Change Impact Analysis in Agile Development

Tor Stålhane
Norwegian University of Science and Technology
+47 73594484, stalhane@idi.ntnu.no

Vikash Katta
OECD Halden Reactor Project and Norwegian University of Science and Technology
+47 45464323, vikash.katta@hrp.no

Thor Myklebust
SINTEF ICT
+47 95779869, thor.myklebust@sintef.no

Abstract

Any change that will influence the system's safety requirements after we have finished the safety analysis and safety planning for development of safety-critical software will require a change impact analysis. One of the reasons for using Scrum is its declarations to "embrace change". Using agile development we may add new requirements, change existing requirements and make current requirements more detailed both in the product backlog and in the sprint backlogs.

Problems, methods and necessary reports related to the consequences of such changes have been discussed and analysed in the SUSS (Agile Development of Safety Critical Software) project. We have come up with an extended Scrum model which includes the change impact analysis for all potential changes to the Scrum backlog. In addition, we have designed a change impact analysis process plus a template for a change impact analysis report.

Before we entered the software development part of the project, which is Scrum's domain, we have already performed a HazOp or FMEA and a risk analysis. Changes to the requirement or the introduction of new requirements may, however, invalidate these analyses. We may thus need to perform a change impact analysis for the whole system which again may force us to repeat all or part of the first safety analyses.

One big challenge when performing a change impact analysis is to identify which parts of the system under development that is affected by a certain change. Manually this is rather costly process. One of the authors has developed a prototype tool that will help in the process. This tool was, however, not specifically built to be used in agile development and we will discuss how this prototype tool can work in or be adapted to an agile development process.

1. Introduction

As agile development is getting more and more used, we need to consider change impact analysis not only in the traditional way but also throughout the complete development cycle. To quote an anonymous developer: "After the first increment (sprint) all the rest is maintenance". The reason for this situation is that agile development, as one of its important ideas, wants to "embrace change". Unfortunately, for development of safety critical systems change also means change impact analysis. When we change a piece of code, we face two important challenge which we will discuss in this paper:

- How much change impact analysis should be used for this change.
- Which parts of the code do we need to analyse – the traceability problem.
- How can a tool help us to solve or reduce the traceability problem

The rest of the paper is organized as follows: Section 2 gives a short summary of some of the work done on change impact analysis. Section 3 discusses the challenge posed by agile development on change impact analysis while section 4 discusses how SafeScrum can be adapted to the IEC

60880 standard. Section 5 discusses change impact analysis, both for IEC 61508 and for IEC 60880. Section 6 describes a tool – ToSS – that can help us with traceability, section 7 discusses threats to the validity of our work, while section 8 sums up the main conclusions and present out thoughts on further work in this area.

2. Change impact analysis – state of the art

There has been a lot of research on change impact analysis – mainly due to its importance for software maintenance. Change impact analysis is closely related to traceability in two ways:

- From code to requirements – which requirements are affected if we change this code? This also gives us information on which tests that need to be re-run.
- From requirements to code – what code must be changed if this requirement is changed?

First and foremost, B. Li et al. [1] has published the result of a survey, where they have identified 23 change impact analysis methods. In addition, another survey, performed by S. Lehnert [2] has reviewed 150 approaches and related literature. We will not go through all these methods in detail. Instead, we will look at two of the methods reviewed by B. Li et al. and two papers that are not mentioned in either survey.

The first paper we will study in some more details is written by M. Acharya and B. Robinson from ABB Corporate Research [3]. The authors discuss the use of automatic program slicing as a means for change impact analysis. They have developed a new framework for change impact analysis based on slicing and developed a tool for this, called Imp. This tool is designed to seamlessly integrate with the nightly build process. The approach is tested in an experiment with 30 changes in two versions of the same system and seems to work well. However, it still has the same weakness as most other change impact analysis methods – it must start with the original and the changed code.

The second paper that we will discuss is written by M.S. Kilpinen et al [4]. This paper does not discus solely software but instead discuss change impact analysis for the whole system – hardware and software. In addition, they do not assume that changes have been done before the change impact analysis is performed. The experiences reported stems from a Rolls Royce project for developing a jet engine controller. Design changes were managed through an informal change impact process early in the detailed design and a more formal process when the design baseline had been defined. Their most important observation is that “the system engineers tend only to use their experience and knowledge of the system rather than any systematic method to brainstorm on the impact on the system requirements given to embedded software” in relation to this, we should keep in mind Lindvall and Sandahl’s observation [5] that “software engineers tend to perform impact analysis intuitively. Despite this common practise, many software engineers do not predict the complete change impact.”

Based on this short survey there seems to be little tool support for change impact analysis decision support, which is what we need. Knowing post festum that the change was a bad idea is important but it is much cheaper to know this before we start to change anything.

