1.4. (Test) Requirements
  1 out of 5 requirements.
- 6.2. Software verification.
  4 out of 13 requirements.
- 6.5 Software quality assurance
  3 out of 18 issues requirements.
- 7.1. Lifecycle and documentation for generic software
  2 out of 3 requirements.
- 7.2 Software requirements
  3 out of 26 requirements.
- 7.4 Component design.
  1 out of 16 requirements.
- 7.5 Component implementation and testing
  1 out of 9 requirements.

Necessary changes and adaptations:

- A project needs a document template. This template can be reused for all projects. Each planning document shall have a paragraph specifying details about its own updating throughout the project: frequency, responsibility and method.
- All issues pertaining to verification are solved through project organization and by extending the SafeScrum model with a verifier process.
- To the extent require by the software safety integrity level, the documents listed in Table A-1 shall be produced for generic software.
- There is not much focus on test documentation in Scrum. In order to be compliant with EN 50128, we will change the way tests are described in Scrum. This is done by using a test tool that contains a both description template and a report template – e.g. Cumber [11] or FitNess [12]. We also need a test report template and adaption for a test tool printout – e.g. SCRIPT [13]
- All problems related to system's requirements testing are solved through including relevant tests in a FAT in the extended SafeScrum model
- New activity at the start of each sprint
  - The Software Quality Assurance Plan, Software Verification Plan, Software Validation Plan and Software Configuration Management Plan shall be drawn up at the start of the project and maintained throughout the software development life cycle.
  - All activities to be performed during a phase shall be defined and planned prior to the commencement of the phase

The sequence of deliverable documents as they are described in the standard's Table A.1 reflects an ideal linear waterfall model. This model is however not intended to be a reference in the sense of schedule and linkage of activities, as it would usually be difficult to achieve a strict compliance in practice.

Our work on EN 50128 resulted in several extensions to the SafeScrum model. The most important changes were inclusion of a verifier role and showing how Scrum development cooperated with the personnel responsible for the FAT

Figure 3: SafeScrum and EN 50128

Original Scrum does not include a verifier role. It is, however, not difficult to include in the SafeScrum model. According to EN 50128, the verifier can be the same person as the validator, given that the assessor agrees. On the other hand, using a person from outside the project will help to disseminate project knowledge and thus lower the project risk.

5. Synopsis

5.1 General points

Scrum and other agile development processes require competent personnel. Competence is more important for agile development than when using a waterfall process. The reason for this is that agile development gives the developers more freedom and thus more responsibility.

The most important ideas are separation of concerns and close cooperation with the assessor:

Separation of concerns:
Software developers are not necessarily good at eliciting requirements from customers, making safety plans or doing hazard identification and risk assessment and mitigation. It is not their domain of expertise. Thus, when using Scrum for the development of safety critical software we need to let Scrum do what Scrum is good at – enable highly qualified developers to develop high quality software that works according to the specifications and the customers' requirements. The rest of the process specified by the standard is performed in the same way as when we use any other model. This decision is the raison d'être for our concept of separation of concerns.

Assessor cooperation:
As we have already mentioned, the problem of what is accepted as PoC is strongly related to what the assessor needs. In addition, we might consider the needs of those who build the safety case and other activities needed for certification, marketing and sale.

In order to do this in an efficient way, the assessor should be involved from the first day of the project. This implies that if we are in doubt of what the assessor will accept as PoC for a certain activity, we should ask him. This will have two important effects: (1) we will create all the documentation that the assessor needs and (2) we will not generate documentation that is not needed. The requirements for documents in category (1) will be inserted into the backlog, just as any other requirement in a Scrum process.

This approach must be used with care so that we do not hold the assessor hostage to our choice of development process. If we need to ensure assessor independency, we can use one assessor as a "sparring partner" during the project and another one – preferably from the same organization – for certification.

