Scrum, documentation and the IEC 61508-3:2010 software standard

Authors: Thor Myklebust[1,a], Tor Stålhane[b], Geir Kjetil Hanssen[a] Tormod Wien[c] and Børge Haugset[a]
[a] SINTEF ICT
[b] IDI NTNU
[c] ABB

Abstract
Agile development, and especially Scrum, has gained increasing popularity.
IEC 61508 and several related standards for development of safety critical software has a strong focus on documentation, including planning, which shall show that all required activities have been performed. Agile development on the other hand, has as one of its explicit goals to reduce the amount of documentation and to mainly produce and maintain working software.
The problem created by the need to develop a large amount of documents when developing safety critical systems is, however, not a problem just for agile development – it has been identified as a problem for all development of safety critical software. In some cases up to 50% of all project resources has been spent on activities related to the development, maintenance and administration of documents. Thus, a way to reduce the amount of documentation will benefit all developers of safety critical systems.
By going systematically through all the documentation requirements in IEC 61508-1 (general documentation requirements) and IEC 61508-3 (software requirements) and by using the combined expertise of the five authors, we have been able to identify documents that are or can be generated by tools used in the requirement and development process, e.g. logs from requirement and testing tools and documents that can be made as part of the planning and discussions, e.g. snap shots of whiteboards. We have also identified documents that normally can be reused when issuing a new version of the software and identified documents that can be combined into one document.

Keywords: Scrum, safety-critical software, documentation, IEC 61508, Certification

1. Introduction

Agile development, and especially Scrum [1] , has gained increasing popularity and has also been applied in the development of safety critical software, for instance in aviation and automotive [2, 3].
IEC 61508 [4] and several related standards for development of safety critical software has a strong focus on documentation, including planning, which shall show that all required activities have been performed. Agile development on the other hand, has as one of its explicit goals to reduce the amount of documentation and to mainly produce and maintain working software. Assessment of compliance with standards like IEC 61508 is outside the scope of agile methods.
The problem created by the need to develop a large amount of documents when developing safety critical systems is, however, not a problem just for agile development – it has been identified as a problem for all development of safety critical software. In some cases up to 50% of all project resources has been spent on activities related to the development, maintenance and administration of documents [5]. Thus, a way to reduce the amount of documentation will benefit companies that develop safety critical systems. We are, however, motivated by the focus on simplicity and pragmatism in agile methods and believe that adapting principles from agile software development to the development of safety critical systems will help to simplify the work with the documentation and thus to reduce costs.
Our work in this paper has been guided by the following research question: How can information from an agile software development process be used to reduce the documentation costs imposed by IEC61508?

[1] thor.myklebust@sintef.no

The authors have already published papers on how to adapt the agile development process to conform to the standards ISO 9001 (quality systems) [6], IEC 61508 (functional safety systems) [7] and IEC 60880 (nuclear systems) [8]. Some companies have been reluctant to adapt an agile approach due to the perceived risk of having to redo a large amount of documentation for each of the frequent and short iterations in the development cycle. How we have solved this problem is described in chapter 3 and 4 below.
This work has been performed as part of the SUSS[2] project, financed by The Norwegian Research Council.

2. Background

As Scrum and other agile processes are introduced also into the part of the software industry that develops safety critical systems, the industry is caught between the relevant standards that are pre-agile and mostly document driven and the agile concept which tries to avoid producing documents that does not directly relate or contribute to the development of working software. This is based on the agile manifest (http://agilemanifesto.org/) that states "Working software over comprehensive documentation ".
In our opinion the relevant standards overdo their focus on documents, mostly because they overdo their focus on process documentation. It is our experience that a large part of this documentation will only be used for proof of conformance (PoC) which is needed in two cases – for certification and in case the product will be drawn into a court case.
Using an agile approach will reduce the amount of in-process document needed. Another factor that will reduce lead time and cost is to tap the large potential for reuse of whole or parts of important documents. This can, however, only be achieved if they are written with reuse in mind.