3. Why is change impact analysis important for agile development

First and foremost – agile development’s dictum about embracing change will lead to a lot of changes during development. There will be changes to functional requirements – new control

function needed – changes to the software's environment, changes to the hardware – a new sensor is introduced – and changes due to anomalies found during testing. The ability to handle such changes without throwing the whole project into turmoil is one of the strong points of Scrum.

One the one hand we need to perform change impact analysis when needed. On the other hand, we do not want to bug down development with a lot of analysis and documentation when the idea is to stay agile. In this paper we thus need to consider two issues:

- When do we need to perform change impact analysis? Will it e.g. be necessary to perform a change impact analysis when a requirement is taken out of the product backlog and elaborated on before being inserted into the sprint backlog?
- Where in the development process should it be done?

IEC 60880, as a lot of other standards, have strict requirements on handling of changes. This is due to the fact that changes can influence the system's safety or the safety analysis already done. Some standards have rather loose requirements for change impacts – e.g. they just require that a change impact analysis shall be performed under certain circumstances. IEC 60880, on the other hand, has a quite elaborate process for change impact analysis.

4. IEC 60880 and SafeScrum

4.1 SafeScrum

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 such as the V-model, which emphasize thorough and detailed planning, an upfront design, and consecutive plan conformance. Several agile methods are in use, whereof Scrum [12] is one of the most commonly used.

In order to realize some of the benefits of agile development, such as better quality, more efficient development and closer involvement of customers, we have proposed a method called SafeScrum [13]. This variant of 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 conforming to standards, e.g. IEC 61508, IEC 60880 and EN 50128, as well as making Scrum a useful approach for developing safety critical systems. The rest of this section explains the components and concepts of this approach.

The basic SafeScrum model is shown in figure 1. The most important differences from the traditional Scrum model are that we have added a trace process and have split the product backlog into two parts – one for safety related requirements and one for functional requirements.

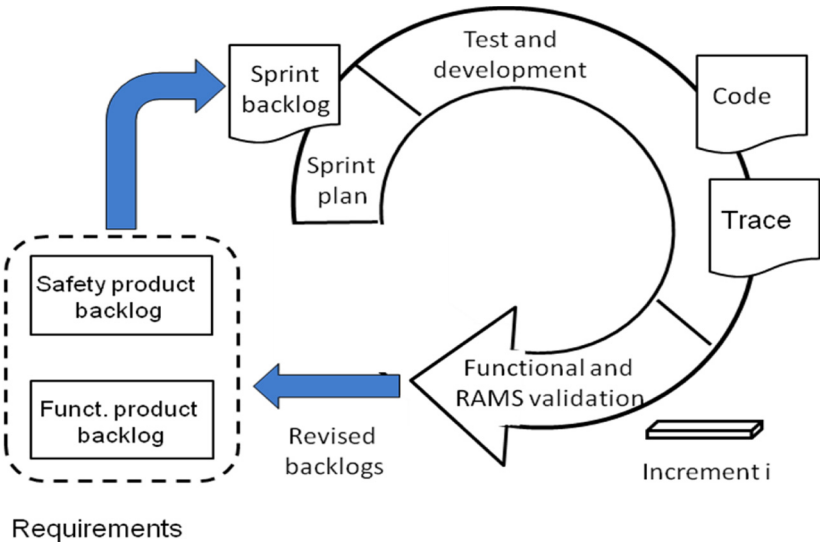

Figure 1 SafeScrum model

As shown in figure 2, our model has three main parts based in the IEC 61508 process. The first part consists of the steps needed for developing the environment description and the Software Safety Requirements Specification (SSRS) 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 as *product backlogs.* A product backlog is a list of all of the functional and safety related system requirements, prioritized by the customer or a similar role having the users and/or owners view and interests in mind. It might be practical to also include the developers in the prioritization process. 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.

