The Application of SafeScrum to IEC 61508-Certifiable Software

by Tor Stålhane, Geir K. Hanssen and Thor Myklebust

Introduction

There is a clear trend for safety-critical systems in the offshore and process industry for functionality to be moved from hardware to software. The reasons for this are increasing hardware performance and an increasing need for flexibility, like being able to rapidly apply new technology and new requirements. Standard hardware components can now be programmed, meaning less effort on hardware development and more on software development. This means a higher tolerance to changes in requirements and design during the development process. Large companies, such as ABB and Autronica Fire & Security, have products in which nearly 100% of the development costs are related to software development. A direct consequence of this is larger and more complex software development projects. Modern safety-critical software systems face the same extensive safety requirements as hardware-based systems, which again leads to a need for more efficient and flexible software development methods. The common practice today when developing safety-critical software systems is to use a plan-based and document-driven development process, which leads to inflexibility in requirements change as well as large costs for producing documentation to manage the certification process. An initial estimate indicates documentation-related costs of between 25% and 50% of the total development costs.

We claim that traditional plan-driven approaches, which are commonly used in the safety domain, do not match the increasing need for flexibility. We thus propose a new approach, called SafeScrum, for agile development of safety-critical software systems. In this, the first article in a two-part series, we will look into IEC 61508-3: 2010 and a handful of issues we have identified with respect to the applicability of Scrum [4], one of the most used agile software development (ASD) methods in the software industry today. We will end this first part with a basic introduction to agile software development. In the second part we will show how we can adapt Scrum so that it can be used to develop software in a way – *SafeScrum* – that is compliant with IEC 61508-3: 2010.

Looking into IEC 61508

The IEC 61508 standard, 'Functional safety of electrical, electronic and programmable electronic (E/E/PE) safety-related systems', is an international standard for guiding development and assessment of safety-critical systems. The development of this standard started back in 1985 when IEC set up a task group to assess the viability of developing a generic standard for programmable electronic systems (PES), which led to setting up a working group to define a new standard. The motivation for initiating this activity was that even if a PES was implementing relatively simple safety functions, its level of complexity was significantly greater than that of the hardwired systems that had traditionally been used [2].

Continued on Page 15

An Introduction to ISO26262: Part 3

Continued from Page 13

opportunity to make change. Hence the importance (and requirement) of planning and resourcing these activities in phase with the development, and with the necessary competencies and appropriate data.

The next article will look at the development of software and the steps needed for integration of the E/E system.

Dave Higham is Head of Functional Systems at Delphi Diesel Systems. He may be contacted at:
<Dave.Higham@delphi.com>

The Application of SafeScrum to IEC 61508-Certifiable Software

Continued from Page 14

The overall strategy of IEC 61508 is to control conformance to a sequential process by producing extensive proof of conformance to an external certification body. This standard is widely used and is considered a prerequisite by both the clients' and the suppliers' side. Although fulfilling its purpose, the standard requires extensive effort and time, resulting in 1) inflexibility in the development process, and 2) high documentation costs.

In order to investigate whether principles from modern software development processes, such as agile software development and, in particular, Scrum can be applied to increase the flexibility and reduce the documentation costs, we did a systematic analysis of the requirements in Part 3 (Software) of the standard. Each section of Part 3 was evaluated, asking, 'Will we fulfil this requirement if we use the Scrum process?' This was done by a team of three experts: an expert on safety and software engineering, a certified safety assessor, and an expert on software engineering and agile development. Forty-nine issues needed further investigation, eventually leading to 15 where we need adaptations or flexibility in order to make the process acceptable to both the Scrum team and to the safety assessors. An overview of these adaptations, referring to the relevant clauses in Part 3 of the standard, is:

- 7.1 – How to structure the development of the software: 2 out of 9 requirements;
- 7.3 – How to develop a plan for validating the software safety: 2 out of 25 requirements;
- 7.4.2 – How to create, review, select, design and ensure the safety of the system: 9 out of 50 requirements;
- 7.4.7 – Requirements for software module testing: 1 out of 4 requirements;
- 7.9 – How to test and evaluate the outputs from a software safety lifecycle: 1 out of 95 requirements.