SafeScrum

We have earlier attacked similar problems related to standards that were, often implicitly, intended for a document driven, waterfall process such as ISO 9001 [6] and IEC 61508 [7]. Our conclusion is the same in both cases: most of the standards' requirements are met without much ado, some requirements can be solved with a little flexibility from the developers and assessors while there are a few stumbling blocks that need new thinking. Of the 50 top-level requirements in ISO 9001, only four fall into this category. For the IEC 61508, we had to develop a new Scrum process – Safe Scrum – in order to cater to the identified problem areas.
SafeScrum [ibid.] is motivated by the need to make IEC 61508 more flexible with respect to planning, documentation and specification, as well as making Scrum a practically useful approach for developing safety critical systems.
Our model has three main parts. The first part consists of the IEC 61508 steps of developing first the environment description and then the SSRS (Software Safety Requirement Specification) phases 1-4 (concept, overall scope definitions, hazard and risk analysis and overall safety requirements). 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 in a *product backlog.* A product backlog is a list of required features and functions of the system prioritized by the customer.
Due to the focus on safety requirements, we propose to use two related product backlogs, one *functional product backlog*, which is typical for Scrum projects, and one *safety product backlog*, to handle safety requirements. We will 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.

---

[2] Norwegian: Smidig utvikling av Sikkerhetskritisk Software. English: Agile Development of safety Critical Software

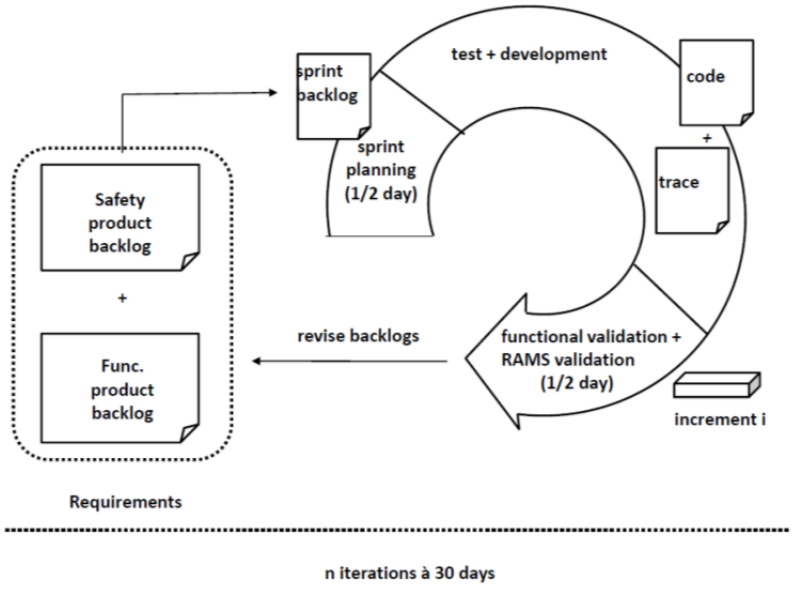

Figure 1: The SafeScrum model

Each Scrum iteration can be considered as a mini waterfall project or a mini V-model, and consists of planning, development, testing, verification and also validation. 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 Figure 1. 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 traceability in addition to many other process support functions. Two out of many examples are Jira (www.atlassian.com/software/jira) and Rally software (www.rallydev.com).
An important practice in many Scrum projects is *test-driven development*, where the test of the code is written *before* the code 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 behaviour of the code, based on the requirements, 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 design, test rig or executable code. The sprint ends by demonstrating and validating the developed code to assess whether it meets the requirements 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 fit with 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, an assessor may take part in this validation for each sprint. The assessor could also take part in the retrospective after each sprint to help the team to keep safety consideration in focus. 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. 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.
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, the final RAMS validation will be less extensive than when using other development paradigms. This will also help us to reduce the time and cost needed for certification.

Test Driven Development
TDD is a popular practice in agile development and is often used to supplement Scrum. We see TDD as a natural part of SafeScrum as well and believe that this may produce documentation being useful as PoC. TDD is a practice where all new code at the procedure or method level first needs to be described as a suite of mock objects and assertions (expected results from given inputs). The total

collection of unit tests grows as the code grows and is automatically executed frequently to test if the code works as defined after each change.

