Agile Change Impact Analysis of Safety Critical Software

Tor Stålhane[1], Geir K. Hanssen[2], Thor Myklebust[2], Børge Haugset[2]

[1] Norwegian University of Science & Technology, Trondheim, Norway
`tor.stalhane@idi.ntnu.no`

[2] SINTEF ICT, Trondheim Norway
`{ghanssen, thor.myklebust, borge.haugset }@sintef.no`

Abstract. Change Impact Analysis (CIA) is an important task for all who develops and maintains safety critical software. Many of the safety standards that are used in the development and use of systems with a certified safety integrity level (SIL) requires changes of such systems to be initiated by a CIA. The resulting CIA report will identify planned changes that may threaten the existing safety level. The challenge with CIA is that there are no practical guidelines on how to conduct and report such an analysis. This has led to a practice where most changes lead to extensive up-front analysis that may be costly and delay the change process itself. In this paper we propose a new strategy for CIA based on the principles of agile software development and the SafeScrum approach to establish a more efficient in-process impact analysis. We discuss several benefits of this approach, like resource savings, shorter time to initiate the change process, better prioritization and management of the change process, and others.

Keywords: Safety critical systems, agile software development, SafeScrum, change impact analysis, IEC61508

1 Introduction

Change impact analysis (CIA) is an important task for anybody who develops and maintains safety critical systems such as gas and fire detection systems, railway signaling systems and process control systems. Several standards and directives require that a CIA has to be done when a system with an approved safety integrity level is to be changed – e.g., IEC 61508 [1] and the EN 5012X series [2]. A CIA produces a CIA report (CIAR), which is an important input both to the development team implementing the changes and to the assessor who will approve the changes according to the relevant standards. Although several standards require a CIA to be performed there are no practical guidelines available. This is a major concern as change of complex software systems is a highly demanding task [3, 4] and even more so for safety critical systems. However, we need to strike a balance between what should be done – the standard's domain – and how it should be done, which to a large degree should be left to the development organization. We have provided guidance for CIA in a previous paper [5]. The key principle of our approach is to split the CIA into two phases. Phase 1 is performed for a group of changes when needed and before starting the change process. This is more efficient than the present ad-hoc practice where all changes from the systems requirements specification (SRS) are evaluated together. See our previous paper for details on this [ibid]. For phase 2 we suggest to perform the rest of the CIA as part of the development process itself. This process is described in this paper. We are motivated by the potential effect that lies in the principles of agile software development [6-9] of safety critical systems and believe that our previously described SafeScrum [10] method can be extended to facilitate an efficient in-process CIA – see section 2. The work on SafeScrum and the more recent development on agile CIA have been done as part of a four year Norwegian research project SUSS (Agile Development of Safety-critical Software).

The approach with a two phase agile CIA that is described in this paper is conceptual and not yet applied in industry. However, the author team has complimentary expertise in the domain with one expert on assessment of safety critical systems according to important standards like IEC 61508 and EN50128, one expert in development and evaluation of safety critical systems and two experts on agile software development and process improvement. We relate this work to our previous suggestion for an agile approach of developing safety critical systems [10] where development is done incrementally and iteratively, and where the management of requirements, assessment and impact analysis is done concurrently.

Our approach is based on extensive discussions in the collegium of experts, investigation of relevant literature and standards, and also through verification of ideas with leading industry partners developing SIL 2 and SIL 3 systems. There is, however, little difference between SIL 3 and SIL 4 when it comes to software. There is thus no reason why the approach should not work also for SIL 4 as well.

The key ideas promoted in this paper are that we provide practical expert guidelines on how to achieve a two-phase agile CIA process as well as on details on the SafeScrum process, and discuss expected savings from effectuating such a process. Our motivation is that there are no guidelines on how to perform and document a CIA at all, and that an agile approach will improve the current industrial practice.

The rest of the paper is organized as follows: Section 2 explains the background for CIA and gives a short summary of some of the relevant literature while Section 3 drafts an agile approach of developing safety critical systems. Section 4 provides the details on how to perform an *agile* CIA. Section 5 discusses some of the benefits we expect from this approach. Finally, section 6 concludes our work and provides directions for further work on these topics.