5.2 Changes and adaptations

Once we have decided to use the strategy separation of concerns and close interaction with the assessor, we need to decide what is inside SafeScrum, what is outside and how these two parts will communicate in the most efficient way. This leads to the changes and adaptations from sections 4.2.1 – 4.2.3 which are collected and shown in table 1 below.

Table 1: Changes and adaptations

Issue for change and adaptation	IEC 61508-3 (ref. ch. 4.2.1)	IEC 60880 (ref. ch. 4.2.2)	EN 50128 (ref. ch. 4.2.3)
Development process	How to structure the development of the software	Software project management	Lifecycle issues and documentation – Requirements
			Lifecycle and documentation for generic software
			Organisation, roles and responsibilities – Requirements
Software validation	How to develop a plan for validating the software safety	Implementation verification	(Test) Requirements
	Requirements for software module testing		Component testing
	How to test and evaluate the outputs from a software safety lifecycle		
Software requirements	-	Specification of software requirements.	Software requirements
Software	-	-	Software verification

verification			
Software quality assurance	-	-	Software quality assurance
Software implementation		Implementation of new software in general-purpose languages	Component design.
			Component implementation
System safety	How to create, review, select, design and ensure the safety of the system	-	-
Number of issues	5	4	10

Even though IEC 61508 and IEC 60880 have almost the same number of issues, the areas they cover are quite different. Lahtinen et al have compared the two standards [15, 16] and concludes that:

"IEC 61508 coverage is greater than IEC 60880 coverage in the areas of requirements specification, verification, traceability and independent assessment. IEC 60880 requirements cover the areas of design, tool selection, pre-developed software, and software diversity more rigorously."

As should be expected, better coverage by the standard causes more issues to be raised in the "Scrumyfication" process.

EN 50128 requires twice as many adaptations to Scrum as the two other standards. To some degree, the reason for this difference is the formulations used in the standard. E.g. "Verification" is also mentioned in IEC 60880 and in IEC 61508 but due to the standard's organization and use of the term, it was in both cases considered to be outside Scrum. The same holds for software quality assurance. To quote IEC 61508, 7.1.2.6: "Quality and safety assurance procedures shall be integrated into safety life cycle activities". This means that it can be almost anywhere in the process.

Another problem is the term "component". The definition presented in EN 50128 does not help much either, since a component is "a constituent part of software which has well-defined interfaces and behaviour with respect to the software architecture and design and fulfils the following criteria:

- it is designed according to "Components" (see Table A.20);
- it covers a specific subset of software requirements;
- it is clearly identified and
    - has an independent version inside the configuration management system or is
    - a part of a collection of components (e. g. subsystems) which have an independent version"

The example above is just one of many identified during the Scrum adaptation process. It is important to get both the Scrum community and those responsible for the standards to agree on the definitions of the terms used.

All the standards raise the same important issues – how shall the development process be organized when the software development is Scrumyfied. More specifically:

- How are the standards' requirements for validation be reconciled with Scrum's ideas of testing and important concepts such as TDD
- How can we make sure requirements, component design, verification and quality assurance are taken care of
- "What is accepted as proof of compliance?" For several documents the only raison d'être is that they are needed to prove that a certain activity has been used or a certain method has been applied.

6. Conclusions and further work

6.1 Main conclusions from the synopsis

We have seen that the SafeScrum model, originally constructed for IEC 61508-3, is robust in the sense that it can easily be adapted to new standards without having to undergo major changes. Our main conclusion from [8] stands – the most important thing to do is to involve the assessor in the project from day one.

Customers play an important role in all agile development – also in SafeScrum. However, in many cases software is developed not for a customer but for a market segment. In this case, the product manger should take the customer's role. It is probably smart to also invite a group of interested customers or end-users into the process.

The terms used are important and there is no commonly accepted source of definitions. Terms like phase, component, activity, stage and software life cycle are used indiscriminately. It will thus always be difficult to assess whether a development project is conformant to requirements like EN 50128: "5.3.2.5: All activities to be performed during a phase shall be defined and planned prior to the commencement of the phase." Similar problems are found in all the three standards we have studied. Different interpretations of a term will often mean different roles in SafeScrum and due to separation of concerns it can be inside or outside the Scrum process.