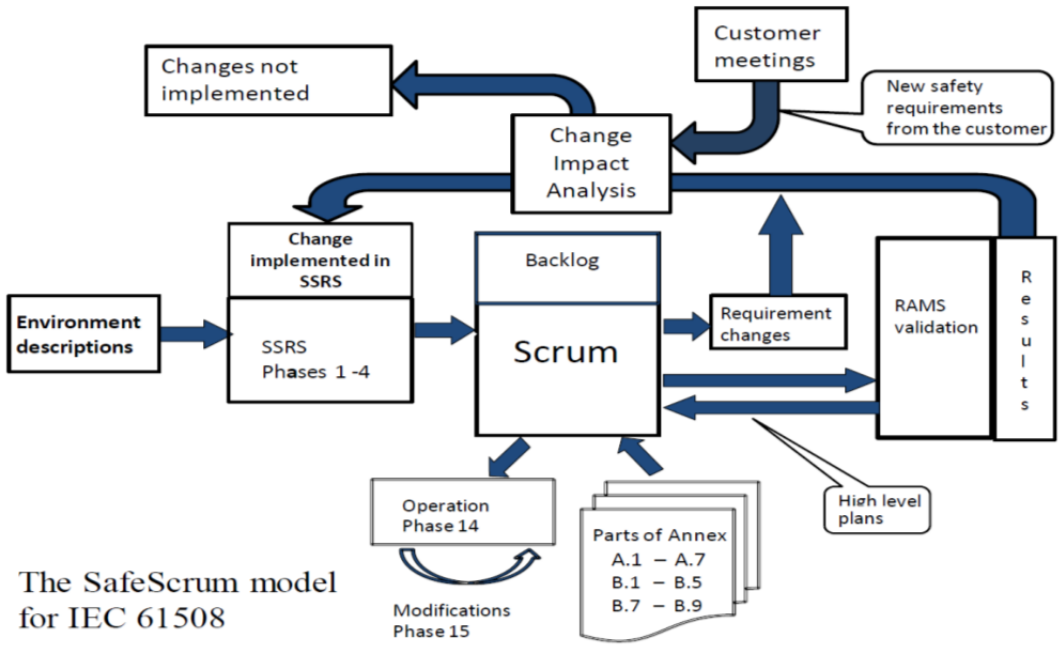

Figure 2 SafeScrum overview – separation of concerns

4.2 What makes IEC 60880 special

From the diagram below, taken from IEC 60880, we see that the software design and implementation only concern a small part of the total process. It is only this part that is touched by SafeScrum, the rest is Scrum independent.

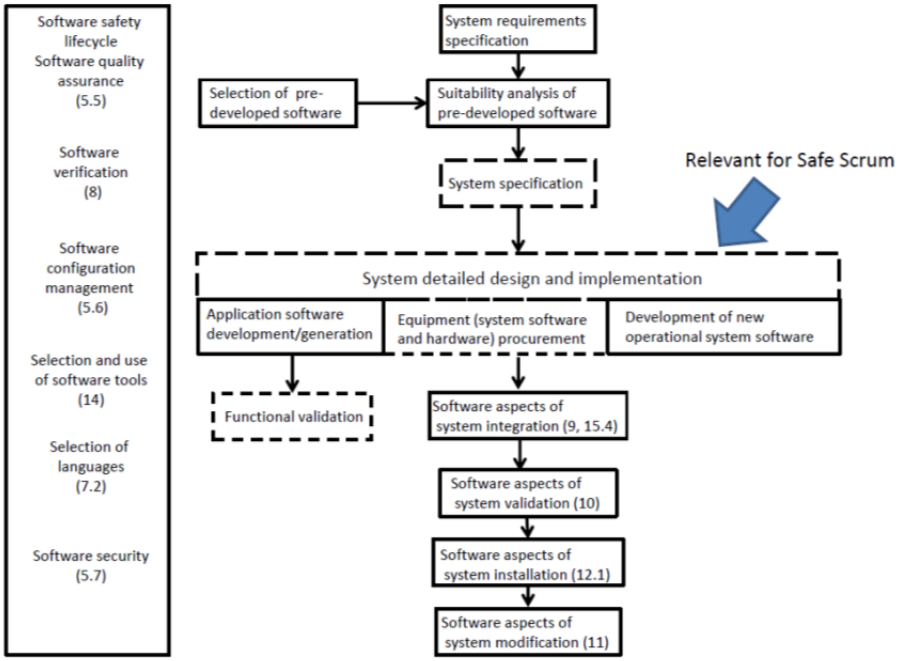

Figure 3 The IEC 60880 process

We studied sections 5 to 10 of IEC 60880 in details. Based on the requirements stated in these sections, we selected the following areas for a closer scrutiny: sections 5.3, 5.4, 5.5, 6.1 and 7. As a consequence of the reference to annex F in IEC 60880, we also include sections 8.2.2 and 8.2.3 for a closer look.

4.3 How can we adapt SafeScrum to IEC 60880

After having studied all relevant areas, we have added some new steps to the SafeScrum process. A complete description can be found in [13]. The following are some of the important observations made during the study.

- According to IEC 60880, each phase will need a set of documents generated at the end of a phase to prove that the activities are done according to the standard. The two documents (see Annex F in IEC 60880) verification report and test report can – at least partly – be generated automatically from appropriate tools.
    - Software test specification document. Given the dynamic nature of requirements handling in Scrum, this report has to be written when handling each requirement, based on the tests designed for this requirement.
    - Software code verification report – written at the end of each sprint.
    - Software test report – written at the end of each sprint.
- “Each phase shall be systematically terminated by a review...” Scrum already has a review at the end of each sprint but this may need to be extended in order to meet the assessor’s interpretation of the standard’s requirements.