2 Background

Change Impact Analysis of safety critical systems

Development and evolution of safety critical software such as fire detection systems or ship controlling systems must comply with extensive safety standards and regulations in order to be approved for use. This also means that changes and extensions of such systems must undergo an assessment to update the certificate. For a system with an established SIL, planned changes to code and architecture are evaluated to see if the system will still meet the requirements specified in the standard after the changes. This is required by several standards, but there are no concrete guidelines on how to perform the analysis and how to document it in a CIAR.

The established practice is to perform the CIA upfront of the new development, have it accepted and then initiate the change and development process without any further CIA. This may represent a problem as the change process potentially can disclose problems with the planned change that the CIA didn't foresee. Also, doing the complete CIA upfront means that the change process cannot start until the analysis is fully completed. In a previous paper [5] we argue that it can be a good idea to perform the CIA in two phases; Phase 1 is performed upfront of development, similar to the common practice today but shorter in time and with less details. Phase 2 CIA is done as an integrated part of the development process itself.

Related work

A search for related work has shown several papers on the traceability problem related to CIA. Another topic of research that is published extensively is the effect of incremental changes e.g. in object-oriented development. We have also seen some publications on research on the effect of process change. However, we have seen no papers on the problems of change impact analysis in agile development of safety critical software apart from one of our own recent papers [11]. The closest is a quote from a paper by Jose Luis de la Vara and Rajwinder Kaur Panesar-Walawege on their meta-model SafetyMet [12] where they identify this topic as an area of future research.

B. Li et al. [13] has published the result of a survey, where they have identified 23 change impact analysis methods. Another survey, performed by S. Lehnert [14] 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 [15]. The authors 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. [16]. This paper discusses 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 [17] that "software engineers tend to perform impact analysis intuitively. Despite this common practice, 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 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, which emphasize thorough and detailed planning, and design upfront and consecutive plan conformance. We do not believe that an agile approach is appropriate for all steps needed when developing a safety-critical system. Thus, SafeScrum has come up with the idea of separation of concerns, as shown in figure 1, based on the process described in IEC 61508. Note that several requirements specified in IEC 61508-3, annex A (normative) and annex B (informative) will influence the Scrum process. We have also added an activity where trace information is collected

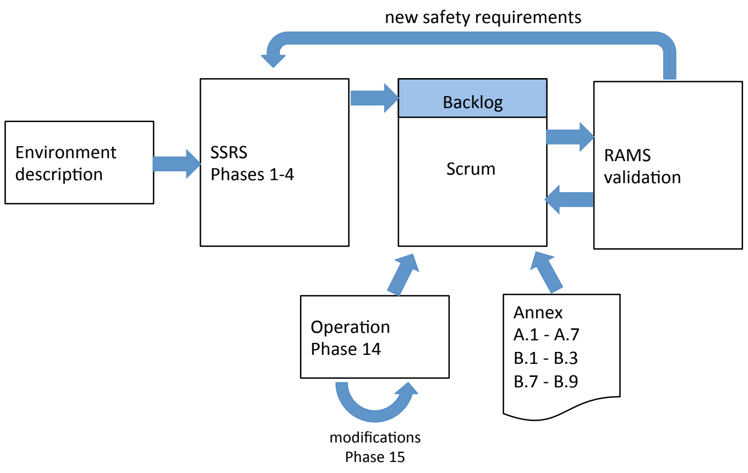

Figure 1 Separation of concerns

Several agile methods exist, whereof *extreme programming* [16] and *Scrum* [7] are the most commonly used. Figure 2 explains the basic concepts of SafeScrum.

The process starts with initial planning, which is short and results in a prioritized list of requirements for the system called *the product backlog.* Developers also *estimate* the implementation cost per backlog item. The following development is organized as a series for *sprints* (iterations) that each lasts a few weeks. Each sprint starts with sprint planning, followed by test and development, a sprint review and a retrospective. Typically, developers will apply the principles of *test-driven development* [13] where automated tests are developed *before* the code.

