Challenges in Flexible Safety-Critical Software Development -- An Industrial Qualitative Survey

Holmén Notander, Jesper; Höst, Martin; Runeson, Per

Published in:
Lecture Notes in Computer Science/Product-Focused Software Process Improvement

DOI:
10.1007/978-3-642-39259-7_23

2013

Document Version:
Peer reviewed version (aka post-print)

Published in proceedings of 14:th International Conference of Product Focused Software Development and Process Improvement (PROFES), Paphos, Cyprus, 12-14 June 2013.
The final publication is available at link.springer.com

Challenges in Flexible Safety-Critical Software Development – An Industrial Qualitative Survey

Jesper Pedersen Notander, Martin Höst, and Per Runeson

Software Engineering Research Group, Dept. of Computer Science,
Lund University, Sweden
{jesper.pedersen_notander,martin.host,per.runeson}@cs.lth.se

Abstract. Context. Development of safety-critical systems is mostly governed by process-heavy paradigms, while increasing demands on flexibility and agility also reach this domain. **Objectives.** We wanted to explore in more detail the industrial needs and challenges when facing this trend. **Method.** We launched a qualitative survey, interviewing engineers from four companies in four different industry domains. **Results.** The survey identifies human factors (skills, experience, and attitudes) being key in safety-critical systems development, as well as good documentation. Certification cost is related to change frequency, which is limiting flexibility. Component reuse and iterative processes were found to increase adaptability to changing customer needs. **Conclusions.** We conclude that agile development and flexibility may co-exist with safety-critical software development, although there are specific challenges to address.

Keywords: Software Engineering, Qualitative Survey, Safety-Critical Software

1 Introduction

The development of safety-critical systems is traditionally governed by document-driven, process-heavy paradigms. Safety standards, such as IEC61508 for automation and EN50126 for railways, assume extensive documentation and strictly defined processes for the product safety certification, including risk analysis, change control and traceability. Consequently, the pace of change is lower in this type of systems, making them less flexible with respect to changing requirements from customers and markets.

In her book "Engineering a Safer World" [8], Leveson identifies several types of changes to the safety-critical systems we build, for example, fast pace of technological change, increasing complexity and coupling, and complex relationships between humans and automation. She concludes that there is a need for a paradigm change in the development to achieve safer systems, which she then proposes in the book. While we share the general description of the changes, we see a need for a more systematic exploration of the industrial context and needs in the development of safety-critical systems in different industry domains.

Therefore, we launched a qualitative survey on industry practices and problems. We particularly focus on how the system development and safety certification processes relate to each other, when developing safety-critical systems. A viewpoint of particular interest is the ability to support flexibility in the development to meet changing customer needs and technological changes. We interviewed five safety and software engineers from four companies in different industries: Aerospace, Automation, Robotics and Transportation. The interviews were transcribed, coded, and qualitatively analyzed. We conclude from the study that human factors and the quality of requirements are as essential for development of safety critical systems, as they are for non-critical systems. In addition, we observe that the cost of certification is proportional to the number of releases, thus creating an incentive for few releases with many changes, which adversely affects flexibility. Furthermore, component reuse and iterative processes were found to increase adaptability to changing customer needs.

The paper is outlined as follows. We define the problem background and related work in Section 2. In Section 3 we present the methodology used for the study, including a characterization of the studied companies. In Section 4 we present and discuss the resulting findings. Section 5 concludes the paper.

2 Background and Related Work

Developers of safety-critical systems are increasingly using software to implement system functions, both safety-critical and non-critical. The inherent flexibility of software, enables system developers to rapidly adapt to changes in customer and market needs, without paying the high costs associated with hardware. It also supports reuse of existing functionality in new products as well as evolution of existing solutions.

Although increasingly used in safety-critical systems, the extent to which software is adopted by industry differs between different domains. In some domains, software has been used extensively for many years in safety-critical parts, e.g. Aerospace, whereas in other domains it has primarily been used for non-safety critical parts, e.g. Robotics. The trend, however, clearly points towards using more software, with increasing complexity, in safety-critical functions.

Safety-critical systems have the potential of causing harm to people and the environment [8]. *Safety-critical software* is used in a safety-critical system to realize a safety-critical function, e.g. the flight control algorithm in an airplane. Software in itself cannot harm people or the environment but unanticipated behavior of the software, either resulting from faults in the requirements specification, failure to implement requirements according to specification or not following operation requirements, can propagate into the physical world and there cause harm.

Safety-critical systems are regulated systems in the sense that system developers are mandated by law or strongly recommended to show compliance with an applicable standard. For example, in the European Union industrial robots need to show compliance with the Machinery Directive by following the ISO10218

standard. Several industries have their own standards that are applicable for specific systems, e.g. medical devices, aircraft systems, railway systems etc. Some standards are general in scope and apply to a broader range of systems, e.g. ISO61508 that covers electric, electronic and programmable electronic safety-related systems, whereas other standards are industry or even system specific, e.g. ISO10218 that only applies to robots and robotic devices.