- "...the process of laying down software requirements shall be rigorous". It is common in Scrum to elaborate the requirements when they are taken out of the sprint backlog. This will, however, not be sufficient in our case and we must adapt Scrum – all requirements must be rigorously defined when they are
    - Inserted into the backlog.
    - Taken out of the sprint backlog.
    - Revised and reinserted into the backlog.

IEC 60880 - part 4 is about subroutines and goes a long way towards recommending test-driven development, albeit without actually using this term. The important challenge is found in Annex B (normative), item B4.gc which requires that "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 Scrum development process. The rational for this requirement is that it avoids duplication of work and will speed up licensing. Thus, it should be possible to argue for other way to document test input and results as long as the two above mentioned conditions are fulfilled.

5. Change impact analysis

5.1 IEC 61508 and Change Impact Analysis

Requirements for change impact analysis during change are found in most standards on the development of safety critical systems. The following requirements from IEC 61508, part 1, section 7.16.2 are important for SafeScrum:

- **7.16.2.3:** An impact analysis shall be carried out that shall include an assessment of the impact of the proposed modification or retrofit activity on the functional safety of any E/E/PE safety-related system. The assessment shall include a hazard and risk analysis sufficient to determine the breadth and depth to which subsequent overall, E/E/PE system or software safety lifecycle phases will need to be undertaken. The assessment shall also consider the impact of other concurrent modification or retrofit activities, and shall also consider the functional safety both during and after the modification and retrofit activities have taken place.
- **7.16.2.5:** Authorization to carry out the required modification or retrofit activity shall be dependent on the results of the impact analysis.
- **7.16.2.6:** All modifications that have an impact on the functional safety of any E/E/PE safety-related system shall initiate a return to an appropriate phase of the overall, E/E/PE system or software safety lifecycles. All subsequent phases shall then be carried out in accordance with the procedures specified for the specific phases in accordance with the requirements in this standard.

The impact analyses requirements in IEC 61508 and in several other standards does, however, not provide any guidance to how this analyses shall be carried out and how it shall be documented. This is in some sense good, since it leaves it to the developers and the assessors to agree on what is needed on a case by case basis. On the other hand, it leaves companies that are new to the trade totally in the dark.

5.2 IEC 60880 and SafeScrum

Due to the strict change impact analysis requirements of IEC 60880 we also need to make sure that the SafeScrum change impact analysis is compliant with the standard's requirements. As always when applying SafeScrum, it is necessary (1) to select an approach to a certain process or activity and then (2) to get the assessor's acceptance for this process or activity.

The following requirements for change impact analysis, found in the IEC 60880 standard on the development of safety critical systems, are important for SafeScrum.

- **11:** A software modification is a change made to the software which usually impacts both the executable code and the documentation. A software modification may be requested for reasons such as:
  - changes to functional requirements;
  - changes to the software environment;
  - changes to the hardware;
  - anomalies found during test or operation.
- **11.1.6:** The following items shall also be examined in the evaluation of the modification request:
  - technical feasibility
  - impact upon the rest of the system (e.g. memory extension) or upon other equipment (e.g. test systems) in which case the request for modification addressing this impact area shall be documented;
  - effects of possible changes in the methods, tools or standards to be applied in the execution of the modification (compared to those which were applied for the development of the version of the software to be modified);
  - impact upon software itself, including a list of affected modules;
  - impact upon performance (including speed, accuracy, etc.);
  - strategy and necessary effort for verification and validation to ensure that the correctness of the existing software is maintained; the analysis of the software re-verification needed shall be documented in an auditable form;
  - the set of documents to be reviewed.

  The evaluation process may consist of a number of phases. The initial request may be reviewed for relevance and feasibility before any detail impact assessment work has been performed. When the full impact has been evaluated, a second more thorough evaluation may be performed. The software modification request is pending until the decision is made, which may be:
  - to reject the request; in this case, it is sent back with justification;
  - to require a detailed analysis, resulting in a software modification analysis report;
  - to accept and process the request.

Figure 4 presents an extended version of SafeScrum, which includes change impact analysis activities. As we see from figure 4, only changes that will lead to new changes in the SSRS have to go through a new change impact analysis. Changes due to e.g. errors found during testing can normally go directly to the product backlog.