In the *sprint planning meeting,* the top items from the product backlog is transferred to the *sprint backlog* – adding up to the amount of resources available for the period. These requirements will be implemented in the following sprint. Each working day starts with a *scrum***,** which is a short meeting where each member of the development team (1) explains what she/he did the previous work day, (2) any impediments or problems that need to be solved and (3) planned work for the work day. Problems related to relevant safety standards should be discussed with the assessor as soon as possible after the meeting.

Each sprint *releases* an *increment* which is a running or demonstrable part of the final system. The increment is *demonstrated* for the customer(s), which will decide which backlog items that have been resolved and which that need further work. Based on the results from the demonstration the next sprint is planned. The product backlog is revised by the customer and is potentially changed / reprioritized. This initiates the sprint-planning meeting for the next sprint. When all product backlog items are resolved and / or all available resources are spent, the final product is released. Final tests can be run to ensure completeness.

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

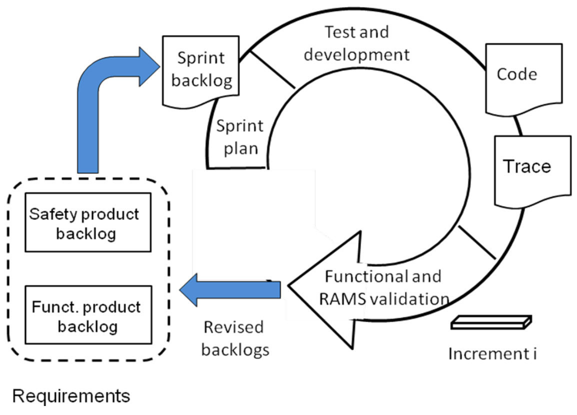

Figure 2 SafeScrum process model

Our model has three main parts. The first part consists of the IEC 61508 steps needed for developing the environment description and then the 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. This is the key input to the second part of the model, which is the Scrum process. The requirements are documented as *product backlog items.* A product backlog is a list of all functional and safety related system requirements, prioritized by the customer. We have observed that the safety requirements are quite stable (e.g. the response time has to be less than the Process safety time for a fire alarm system), while the functional requirements may change considerably over time. Development with a high probability of changes to requirements will favour an agile approach.

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. The safety requirements will come from three sources (1) applicable standards, (2) safety analysis – e.g. HazOp – and (3) from the system's customer. It is not necessary to have two physically separated backlogs – adding a tag to the safety product backlog items will suffice. 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. It can also be supported with an explanation of how the requirements are related if this is needed to fully understand a requirement. One of the participating companies includes the backlog and the necessary linking in a Jira tool. Using a tool like Jira enables us to adapt the process depending on whether a requirement is safety critical or not.

In order to be performed in an efficient manner, traceability requires the use of a supporting tool. There exist several process-support tools that can manage this type of traceability in addition to many other process support functions. One out of many examples is Jira plus RMsis.

To make Scrum conform to IEC 61508, the final validation in each iteration should be done both as a validation of the functional requirements and as a RAMS (Reliability, Availability, Maintainability, and Safety) validation, to address specific safety issues. If appropriate, the independent safety validator should take part in this validation for each sprint. If we discover deviations from the relevant standards, the assessor should be involved as quickly as possible as he is normally not involved in the validation for each sprint. Using an iterative and incremental approach means that the development project can be continuously *re-planned* based on the most recent experience with the growing product. This principle is related to the well-known principle of the Deming/Shewhart cycle [18]. Between the iterations, it is the duty of the customer or product owner to use the most recent experience to re-prioritize the product backlogs.

In addition to the re-planning mentioned above, applying the RAMS validation process to each increment will also give risk and hazard analyses a gradually evolving scope. This will improve the quality of these analyses. Even if the increments cannot be installed at the customer's site, they can still be tested and run as part of a system simulation. In addition, safety analysis performed on small increments can be more focused and thus give better results [19].