A problem that seems to keep coming back to haunt us again and again is the requirements for a PoC. Even if the PoC is primarily defined for the assessors, other parties may also involved – those who build the safety cases, who will need the same documentation as evidence and, in some extreme cases, experts in the court of law where the issue will be "Has the developers really done everything that was within their power to make the systems safe, given the available resources?"

6.2 Where do we go from here?

So far, our ideas of how to adapt and apply Scrum in the development of safety critical systems are based on expert judgements. To move on and to see if these ideas are realistic we need to pilot and adapt the SafeScrum approach in close collaboration with industry *and* assessors. Presently we are planning such a project in collaboration with two Norwegian actors which will apply SafeScrum in upcoming development projects. We are also open to other partnerships in relation with this work.

We see several challenges in the work ahead. Firstly, in order to gain industrial experience we need to apply SafeScrum in real projects meaning that we have to introduce uncertainty and an overhead by altering already working development processes. Secondly, we need to ensure the independence of the assessor although part of the idea is to have the assessor work closer and more frequently with the development organization. Thirdly, accepting instability and flexibility in requirements for safety critical systems may create a tension in an industry, which seeks control by extensive planning, and conformance to plans. However, to move on and test and improve our ideas we need to put them into use in pilot projects.

In the long term, standards should move from prescriptive to goal based. This process is already well under way in the Norwegian oil and gas industry. By focusing on the goals, we will achieve two important things:

- Stop the growth in size of the standards. The IEC 61508, part 3 has grown from 52 pages in 1998 to 118 pages in 2010.
- Make it easier to introduce new tools and methods. If we fail here, we run the risk of having standards that are getting more and more out of step with realities.

References

[1] IEC: IEC 61508-3, Functional safety of electrical/electronic/programmable electronic safety-related systems. Part 3: Software requirements, 2010-04

[2] ISO: ISO 26262 – Road vehicles - Functional safety – Part 6: Product development: software level, 2008 May 30.

[3] IEC: IEC 60880 Nuclear power plants – Instrumentation and control systems important to safety – Software aspects for computer-based systems performing category A functions. Second edition, 2006

[4] EN: EN 50128 Railway applications - Communication, signalling and processing systems. Software for railway control and protection systems, June 2011

[5] Schwaber, K. and Beedle, M. “Agile Software Development with Scrum”, 2001, Prentice Hall

[6] B. Boehm: Software Engineering Economics. Prentice-Hall Inc., Englewood Cliffs, New Jersey, 1981, ISBN 0-13-822122-7

[7] R. Morsicano and B. Shoemaker: Tutorial: Agile Methods in Regulated and Safety-Critical Environments. ShoeBar Associates

[8] T. Stålhane, T. Myklebust and G.K. Hanssen: The application of Safe Scrum to IEC 61508 certifiable software, ESREL 2012, Helsinki, Finland

[9] T. Stålhane, V. Katta and T. Myklebust: Scrum and IEC 60880, Enlarged Halden Reactor Project meeting, Storefjell, Norway.

[10] ISO: ISO 12207 Systems and software engineering – Software life cycle processes, 2008

[11] Cucumber: visited April 16, 2013. http://cukes.info/

[12] FitNess: visited April 16, 2013. http://realsearchgroup.org/SEMaterials/tutorials/fit/

[13] SCRIPT: visited April 16, 2013. http://www.tucows.com/preview/207136/Script-Tool-Box

[14] J. Grenning: Planning Poker or How to avoid analysis paralysis while release planning April 2002

[15] J. Lahtinen et al.: "Comparison between IEC 60880 and IEC 61508 for Certification Purposes in the Nuclear Domain.

[16] European Commission's Advisory Expert Group, Nuclear Regulators Working Group. "Licensing of safety critical software for nuclear reactors – Common Position of seven European nuclear reactors and authorized technical support organizations". Revision 2010.