Trust
We have checked requirements related to "Trust" in several IEC and ISO standards [9-17]. During assessment work, we have observed that the level of trust that the assessor have in the manufacturer may affect the level of documentation needed for the approval of the product. In the standards evaluated, only ISO/IEC 17021 [13] mentioned the level of trust the assessor have in the manufacturer. Quote"*Familiarity (or trust) threats: threats that arise from a person or body being too familiar with or trusting of another person instead of seeking audit evidence*". This standard is also the only standard that mentions the requirements for trust related to the assessor (third party).The level of trust the assessor have in the manufacturer is a subjective issue so it is important to discuss the level of details, possible excessive bureaucracy and pragmatism with the assessor at the beginning of the certification process. The important issue is that the manufacturer has the information they need to do their job and the assessor to do his job.
Trust as a topic in this respect is closely linked to the level of competence and experience of the personnel.
In practice trust is mainly related to people, not organizations. This has been experienced by one of the SUSS participating companies when the certification body changed several of their assessors and as a result, trust was decreased.

Industrial challenges
The development of safety-critical systems is guided by document-driven and process-heavy standards. The safety-standard, IEC61508, assumes extensive documentation and strictly defined processes for the product safety certification including risk analysis, change control and traceability. Therefore the speed of change is lower in such projects, making them less flexible with respect to changing requirements from customers and markets.
The safety process being mostly a document driven process, where each step from planning and specification to design, coding, testing and validation and verification need to be documented as a Proof of Concept, put a lot of emphasis on the project organization and the competence and experience of the people involved. Furthermore, the requirement that there shall be unique traceability all the way from requirements to design, implementation, testing and validation and verification, complicates the picture and assumes the use of labor extensive procedures to be able to cope with often large amount of data. Data, which over the lifetime of a project that can span several years, is not necessarily static. Compared to a "normal" software development project, testing is without doubt the task requiring most additional effort. This is due to the rigid requirements on the documentation process to verify that the required functionality is implemented as specified. Tests must be implemented at all levels (unit, functional, system) with unique traceability, covering normal, exceptional and erroneous operation.

3. Requirements related to documentation
In search of a potential reduction of the necessary documentation we believe that proper adoption of agile software development principles from the Scrum methodology may reduce the costs of documentation. We expect to see two cost saving effects: 1) it will reduce lead time and increase the development process flexibility, thus reducing development costs and, 2) it will reduce the number of new documents. When doing modification of an already certified product, only a few documents are new e.g. test reports. Furthermore these documents can be based on templates or reuse (see IEEE std 1517:2010 [18] for more information related to reuse) or be automatically generated to further reduce documentation costs. See table I.

The challenge with this solution is to keep the process and available documentation in line with the IEC 61508 requirements while at the same time gaining the benefits from an agile development process. As described below, we can achieve this through a systematic walkthrough of the IEC 61508

requirements and only keep the minimum of documents or information that are needed to meet the standard's requirements.

Method when evaluating IEC 61508-1 documentation requirements
We have used the same method for the work reported here as we have used earlier – see [6, 7]. The process consists of the following two steps:

1. Check each relevant part of the standard (Part 1 ch. 5) and for each requirement ask "If we use Scrum, will we still fulfil this requirement?" this check is used to move the requirements into one out of three parts of an issues list – "OK", no further action requirement, "?", needs to be discussed further and "Not OK", will require changes to Scrum and, in a long term perspective, to IEC 61508. In addition to the issues list we will also get a lot of input to how to modify the Scrum process in order to reduce the amount of conflicts.
2. Check all requirements that are in the categories "?" and "Not OK" against a modified Scrum process model – in our case Safe Scrum. This will reduce the number of problematic requirements further. In addition, the accompanying discussions will enable us to identify new ways of tackling some of the problems discovered.

This process is used on the case at hand in the section "IEC 61508 walkthrough". Most of the categorizations done on the standard's requirements are to a certain degree subjective. For this reason we have included all relevant roles in the assessment: one assessor, two Scrum expert, one safety experts and one representative for a company that routinely have to have their software products certified.