As the final step, when all the sprints are completed, a final RAMS validation will be done. Given that most of the developed system has been incrementally validated during the sprints, we expect the final RAMS validation to be less extensive than when using other development paradigms. This will also help us to reduce the time and cost needed for certification, enabling a shorter time to market. We also expect that it will be quicker and less expensive to perform updates to an existing system this way.

4 Agile change impact analysis

The proposed agile CIA-approach is organized as two phases. Phase 1 analysis is done before the change implementation process and resembles the present practice, but requires less effort and time because some analysis is postponed to phase 2. This also means that the change and development process will start earlier. See Myklebust et al. [5] for details on phase 1. In the following of this paper we look into the details of what we call phase 2 CIA.

The key principle of phase 2 is that the impact analysis is performed continuously and in synchronization with the SafeScrum development process. This is based on the same principle of simultaneity that justifies the idea of doing formal assessment as an integrated part of the SafeScrum development process itself [10]. In practical terms, this phase 2 in-process CIA is implemented as an extension to the SafeScrum process (see section 3) and more specifically, the sprint review meeting and the sprint-planning meeting. (See in-line references to figure 3 - ☆):

Sprint-planning meeting ①: Each sprint starts with a planning meeting where the team estimates, selects and details items from the product backlog and moves them to the sprint backlog to fill up the available resources (working days) for the upcoming sprint ②. With respect to the CIA we suggest that the team considers any effects the detailing of requirements and design decisions [20] might have for system's safety. The important question is: *will the requirement and design affect the safety?* A potential aid to use for decision support is Failure Mode and Effects Analysis (FMEA) [21]. Potential issues and un-clarities should be resolved immediately – either by reconsidering the requirement or the design. In case the requirement needs to be reconsidered, the product owner must be consulted. Alternatively, the design of the solution must be reconsidered. The product owner is not normally a part of the Scrum planning meeting so we suggest that the product owner is available, or that he can be contacted in cases where he needs to make decisions. One of our industrial partners uses the business manager and the manager of development here. All CIA issues that are raised and resolved during the planning meeting should also be documented in the CIAR ③.

Sprint-review meeting ④: After the completion of a sprint, the product owner joins the Scrum team to evaluate the outcome and results of the sprint. This involves approving or disapproving the recent sprint result ⑤, often based on a demonstration by the team. The product owner will revise the product backlog and decide whether there are items that should be removed, changed or re-prioritized ⑥. This is done using the most recent knowledge of the problem that is being solved and the solution that is being developed to solve the problem. For each change, the team will evaluate the impact of the change and consider whether it has an impact on the safety integrity level of the system. In cases where the team is not able to make this judgment they need to clarify this with relevant roles such as safety managers, product management, sales, and others – in general roles that are in positions to make a qualified judgment.

The dual backlog, which is an important concept in SafeScrum, will be an important aid to raise attention to changes that may affect the safety integrity level because of the potential relationships between functional requirements and safety requirements. If a functional requirement has a strong influence on one or more safety requirements we might consider moving it to the safety requirements backlog. Also, backlog items should be stated in the form of user stories explaining **who** the user role is, **what** the goal is (what the product owner wants to achieve), and **why** the user story is required (the rationale). This information is useful in order to evaluate the effect on the safety integrity level.

Identified issues need to be discussed either right away or in the following sprint review meeting, possibly also involving the product owner. This may result in further additions to the product backlog. Furthermore, all issues that are identified, and resolved, should be documented. This becomes important input to the CIAR 7, which grows incrementally in parallel with the system. We suggest that this should be the responsibility of the Scrum master. The updated CIAR becomes important information for the external assessor, which also is given a closer role in the development process [10].