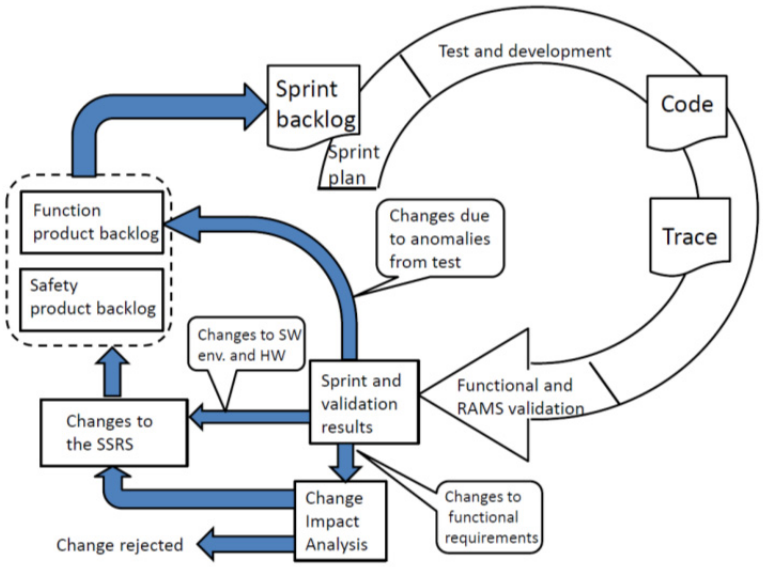

Figure 4 SafeScrum extended with the IEC 60880 change impact analysis

To get a better understanding of change impact analysis in IEC 60880, we have gone through "Software aspects of system modification" – section 11 of the standard – see figure 3. The following observations are important:

- 11.1.3 places a stricter requirement related to documentation than an agile approach like. However, the required level of documentation needed for each step is still up to the assessor. For a software modification to be considered, the following steps shall be followed:
    - generation of a software modification request;¨
    - evaluation of the request;
    - decision.
- 11.1.6 state that the analysis of the software re-verification needed shall be documented in an auditable form.
- 11.1.7: If a software modification analysis report has been required, this report shall be written by software personnel knowledgeable in the software of the system.
- 11.2.4: All the documents affected by the modification shall be corrected and refer to the identification of the software modification request.
- 11.2.5: A software modification report shall sum up all the actions made for modification purposes.
- 11.2.6: All these documents shall be dated, numbered and filed in the software modification control history for the project.

The standard's description of the process is quite detailed but none of the requirements will create problems for an agile approach. There are two challenges here: the number of required documents and the document-intensive process – e.g. 11.1.6 and 11.2.5. An agile process will however, results in fewer documents which implies that there are less documentation to maintain – e.g. 11.2.4. An agile approach will make the process more efficient, since the changes are done on the current code base, making it necessary to analyse, validate and verify less code. If we, in addition, use test driven development, we will already have all the necessary tests.

In our opinion, requirement 11.1.6 is satisfied through the change impact analysis report and the details from this requirement, shown in the bullet list below, can be used as a table of contents for the change impact analysis report. The same holds for 11.1.7 and 11.2.5. Requirements 11.2.4 and 11.2.6 are just general requirements for document handling and should be taken care of by a tool.

6. A tool prototype for change impact analysis

In this section, we present our ideas on how a prototype traceability tool called ToSS (Traceability of Safety Systems) could be used to support some of the traceability needs, in particular to perform change impact analysis, of SafeScrum. Earlier results related to this tool were reported in [6, 7, 8] and this report uses and refers to the work reported in the earlier publications.

6.1 Tool description

ToSS is a tool for traceability implementation during projects developing safety systems [6, 7, 8]. With the tool's functionality, stakeholders can represent artefacts, capture traces between artefacts in the form traceability graphs and perform different traceability analysis. The tool supports horizontal (same-level), vertical (between-level), forward, and backward traceability analyses.

ToSS implements the traceability approach called SaTrAp (Safety Traceability Approach) [6, 7, 8]. SaTrAp was developed as a part of the second author's on-going work on improving traceability for safety systems. The approach aims to provide traceability support for some of the tasks of the stakeholders, in particular the tasks of the safety analyst. The safety analyst could for example use traceability information to verify and validate whether all the safety requirements have been implemented. The approach consists of a process model and meta-models. A process model (see figure 5) is like a blueprint or a template describing a process that needs to be followed by the stakeholders for capturing traces during development process. The process model describes "what" kind of artefacts and relations should be captured and "when", i.e. at what levels, to capture them. The scope of SaTrAp's process model is ten abstraction levels starting from development of system concept to system installation. Meta-models categorise different types of artefacts and relations and define rules (syntax and semantics) for them.