IEC 61508-1 walkthrough of chapter 5 "Documentation"
We have gone through the section 5.2 - Requirements on documentation - in IEC 61508, part 1. The documentation requirements in IEC 61508, part 3 is just a reference to part 1 of the standard. The result from the first iteration of the IEC 61508, part 1, section 5.2 walkthrough was that out of a total of 11 issues, we found that

- Five was "OK".
- One was "not OK" (5.2.3 below). As a result Scrum has to be adapted. The adaptation is included in SafeScrum.
- Five needed further investigation – "?"

The second iteration focused on the following six issues:

- 5.2.1. The documentation shall contain sufficient information, for
    - each phase of the overall, E/E/PES and software safety lifecycles completed.
      These documents will fall in the class Reusable documents (see ch. 4 below)
    - necessary for effective performance of subsequent phases.
      SafeScrum is mainly performed as part of phase 10 Realisation. Anyway an agile approach should, where possible, also be used for the other phases to ensure optimalization of the work involved
    - verification activities.
      The verification process should use automatic testing tools – e.g., Cucumber (http://cukes.info/) or Fitnesse (http://fitnesse.org/). This will also enable a considerable amount of pragmatic reuse.

  The problem for Scrum, compared to traditional Scrum, is traceability. In order to handle this problem, we have added an extra activity to handle all traceability in SafeScrum.
- 5.2.3 The documentation shall contain sufficient information required for the implementation of a functional safety assessment, together with
    - the information and
    - results derived from any functional safety assessment.

  This problem is partly taken care of by the SafeScrum process but the assessor will need more information, which is not available from Scrum as it is practiced now. This means that SafeScrum needs to be complemented by normal functional safety assessment.

- 5.2.4 The information to be documented shall be as stated in the various clauses of this standard unless justified or shall be as specified in the product or application sector international standard relevant to the application
  We should be pragmatic when fulfilling this clause, since this opens up for a wide range of interpretations for what should be accepted as PoC. The most important thing here is, however, to discuss this with the assessor ***before*** *the project starts in order to get an agreement on the information that will be needed.*
- 5.2.5 The availability of documentation shall be sufficient for the duties to be performed in respect of the clauses of this standard.
  In order to make all relevant documents available for the assessor we need first of all to register all relevant information. The simplest way to do this is to use a whiteboard and to take snap-shots. Theses snap-shots, together with the date and a list of participants should be accepted as process documentation. When the relevant documents are registered there exist several tools for sharing information like e.g. www.projectplace.com.
- 5.2.10. The documents or set of information shall be so structured as to make it possible to search for relevant information. It shall be possible to identify the latest revision (version) of a document or set of information.
  All relevant documents must be stored in a project database and indexed properly.
- 5.2.11. All relevant documents shall be revised, amended, reviewed and approved under the control of an appropriate document control scheme.
  The important question here is when – e.g., after each iteration, after some iterations or just when we have finished all development iterations. Using the methods suggested for section 5.2.5 it is easy to conform to the two first points – revised and amended – while the last two – reviewed and approved – might be problematic in the sense that it will bureaucratize and delay the Scrum process, thus reducing its effect. These review aspects are normally included in the contract between the manufacturer and the assessor.

  Two important things can be done:
  - *Move much of the necessary documents out of the Scrum iteration loop.*
  - *Get an agreement with the assessor as to which iterations need to be included in 5.2.11 and how this can be performed when using e.g. databases.*

IEC 61508 walkthrough of the normative Annex A "Guide to the selection of techniques and measures" of Part 3

Although annex A in IEC 61508, part 3 is not directly related to documents and PoC, it gives an overview of the needed activities and thus indirectly an overview of the necessary PoC. The 10 tables – A1 – A10 – contains a total of 70 requirements. In order to simplify a walkthrough of these tables we have decided to assume SIL 2 development, remove all issues related to maintenance and only consider the activities that are marked as HR – Highly Recommended (although, in practice, some R activities should be performed). This reduces the number of issues to 19. The two tables A3 and A4 are only concerned with pre-development activities. Three tables – A5, A6 and A7 – are only concerned with testing and the PoCs can be sufficiently covered by the automatically generated test logs. Table A2 is concerned with design activities. In our opinion, the PoC will in some cases be satisfied by white-board snapshots plus a list of participants. High level design – architecture – is decided before we enter SafeScrum. Using the whiteboard for detailed design has some pros and cons. Pro: quick, can document the design process, not only the final result. Con – may lack the formality achieved by a document.