Standards can be classified, for instance by their scope, generic vs. domain-specific as mentioned earlier, but also whether they are *means-prescriptive* or *objective-prescriptive*. A means-prescriptive standard, e.g. ISO61508, focuses on the *means* to achieve certain high-level safety goals and typically provides lists of methods and suggestions that a developer would be recommended or forced to include in their development process. An objective-prescriptive standard on the other hand, e.g. RTCA/DO-178, defines (low-level) objectives that should be reached, but does not necessarily define how to reach them. High-level safety goals are achieved when the objectives are fulfilled.

Common for all standards regarding safety-critical software, which are applicable in the context of this study, are that they consider software as a deterministic artifact, whose failures can only be caused by residual specification, design or implementation faults. Thus, safety is assured through the application of a standard-dependent design assurance process consisting of process-based and product-based development activities [1].

Although we do not claim to have done an extensive literature study, it would seem that there is a lack of empirical research in safety-critical software development, which investigates the flexibility aspect of safety-critical software from a holistic point of view. Land *et al.* [7], investigated component reuse in safety-critical systems through an industrial case study as well as action research. They identified challenges of component reuse with the aim of getting an overall picture of safety-critical development. They addressed flexibility by means of component reuse and provide a broad picture of challenges, including component interfaces and abstraction, traceability and certification, but they do not address the development process as such.

McHugh *et al.* [9] address flexibility from a process perspective, basing their work on an industrial survey, in which they investigated barriers preventing the adoption of agile practices in safety-critical medical device development. They found that the barriers they identified were tightly coupled with current regulatory constraints on medical software, but they emphasized that the barriers were not insurmountable even with todays best practices and standards. Kornecki and Zalewski [6] approach flexibility through an industrial survey of software tool support in the development and verification of safety-critical systems. Due to the increasing use of software tools in modern, highly complex, safety-critical systems development, the authors wanted to identify issues and concerns in software tool qualification and certification.

There exist several papers that present insights, into safety-critical development and related areas, which were gained from analytical and design research performed as academic case studies, sometimes basing the case on real world sit-

Table 1. Research steps

Research step	Result
Definition of research questions	3 questions
Planning of interviews	9 main questions
Conducting interviews	$\sim 5 \times 1$ h interview recordings Interview transcripts
Definition of codes	19 initial codes
Coding	Coded transcripts, 13 additional codes
Analysis of coded transcripts	Reduced code set to 27 codes, 7 clusters of related codes, additional areas of interest
Summary of areas of interest	Identified themes among clusters, Thematic conclusions

uations with industrial data but in equal amounts on fabricated, but plausible, data. For instance, Hawkings *et al.* [4] present a safety argument pattern catalog, which was evaluated in two case studies. The two cases were based on industrial data from two real products. The presented pattern catalog is an appealing way of aiding the construction of safety cases in a repeatable and consistent way, which might be of benefit in a situation were safety-critical systems are composed of flexible components.

3 Research Method

In this section we present the research method used in the study, the case companies and the interviewees. We also discuss the validity of the study in relation to the used method and selected companies.

The research is based on a qualitative survey using interviews for data collection [3]. The methodology in a qualitative survey resembles that of a multiple case study [11, 5], but the cases are not studied in depth in the survey. The objective of the research was to identify potential conflicts between safety certification procedures and more flexible methods for composition and development of software intensive systems. A significant part was to understand todays practices. By *composition* we mean integration and configuration of reusable components with the specific aim of creating variants of a system, whereas by *development* we mean the activities that is undertaken to create the components and the system design.

Because qualitative methods were used, our conclusions are based on an *understanding* of the collected data, created through a structured and systematic reading-process. The different steps in the research study, which are based on the guidelines of Runeson *et al.* [11], are summarized in Table 1, and described in more detail below.

3.1 Definition of research questions

Based on the objective, the following research questions were defined, which were recorded in a case study protocol:

RQ1 What are the challenges related to flexibility and safety in *software development* for software-intensive safety-critical systems, with respect to *agile practices*?
RQ2 What are the challenges related to flexibility and safety in *system composition* for software-intensive safety-critical systems, with respect to *safety certification*?
RQ3 What is the role of *system and software architectures* with respect to *flexibility and safety* in software-intensive safety-critical systems development?

3.2 Planning of interviews

Based on the research questions, a set of interview questions were derived. The following top-level questions were defined:

IQ1 What is your role at the company, particular in relation to the certification process?
IQ2 How much experience do you have with the current certification process?
IQ3 Could you describe the product and outline the main challenges regarding the safety certification of it?
IQ4 Could you give a brief description of the certification process in general and for software in particular?
IQ5 How does the certification process impact the development process?
IQ6 What is the role of the system and software architecture in the certification process?
IQ7 What are the main concerns with the current certification process?
IQ8 What are the main driving forces behind the introduction of more flexibility?
IQ9 What is the next step towards more flexibility in the certification process?