There are two possible ways of using the ToSS tool. One is to use ToSS as standalone, where the stakeholders have to explicitly create/capture the traces. The other way is to integrate the ToSS tool with existing development environments (e.g. Jira) to enforce automated trace capturing. Once traces are captured (automatically or manually), the stakeholders can auto-extract traces using the traceability analyses.

6.2 Using ToSS to support SafeScrum – initial perspectives

ToSS has not been used in an agile setting. Here we provide our initial views on the possibility of using ToSS in projects using SafeScrum. As stated earlier, SaTrAp and therefore ToSS supports implementing traceability through ten abstraction levels.

The process model begins at the level "Development of system concept" (D1), where artefacts related to the system domain and environment are described. In the first stage of the safety assessment process - "Functional hazard & risk analysis" (S1) – possible hazards to system, risk associated with hazards, and identified hazard mitigations are documented. System-level requirements including functional safety requirements are described in D2 and system safety integrity levels are assigned during S2. Artefacts corresponding to system functions allocation, system architecture, and sub-system requirements are produced in D3 and D4. Separation, independency and diversity requirements are described in S3 and S4. D5 presents the sub-systems architecture and requirements of sub-systems' components - programmable electronic (PE) and non-PE component. In D5, safety sub-systems are considered while non-safety sub-systems are out of scope of our work. PE architecture and requirements of PE items – hardware

and software items – are described in D6. Integrity levels corresponding to components and PE items are identified and apportioned to respective components in S5 and S6 respectively. D7 and D8 produce artefacts related to hardware and software design and implementation activities. Results from verification and validation activities of hardware and software are contained in S7 and S8. Finally, results from sub-system (D9) and system (D10) integration and their respective verification and validation activities (S9 and S10) are specified.

However, some of these abstraction levels might not be applicable to SafeScrum, for example abstraction levels or artefacts relating to development of system/sub-system/PE/software architecture and detailed design might not be needed. In order, to use the tool in SafeScrum setting, the traceability process model and meta-models have to be tailored to fit with SafeScrum. Figure 5 presents parts of the process model with the most relevant abstraction levels applicable to SafeScrum.