The only challenge is table A9 "SW verification", which is concerned with static and dynamic analyses. When we check the more detailed tables – B2 "dynamic analysis and testing" and B8 "static analysis" – we see that the PoC for the requirements in B2 are covered by the test logs. The only remaining challenges are in B8, which requires analysis of control- and data flow. This document will have to be done separately (outside SafeScrum) but only when the system is finished and ready for certification.

4. Classification of the documentation

The relevant documents for Part 3 are presented in Table A.3[3] "Example of a documentation structure for information related to the software lifecycle" in Part 1 of IEC 61508.

Copy from Part 1:

Tables A.1, A.2 and A.3 provide an example documentation structure for structuring the information in order to meet the requirements specified in Clause 5. The tables indicate the safety lifecycle phase that is mainly associated with the documents (usually the phase in which they are developed). The names given to the documents in the tables are in accordance with the scheme outlined in A.1. In addition to the documents listed in Tables A.1, A.2 and A.3, there may be supplementary documents giving detailed additional information or information structured for a specific purpose, for example parts lists, signal lists, cable lists, wiring tables, loop diagrams and list of variables.

There are several levels of documentation in a software project. The documents at these levels have different sources, different costs but often the same roles, both in the project itself and when it comes to certification.

- **Reusable documents** – low extra costs. This is documents where large parts are reused as is, while small parts need to be adapted for each project and even for each sprint for some documents. If reuse is the goal right from the start, the changes between projects or iterations will be small. For further information about reuse see IEEE std 1517 [18].
- **Combined -** Identify documents that can be combined into one document
- **Automatically generated documents** – high initial costs but later low costs. This is documents that are generated for each new project or iteration by one or more tools. Examples are test results and test logs from Jira and requirements documents from Doors (www-03.ibm.com/software/products/en/ratidoor/).
- **New documents** – high costs. This is documents that have to be written more or less from scratch for each new project.

In the table below, we have classified the documents that are specified in table A.3 regarding software in Part 1 of IEC 61508.

IEC 61508-1, table A.3 for SW	Classification and comments
1. Specification (software safety requirements, comprising: software safety functions requirements and software safety integrity requirements)	Generated from e.g. a requirement management tool and/or backlog management tool and is reusable. For further information see IEEE Std 830-1998 [19] and IEEE Std 1233-1998 [20].
2. Plan (software safety validation)	Reusable. The document can be combined with document 26. For further information see IEEE Std 730-2002 [21].
3. Description (software architecture design)	Reusable. For further information, see ISO/IEC/IEEE Std 42010 [22], IEEE Std 1016 [23] and www.sysmlforum.com/ regarding SysML model management.
4. Specification (software architecture integration tests);	Reusable. The standard ISO/IEC/IEEE 29119-3:2013 [24] "Test Documentation" includes relevant information related to specification of tests.
5. Specification (programmable electronic hardware and software [4]integration tests);	Reusable