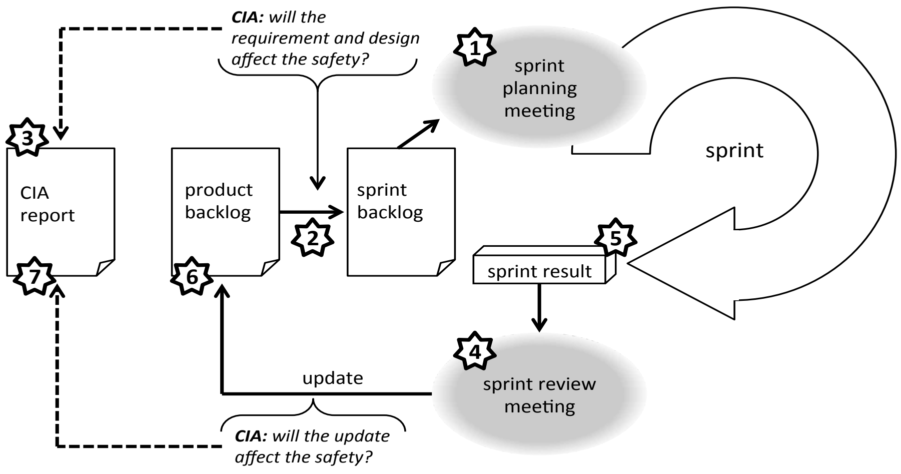

Figure 3 SafeScrum in-process (phase 2) change impact analysis

5 Discussion

To quote an anonymous developer: “After the first sprint, everything is maintenance of an existing system”. For the CIA we need to consider two types of changes: (1) changes to existing safety requirements and (2) changes that will influence code that directly or indirectly belongs to a safety requirement. All such changes can be categorized into one of two classes: simple and not simple. For simple changes we consider it to be enough if the developer categorizes the change as simple, perform a CIA and documement why he has made this decision. For the other cases, and especially those that concern safety requirements, we need to have a more elaborate CIA.

- Use trace information to see which parts of the code that will be affected
- Check the code for potential impact on safety in a code review
- Make a decision – change or not – and write a report. Note that the CIA reports will later be an important input for the assessor.

Changes to the safety requirements in the product backlog should always go through a CIA. We see several benefits of our two-phase CIA:

1) An in-process CIA is done when it is practical to do it, when the knowledge of the total system being affected by the change is as updated and detailed as possible [22]. This means that the impact analysis will be as complete as possible. This complements the CIA analysis from phase 1, which was based on preliminary information.
2) Because phase 1 is shorter than a traditional upfront CIA, the change and development process will start earlier and thus deliver results earlier. This is beneficial in cases where time to market is of great importance [23].
3) The CIA will inherit one of the key benefits of an agile process – better prioritization of changes due to the fact that decisions are being made at the latest possible time in the development process [22]. This also means that changes that were originally planned may be avoided if they are found to be unnecessary.
4) A less extensive CIA upfront may reduce the threshold for initiating a change process in the first place. The agile development process and the integrated CIA in phase 2 also gives the change/development project the opportunity to end the change process at an earlier time if necessary – this is a property of an agile development approach where design and requirements management may be adjusted during the course of the project. This can e.g. be a decision that is made by the product owner that sees that the (prioritized) changes that have been implemented so far are sufficient and that the product should enter the market. Thus, the change process becomes more flexible and controllable.
5) A classical CIAR is useful both to developers and to assessors, but we argue that the incremental CIAR will be even more valuable. It produces more updated information since it is made progressively during the system development, but also because the developers are continuously involved in the process and that they don't have to rely on a document that might be outdated.
6) A phase 2 approach like the one we have drafted will contribute to a more streamlined and synchronized process where impact analysis, development and assessment are done concurrently, instead of – as today – sequentially.

6 Conclusions and future work

The motivation for this paper is the need for a better way of analyzing and managing changes in a certified safety critical software system. Based on our knowledge of the industry we see a tendency to stick with traditional approaches of heavy upfront analysis. We also see that the obligatory standards that the industry needs to adhere to are weak on providing practical guidelines on *how* to perform the CIA. Thus, we have described how a modern software engineering process like Scrum can be adapted to improve the CIA and address some of the main limitations. We have also discussed what we expect will be the benefits and earnings from applying this approach.