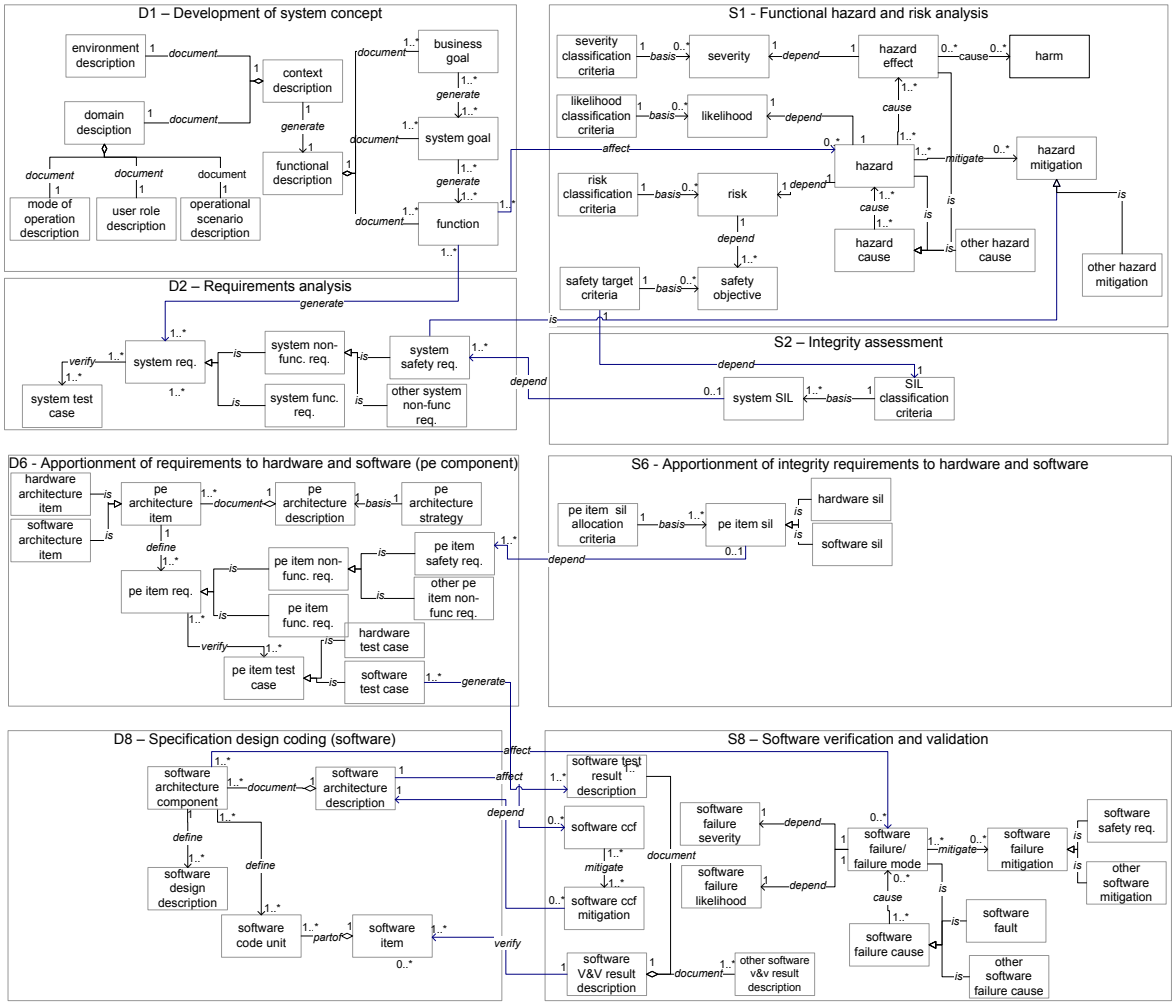

Figure 5 Applicable levels of traceability process model for SafeScrum

ToSS can support the capturing and tracing the following artefacts belonging to E/E/PE systems or software safety lifecycle phases during projects applying SafeScrum:

1. Environment description and the SSRS – i.e. concept description, overall scope definitions, hazard and risk analysis and overall safety requirements, functional and non-functional requirements.
2. Software modules, results from software safety analysis

3. Verification and test reports. ToSS does not automatically produce the reports. However, it should be possible to automatically generate a log or list with requirements with their respective test case descriptions and test results. The log or list could be used as part of the test reports.
4. ToSS will not generate the necessary change impact analysis reports and will only provide the necessary information. In addition. The tool does not yet adapt to agile development, which could be part of our future work.

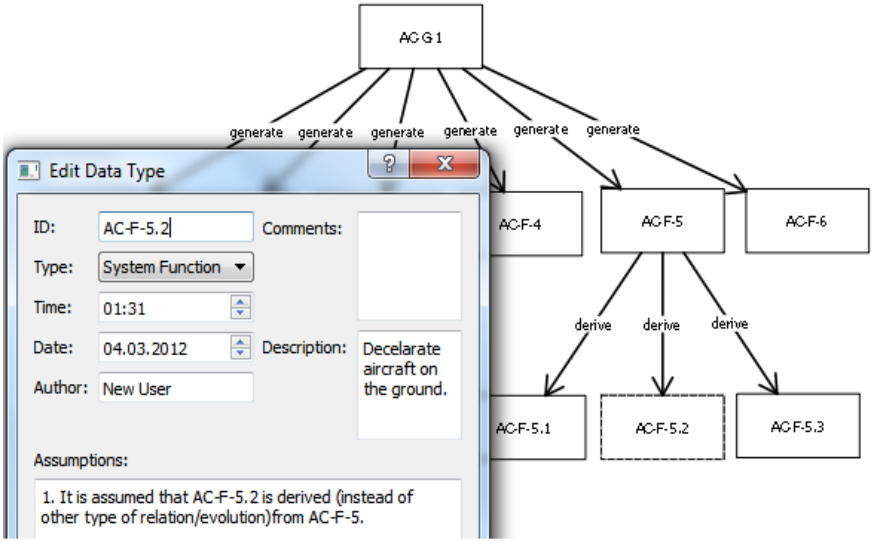

Figure 6 ToSS screenshot – capturing traces from goals to functions

In the current version of ToSS, the traceability information related to artefacts has to be manually inputted. Figure 6 is a screenshot from ToSS, which shows the traceability tree/graph with some of the artefacts at system concept level (D1 in process model). The figure also shows the attribute table for one of the functions. Attributes capture properties and additional information related to an artefact. Some of the supported attributes are: ID (unique identifier), author (who created or modified the artefact), date (when it is created or modified), description, and assumptions. User of the tool has to create a node in the traceability tree for each artefact and enter relevant information about the artefact in the attribute table. As shown in the figure, the traceability graph starts with creation of a hypothetical system goal AC-G-1 “The S18 aircraft is a four engine aircraft designed to carry 300 to 350 passengers...” System functions (features) are generated from the system goal, and therefore traced. Aircraft function AC-F-5 “Control aircraft on the ground” is further decomposed to lower-level functions.

6.3 Tool experiences

The approach and the tool so far have been applied on two desktop examples. The first application is on the fictitious S18 Aircraft example presented in ARP4761 guideline [9]. The second application is on the Air Traffic Management Remote Tower (RT) desktop example developed as a part of an ATM related project. RT was modelled and assessed for safety and security risks using a method called CHASSIS [10, 11]. SaTrAp meta-models were extended to include artefacts, such as textual misuse cases and failure sequence diagrams, used by CHASSIS.