[3] Similar tables exists for Part 1 and Part 2

[4] Observe definition 3.8.1 in Part 4 related to integration tests

IEC 61508-1, table A.3 for SW	Classification and comments
6. Instruction (development tools and coding manual)	Reusable. New development tools have to have relevant instructions. See existing coding manuals/information issued by Exida for C/C++ [25] and a Guideline issued by MISRA[5] for C++ [26]. See www.misra-cpp.com/ for further information.
7. Description (software system design);	Reusable For further information, see IEEE Std 1016 [23].
8. Specification (software system integration tests)	Reusable. The document can be combined with documents 9 and 10.
9. Specification (software module design);	Reusable. The document can be combined with documents 8 and 10. For further information, see IEEE Std 1016 [23].
10. Specification (software module tests)	Reusable Can be combined with documents 8 and 9
11. List (source code);	Source code can easily be generated directly from the code management system. Also, there are many tools that may automatically produce code documentations. E.g. Doxygen (www.doxygen.org) and other similar tools.
12. SW module design: Report (software module tests);	Generated. Some of the tests are generated automatically, others are semi-automatic and some are manually.
13. Report (code review)	Combined. Doc 13, 14, 15, 16 and 17 can be one report. The documents can be developed gradually so. There exist several tools for static code analysis (e.g. http://cppcheck.sourceforge.net/ for for static C/C++ code analysis) and code review (e.g. www.parasoft.com/cpptest). See also IEEE 1028:2008, IEEE Standard for software reviews and audits [27]. This standard defines five types of software review and audits. In this edition of the standard there is a clear progression in informality from the most formal, audits, followed by management and technical review, to the less formal inspections, and finishing with the least formal inspection process - walkthroughs.
14. SW module testing: Report (software module tests)	Generated. Doc. 13, 14, 15, 16 and 17 can be one report Some of the tests are generated automatically, others are semi-automatic and some are manually.
15. Report (software module integration tests);	Generated. Doc 13, 14, 15, 16 and 17 can be one report. Some of the tests are generated automatically, others are semi-automatic and some are manually.
16. Report (software system integration tests);	Generated. Doc 13, 14, 15, 16 and 17 can be one report Some of the tests are generated automatically, others are semi-automatic and some are manually.
17. Report (software architecture integration tests)	Generated. Doc 13, 14, 15, 16 and 17 can be one report. Some of the tests are generated automatically, others are semi-automatic and some are manually.

[5] MISRA: The Motor Industry Software Reliability Association. www.misra.org.uk

IEC 61508-1, table A.3 for SW	Classification and comments
18. Report (programmable electronic hardware and software integration tests)	Generated. Some of the tests are generated automatically, others are semi-automatic and some are manually.
19. Instruction (user);	Reusable Can be combined with 20. For further information, see IEEE Std 1063 [28].
20. Instruction (operation and maintenance)	Reusable. Can be combined with document 19
21. Report (software safety validation)	Newly developed.
22. Instruction (software modification procedures);	Reusable
23. Request (software modification);	Newly developed. Can be combined with document/database 25.
24. Report (software modification impact analysis);	Newly developed. A template has been presented in [29].
25. Log (software modification)	Newly developed. Tools exist for software modifications like e.g. the open source tool bugzilla, www.bugzilla.org. Can be combined with document/database 23.
26. Plan (software safety);	Reusable The document can be combined with document 2. For further information, see IEEE Std 1228 [30].
27. Plan (software verification);	Reusable
28. Report (software verification);	Generated. Some of the tests are generated automatically, others are semi-automatic and some are manually.
29. Plan (software functional safety assessment);	Reusable.
30. Report (software functional safety assessment)	Reusable. Finished after the last test/verification/validation report
31. Safety manual for compliant items	Reusable. May have a few remaining parts after the last test/verification/validation report

Table 1: Table A.3 regarding SW documentation in Part 1 and corresponding classification

Overview of document types as presented in A.3 in Part 1 of IEC 61508:

Documents Nu as listed in Table 1 Above	Comments
11 reports (Nu 13, 14, 15, 16, 17, 18, 21, 24, 28 and 30).	The standard ISO/IEC/IEEE 29119-3:2013 includes procedures and templates for Test status report, Test completion report, Test data readiness report, Test environment readiness report, Test incident report, Test status report and Test completion report.
6 specifications (Nu 1, 4, 5, 8, 9 and 10. 4, 5, 8 and 10 are test specifications)	The standard ISO/IEC/IEEE 29119-3:2013 includes both agile and traditional procedures for specifications and examples regarding Test design, Test case and Test procedure.
four plans (Nu 2, 26, 27, 29)	Validation, safety (can be based on e.g. EN 50126 [31] or IEEE Std 1228 [30]), verification and functional safety assessment
four instructions (Nu 6, 19, 20 and 22)	Development tools and coding manuals User, operation and maintenance instructions Modification procedure