One limitation of our work though is that the proposed agile CIA is currently being evaluated in an industrial context. However, the author team behind this paper has an extensive and complimentary expertise that is unique and we have chosen to develop and present our ideas here as an invitation to the academic community as well as the industry to consider ways of improving the CIA process. We would strongly encourage our peers to comment on our ideas and to try them out in practice.

To develop these ideas further we have started two pilot projects using the agile CIA approach in Norwegian industry[1]. The results of these empirical studies will be reported in following publications. Another important thread of activity is our engagement in the IEC 61508 standardization committee where one of the authors holds a position. Our goal is to encourage changes in the next revision of this important standard and to incorporate practical industrially proven guidelines like we have suggested in this paper.

Acknowledgements: This work was partially funded by the Norwegian research council under grant #228431 (the SUSS project). Research has been done in collaboration with Autronica Fire & Security AS.

7 References

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

[2] "EN 5012X series. Railway applications," ed.

[3] M. M. Lehman and J. F. Ramil, "Software evolution - Background, theory, practice," *Information Processing Letters,* vol. 88, p. 11, 2003.

[4] M. M. Lehman and J. F. Ramil, "An Approach to a Theory of Software Evolution," presented at the IWPSE, Vienna, Austria, 2001.

[5] 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.

[6] "Agile Manifesto," ed. http://www.agilemanifesto.org/, 2009.

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

[8] H. Takeuchi and I. Nonaka, "The New New Product Development Game," *Harward Buisiness Review,* 1986.

[9] T. Dingsoyr, S. Nerur, V. Balijepally, and N. B. Moe, "A decade of agile methodologies: Towards explaining agile software development," *Journal on Systems and Software,* vol. 85, pp. 1213–1221, 2012.

[1] http://www.sintef.no/safescrum

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

[11] 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, Hawaii, USA, 2014.

[12] J. L. de la Vara and R. K. Panesar-Walawege, "SafetyMet: A Metamodel for Safety Standards," in *Model-Driven Engineering Languages and Systems*. vol. 8107, A. e. a. Moreira, Ed., ed Berlin Heidelberg: Springer Verlag, 2013, pp. 69-86.

[13] B. Li, X. Sun, h. Leung, and S. Zhang, "A survey of code-based change impact analysis techniques," *Software Testing, Verification and Reliability,* vol. 23, pp. 613-646, 2012.

[14] S. Lehnert, "A Review of Software Change Impact Analysis," Ilmenau University of Technology, Department of Software Systems / Process Informatics, Germany2011.

[15] M. Acharya and B. Robinson, "Practical change impact analysis based on static program slicing for industrial software systems," presented at the 33rd International Conference on Software Engineering (ICSE'11), Honolulu, USA, 2011.

[16] M. S. Kilpinen, P. J. Clarkson, and C. M. Eckert, "Change Impact Analysis at the Interface of System and Embedded Software Design," presented at the International Design Conference, Dubrovnik, 2006.

[17] M. Lindvall and K. Sandahl, "How Well do Experienced Software Developers Predict Software Change?," *Journal on Systems and Software,* vol. 43, pp. 19-27, 1998.

[18] W. E. Deming, *Out of the Crisis*. Cambridge: The MIT Press, 2000.

[19] M. Vuori, "Agile Development of Safety-Critical Software.pdf," Tampere University2011.

[20] J. Armitage, "Are agile methods good for design?," *Interactions* vol. 11, pp. 14-23, 2004.

[21] IEC, "60812: Analysis techniques for system reliability - Procedure for failure mode and effects analysis (FMEA), 2nd ed.," ed, 2006.

[22] M. Poppendieck and T. Poppendieck, *Lean Software Development: An Agile Toolkit for Software Development Managers*. New Jersey: Addison Wesley, 2003.

[23] R. Baskerville, B. Ramesh, L. Levine, J. Pries-Heje, and S. Slaughter, "Is "Internet-speed" software development different?," *IEEE Software,* vol. 20, pp. 70-77, 2003.