Based on our experiences, ToSS was able to capture and extract traces to, among other things:

- estimate the impact of a change through development, safety assessment and security assessment
- trace and understand how safety requirements and security originated and the rational for their creation
- identify the interdependencies between artefacts, especially safety and security requirements
- provide evidence on the verification of safety requirements, i.e. safety requirements have been allocated and thereafter implemented by the components of the system
- demonstrate that the safety requirements reflect the results of the safety analysis, by providing evidence in the form of traces between the results from safety analysis and the identified safety requirements
- identify valid and complete set of evidences needed to justify safety claims in a safety case and to automatically generate and maintain parts of the safety case

7. Threats to validity

First and foremost, we have gone through all requirements for change impact analysis in the IEC 60880 standard. All the standard's requirements have been addressed and included in the proposed process – see section 5.2

The authors of this paper cover a wide range of relevant topics: software development for the nuclear industry, safety assessment of railway systems and software development for safety-critical systems. The processes suggested for change impact analysis in this paper is based on similar work done for the IEC 61508. The change impact analysis process for IEC 61508 has been presented to both our industrial partners and is currently tested in the SUSS project.

In our opinion, the main differences between the change impact analysis in IEC 61508 and IEC 60880 is the rigor and amount of information required in the change impact analysis report. Thus has been catered to in our proposed process.

Thus – in our opinion, the proposed process for change impact analysis is sound and compliant with the IEC 60880 standard.

8. Conclusions and Further work

In this paper we have presented a process to be used for change impact analysis when we are developing software compliant with IEC 60880 using an agile development process – in our case SafeScrum. Based on our discussion of threats to validity in section 7, it is our opinion that the described change impact analysis will satisfy the standard IEC 60880 and that the described tool – ToSS – will help us to handle the important challenge of traceability.

The next task will be to try out the ToSS tool in an agile setting to see how well it will handle frequent system changes and the change impact analysis needs for trace information. In addition, ToSS is built as a stand-alone tool. It will improve the tool's usability if it could be integrated with other tools such as DOORS and Jira.

9. References

[1] Li, B., Sun, X., Leung, H. and Zhang, S.: A survey of code-based change impact analysis techniques. Software Testing, Verification and Reliability (2013), Published online in Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/stvr.1475.

[2] Lehnert, S.: A review of software change impact analysis, Technical Report, Ilmenau University of Technology, Department of Software Systems / Process Informatics, Ilmenau, Germany, December, 2011.

[3] Acharya, M. and Robinson, B.: Practical change impact analysis based on static program slicing for industrial software systems, ICSE '11 Proceedings of the 33rd International Conference on Software Engineering, 2011.

[4] Kilpinen, M. S., Clarkson, P. J. and Eckert, C. M.: Change impact analysis at the interface of system and embedded software design, International design conference - design 2006, Dubrovnik, Croatia, May 15 - 18, 2006.

[5] Lindvall, M. and Sandahl, K., How well do experienced software developers predict software change?, Journal of Systems and Software, Vol. 43, No. 1, pp. 19-27, 1998.

[6] Katta, V., Stålhane, T. and Raspotnig, C.: Presenting a traceability based approach for safety argumentation. I: Safety, reliability and risk analysis: beyond the horizon. Proceedings of the European Safety and Reliability Conference, ESREL 2013, Amsterdam, the Netherlands, 29 September-2 October 2013. CRC Press 2014 ISBN 978-1-138-00123-7.

[7] Katta, V., Raspotnig, C., Karpati, P. and Stålhane, T.: Requirements management in a combined process for safety and security assessments. I: 2013 Eighth International Conference on Availability, Reliability and Security (ARES), Regensburg, 2-6 September 2013. IEEE Computer Society 2013 ISBN 978-0-7695-5008-4.

[8] Katta, V. and Stålhane, T.: Traceability of safety systems: approach, meta-model and tool support, Technical Report HWR-1053, OECD Halden Reactor Project, Institute for Energy Technology, 2013.

[9] Society of Automotive Engineers: ARP4761 - Guidelines and methods for conducting the safety assessment process on civil airborne systems and equipment, 1996.

[10] Raspotnig, C.: Requirements for safe and secure information systems, PhD thesis, University of Bergen 2014, ISBN 978-82-308-2992-9.

[11] Raspotnig, C., Karpati, P. and Katta, V.: CHASSIS Guide-line (draft), https://bora.uib.no/handle/1956/6172.

[12] Schwaber, K. and Beedle, M.: Agile software development with scrum, New Jersey: Prentice Hall, 2001.

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