Documents Nu as listed in Table 1 Above	Comments
two descriptions (Nu 3 and 7)	SW architecture design and SW system design
a list (Nu 11)	List source code
a request (Nu 23)	Request SW modification. Tools exist for software modifications like e.g. the open source tool bugzilla, www.bugzilla.org. Can be combined with document/database 23.
a log (Nu 25)	SW modification
a manual (Nu 31)	Safety manual for compliant items

Table 2: Overview of Table A.3 SW documents

The main documents are the reports, specifications and plans. As seen from the overview above, these documents should be the focus when trying to reduce the documentation work.

Overview of the document classes is shown in the table 3 below.

Class	Document number	Comments
Reusable	16 documents: 2, 3, 4, 5, 6, 7, 8, 9, 10, 19, 20, 22, 25, 26, 29 and 30	Reusable documents should be made more generic by the manufacturer. For documents that shall be updated as part of several sprints, reuse solutions is very important. These documents could e.g. include tables or a point list that are easily updated. For more information, see IEEE std 1517:2010 [18].
Combined	2 documents: 2 and 26 3 documents: 8, 9 and 10 5 documents: 13, 14, 15, 16 and 17 2 documents: 19 and 20 2 documents/databases: 23 and 25	12 documents can be merged to four documents. References are simplified when combining documents. The general parts are often the same. The relation between activities etc, is more visible. However this, to some extent, depends on e.g. the size of the project.
Generated	9 documents: 1, 11, 12, 14, 15, 16, 17, 18 and 28	Several possibilities exist. This will be studied later in the project.
New documents	5 documents: 6 (new tools) 21 (SW safety validation), 23 (request: SW modification), 24 (SW modification impact analysis) and 25 (log: SW modification).	Discussions with the assessor: As part of the Scrum mindset it is important to reduce the amount of documentation and it is assumed that the assessor should be involved early in the project. What could be a minimum of documentation should therefore be discussed with the assessor before starting to develop any new document. Templates and examples: For some documents templates and examples has already been developed as part of research, standardization and organizational work. See e.g. [29], ISO/IEC/IEEE 29119-3:2013 and www.misra.org.uk.

Table 3: Classes of documents

5. Discussion and conclusion

The acceptance of a system that has safety critical components rests on three pillars – agreements with the assessor, trust in the developers and competent work. This holds, independent of standard and development methods applied. The pillars are, however, not constructed independently. In our experience, an agreement with the assessor must come first. This will enable us to settle important questions such as:

- Which parts of Scrum may pose problems later in the project?
- What is accepted as PoC for each activity?
- Which documents are needed, in which form and when?

When this is in place, we can start to build trust based on demonstration of competence and strict adherence to all agreements.

Our conclusion is simple – the requirement that we need to certify a system according to IEC 61508 cannot be used as an argument against using the Scrum development process. The problems that exist are not a consequence of formulations of the standard's requirements but are related to what the individual assessor will accept as PoC for an activity.

We have looked into the documents necessary for approval of the software and grouped them according to the opportunity for reuse, combination of several documents into one, documents generated automatically and new documents.

Only five of the documents are new documents when doing recertification. In addition we suggest that new documents should initially be discussed with the assessor, having trust and Scrum philosophy in mind to ensure correct level of documentation.

As part of our ongoing work on safety critical systems development we will try out the described approach in an industrial environment. This will partly be done to see if the approach needs modifications and partly to see how the assessors can be involved so that we can get a more efficient cooperation. We will also study how we can build trust between developers and assessors. This will not remove the need for PoCs but it will allow the assessor to focus on the few, critical parts of his works and leave the rest to the developers.

References

[1] K. Schwaber, Beedle, M., *Agile Software Development with Scrum*. New Jersey: Prentice Hall, 2001.

[2] M. Müller, "Functional Safety, Automotive SPICE® and Agile Methodology at KUGLER MAAG CIE GmbH," presented at the 8th Automotive Software Workshop, 2011.