Thus, we have to deal with 15 IEC 61508-3: 2010 requirements where Scrum and the assessor have to show some flexibility. These requirements are mostly related to documentation and planning.

Agile Software Development (ASD) in a Nutshell

ASD is a way of organizing the development process, emphasizing direct and frequent communication, frequent deliveries of working software increments, short iterations, an active customer engagement throughout the whole development lifecycle, and change responsiveness rather than change avoidance. This can be seen as a contrast to waterfall-like models, which emphasize thorough and detailed planning, design upfront, and conformance to consecutive stages of the plan. Several agile methods are in use, whereof extreme programming [1] and Scrum [4] are the two most commonly used. Figure 2 explains the basic concepts of an agile development model. Two concepts are central when understanding the Scrum process – sprint and backlog.

- A sprint is a time box where a part of the code is developed to form a set of work items in the sprint backlog. Each sprint builds a part of the system and this part is integrated with the previous parts at the end of the sprint. In this way, the system (product) is built through a stepwise building process.
- A backlog is a store of jobs waiting to be done. Each job has a cost (usually person-hours) and a priority. The priority may and will be changed during the project. If errors are discovered, the problem report, together with a requirement for correction, goes back into the backlog. If an error is discovered in an already integrated part, a requirement for correction of this part goes back into the backlog, with a cost estimate and a priority, which depends on how important the correction is as compared to the other items already in the backlog. There are two types of backlog – project and sprint. The former lists all the jobs to be done. A sprint backlog contains those jobs to be carried out in a

Continued on Page 16

The Application of SafeScrum to IEC 61508-Certifiable Software

Continued from Page 15

particular 'sprint' (see below), or release.

Each backlog item (project and sprint) contains a requirement, a priority and a cost – denoted by R, P and C respectively (see Figure 1). The cost is usually decided via planning poker – a process similar to B. Boehm's version of the Delphi process.

We need Cn + Cx + Cy +Ca to be less than or equal to the amount of resources available for the next sprint. The total amount of available resources (person hours) is the sum of available person hours from the members of the development team in the next sprint and is known up-front.

The main constructs of the Scrum process are:

- Initial planning is short and results in a prioritized list of requirements for the system. The requirements are inserted into the system (project) backlog – one item per requirement. Developers also make a time estimate for each item.
- Development is organized as a series of sprints (iterations) and each sprint usually lasts one to four weeks. Typically, developers will use test-driven development [3] where automated tests are developed before the code. A work item in a sprint may be: development of new code based on customer requirements, or redevelopment and error correction of an item that was not accepted by the customer in the previous release.
- Each sprint starts with a sprint planning meeting where the top items from the project backlog are moved to the sprint backlog – adding up to the amount of resources available for the sprint. These requirements will be implemented in the sprint being planned.
- Each working day starts with a scrum, which is a short meeting at which (1) each member of the development team explains what she/he did the previous work day, (2) any impediments or problems that need to be solved are addressed, and (3) work is planned for the current work day.
- Each sprint releases an increment (executable code) which is a fully functional, or in other ways demonstrable, part, of the final system (e.g. a piece of software that runs on a simulator). This increment consists of code developed in the previous sprints plus the increment developed in this