[3] C. Webster, N. Shi, and I. S. Smith, "Delivering Software into NASA's Mission Control Centre Using Agile Development Techniques," presented at the Aerospace Conference, Big Sky, USA, 2012.

[4] IEC, "61508:2010 Functional Safety of Electrical/Electronic/Programmable Electronic Safety-related Systems (E/E/PE, or E/E/PES)," ed.

[5] T. e. a. Wien, "Reducing Lifecycle Costs of Industrial Safety Products with CESAR " presented at the Emerging Technologies and Factory Automation (ETFA), Bilbao, Spain, 2010.

[6] T. Stålhane and G. K. Hanssen, "The application of ISO 9001 to agile software development," presented at the Product Focused Software Process Improvement (PROFES 2008), Frascati, Italy, 2008.

[7] T. Stålhane, T. Myklebust, and G. K. Hanssen, "The application of Scrum IEC 61508 certifiable software," presented at the ESREL, Helsinki, Finland, 2012.

[8] T. Stålhane, V. Katta, and T. Myklebust, "Scrum and IEC 60880," presented at the FFI seminar, Storefjell, Norway, 2013.

[9] ISO, "19011: Guidelines for auditing of management systems. Ed. 2," ed, 2011.

[10] ISO/IEC, "17000: Conformity assessment – Vocabulary and general principles. Ed.1," ed, 2004.

[11] ISO/IEC, "17011: Conformity assessment – General requirements for accreditation bodies accrediting conformity assessments bodies. Ed. 1," ed, 2004.

[12] ISO/IEC, "17020: General criteria for the operation of various types of bodies performing inspection. Ed. 2," ed, 2012.

[13] ISO/IEC, "ISO/IEC 17021 Conformity assessment – Requirements for bodies providing audit and certification of management systems," ed, 2011.
[14] ISO/IEC, "17024: General requirements for bodies operating certification of persons. Ed. 2," ed, 2012.
[15] ISO/IEC, "17025: General requirements for the competence of testing and calibration laboratories. Ed. 2," ed, 2005.
[16] ISO/IEC, "17065: Conformity assessment – Requirements for bodies certifying products, processes and services. Ed. 1," ed, 2012.
[17] ISO/IEC, "17067: Conformity assessment – Fundamentals of product certification and guidelines for product certification schemes," ed, 2013.
[18] IEEE, "1517 standard for information technology – System and software life cycle processes – Reuse processes. Ed. 2," ed, 2010.
[19] IEEE, "Std 830 Recommended Practice for Software Requirements Specifications," ed, 1998.
[20] IEEE, "Std 1233 Guide for Developing System Requirements Specifications," ed, 1998.
[21] IEEE, "Std 730 Standard for Software Quality Assurance Plans," ed, 2002.
[22] ISO/IEC/IEEE, "Std 42010 Systems and software engineering - Architecture description. ," ed, 2011.
[23] IEEE, "Std 1016 Recommended Practice for Software Design Descriptions," ed, 2009.
[24] ISO/IEC/IEEE, "29119-3. Software and systems engineering – software testing – Part 3: Test documentation. Ed. 1," ed, 2013.
[25] Exida, "C/C++ Coding Standard recommendations for IEC 61508," ed, 2011.
[26] MISRA, "Guidelines for the use of the C++ language in critical systems," ed, 2008.
[27] IEEE, "Standard for software reviews and audits. Ed. 2," ed, 2008.
[28] IEEE, "Std 1063 Standard for Software User Documentation," ed, 2001.
[29] T. Myklebust, T. Stålhane, G. K. Hanssen, and B. Haugset, "Change Impact Analysis as required by safety standards, what to do?," presented at the Probabilistic Safety Assessment & Management conference (PSAM12), Honolulu, USA, 2014.
[30] IEEE, "Std 1228 Standard for Software Safety Plans," 1994.
[31] EN, "50126 Railway applications - The specification and demonstration of Reliability, Availability, Maintainability and Safety (RAMS)," 1999.