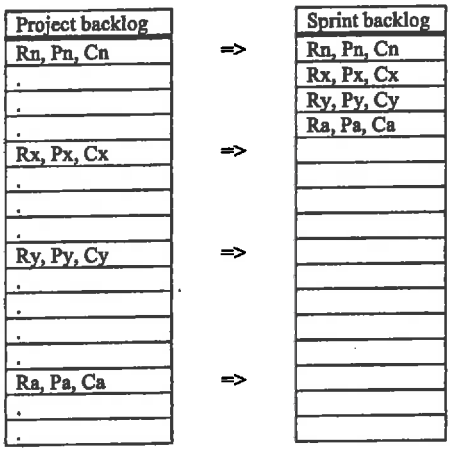

Figure 1: Project and Sprint Backlog Items

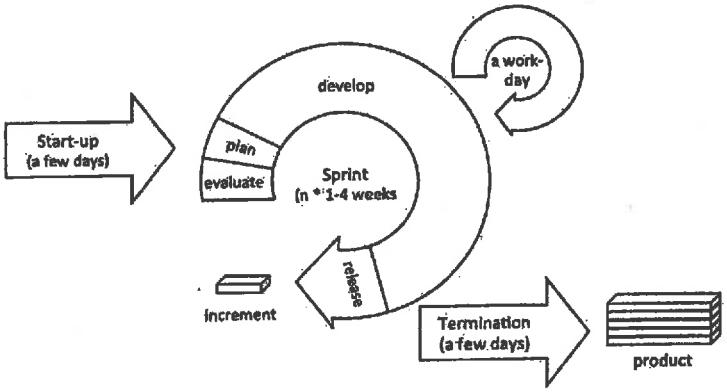

Figure 2: The Basic Agile Software Development Model

Continued on Page 17

The Application of SafeScrum to IEC 61508-Certifiable Software

Continued from Page 16

sprint.

The increment is demonstrated for the customer(s), who decides which backlog items have been resolved and which will need further work. Based on the results from the demonstration, the next sprint is planned. The project backlog is revised by the customer and is potentially changed/reprioritized. This initiates the sprint-planning meeting for the next sprint. When all project backlog items are resolved and/or all available resources are spent, the final product is released. Final tests – e.g. a factory acceptance test – will be run to ensure completeness and correctness.

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

The Need for a Revised IEC 61508 Standard

We will, in the next and final part of this article, present a Scrum version that is adapted to development of safety-critical software. However, this is just a stop-gap action. We need a revision of IEC 61508-3: 2010 in order to accommodate agile development processes and all the new development processes that will emerge in the future. We see two possible roads ahead: (1) a goal-based standard, or (2) include the agile development paradigm in the standard. While the first alternative is the most promising in the long term, the latter alternative will require less change and is thus the simpler option. We will thus only discuss this alternative. Our most important suggestions are:

- Open up the standard to iterative development. This has already been done for safety-critical software in the nuclear domain, see IEC 60880. It is a simple change, which would accommodate not only Scrum but also many other ways of working, such as evolutionary development.
- Increased involvement of the assessor during development. This would enable the project participants to get a clear idea of which documents they need to develop for proof of compliance (PoC) and in which cases a simple PoC documentation is sufficient. A related concept – assurance driven development – is under development by Det norske Veritas.
- Advise development organizations on how to develop reusable documents – what can be reused and what needs to be developed separately for each new project. This should be an informative annex to the standard.

In the second part of this article, we will present SafeScrum. This is an IEC 61508 adapted version of the agile process presented in this part. It is also the process that we will try out in two pilot projects in Norwegian industry during the next four years. Stay tuned. This will be interesting.

References

[1] Beck, K. and Andres, C., Extreme programming explained: embrace change (2nd Edition). 2004, Boston: Addison-Wesley Professional.

[2] Bell, R. Introduction to IEC 61508. In proceedings of 10th Australian workshop on Safety critical systems and software. 2005. Darlinghurst, Australia: ACM.

[3] Koskella, L., Test Driven. 2008, Greenwich, UK: Manning.

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

Prof. Tor Stålhane is at the Norwegian University of Science and Technology, Dr. Geir K. Hanssen is a Senior Research Scientist at SINTEF ICT, and Thor Myklebust is an Assessor at SINTEF ICT. They may be contacted at:

<stalhane@idi.ntnu.no>, <Geir.K.Hanssen@sintef.no> and <Thor.Myklebust@sintef.no>