We also derived, for most of the main questions, 3–4 additional sub-questions, which have more explicit connection to the three research questions (RQ1–3). These are not presented in detail in this paper due to space constraints.

3.3 Conducting interviews

Interview candidates were selected with the aim of acquiring a diversified view from different industries and companies, see Section 3.6, as well as different roles, see Table 2. The identification and selection of interviewees was an ongoing process throughout the study.

The interviews were semi-structured, which means that open ended questions were asked about specific areas [11, Chapter 4]. Questions were not necessarily asked in the same order as presented in Section 3.2 and tended to be more

Table 2. Summary of interviewee characteristics.

ID	Role	Experience	Responsibilities
I1	System Architect	18 years, 5 with safety	Safety related issues at a system level.
I2	Safety Manager	30 years, 4-5 in current role	Coordinates the company safety and certification activities.
I3	Software Architect	14 years	Manages the development of a software based safety framework as well as working as the project liaison to I2.
I5	Safety Manager	10 years, 5 in current role	Responsible for the implementing, evolving and enforcing the software development process at of the company.
I4	Safety Manager	10 years, 5 in current role	Manages certification activities and contact with the certification authorities.

open at the end of the interviews. Interviewees were not interrupted when their answers diverged from the asked question, as long as the answer was within the scope of the study. The researchers kept track of the interviewees answers and tried to take that into account when additional questions were asked, so as not to ask a question that had already been covered.

In total, five interviews were held with interviewees from four different companies. The five interviewees could roughly be classified into three roles: safety managers, system architects and software architects. All interviews were conducted with two researchers each except for interview I5, and were recorded and later transcribed for further analysis.

3.4 Analysis

Definition of codes. Initially, a set of codes was defined based on the interview questions, the study objective and the knowledge that was gained during the interviews and from the transcription process. A set of 19 codes were found, see Column S_{ci} in Table 3.

Coding. The coding was done in two steps. In the first, the interview transcripts were compiled into a list of codeable text segments corresponding roughly to a paragraph in the source transcript. In the second step, each text segment was assigned a set of codes, i.e. a segment could be given any number of codes ($\leq 19 + 13 = 32$). The maximum number of codes assigned to a segment was 7, the minimum 1 and the median 2. The coding was conducted by two of the researchers. One researcher coded three transcripts and the other two.

While coding the two first interview transcripts, 13 additional codes were defined, as reported in Column S_{ce} in Table 3. After coding, non-used codes were removed, as well as specialized codes that were subsets of other codes. The final set contains 27 codes, see Column S_{cf} in Table 3.

Table 3. Code sets during the stages of the analysis, including the code clusters. An x means the code is member of the set, r and c indicate whether a code was in the row or column cluster set in the "heat map", b if in both.

Codes		Code Sets				Code Clusters						
Id	Name	S_{ci}	S_{ce}	S_{cf}	Substitute	C_1	C_2	C_3	C_4	C_5	C_6	C_7
C1	Certification Process	x	x	x						c	b	c
C2	Development Process	x	x	x			r			c	c	b
C3	Safety-Critical	x	x	x		c		r				
C4	Standard	x	x	x						c	b	c
C5	Stakeholders	x	x	x						c	b	
C6	Safety Analysis	x	x	x				r	b	r		
C7	Variants	x	x	x				b				
C8	System	x	x	x				b		r		
C9	Component	x	x		C14							
C10	Software	x	x	x				r	b	r		
C11	Challenges	x	x	x		c	r					r
C12	Activity	x	x	x			b					r
C13	Artifact	x	x	x				b			r	
C14	Architecture & Design	x	x	x				b		r		
C15	Safety	x	x	x		c		r				
C16	Tools & Methods	x	x	x				b				
C17	Composition	x	x		C7							
C18	Flexibility	x	x	x				r	b	r		
C19	Future	x	x		C18							
C20	Economy		x	x		b	r					r
C21	Quality		x		None							
C22	Verification & Validation		x	x			b					r
C23	Skills & Experience		x	x		b						r
C24	Safety Case		x	x				r	b	r		
C25	Requirements		x	x			b	c				r
C26	Safety Awareness		x	x		r						r
C27	Company Culture		x	x		b						r
C28	Hardware		x	x				r	b			
C29	COTS & OSS		x	x				r	b		c	
C30	Late Fault Identification		x		C11, C22							
C31	Legal Responsibility		x	x							b	
C32	Roles		x	x			r				b	
	Count:	19	32	27	Unique pairs:	18	15	63	15	24	27	26

Analysis of coded transcripts. When all transcripts were coded, we began searching for code patterns, i.e. themes, in the coded transcripts. Although, we already had an impressions of which code patterns were more frequent than others, we sought a more systematic approach.

To guide our reading, we created an overview of how different codes were related and to what degree, by calculating a relative *overlap* score for each

possible pairs of codes, defined for any two codes A and B as:

$$\text{overlap}(A, B) = \frac{\#\text{ text segments with both code } A \text{ and code } B}{\#\text{ text segments with code } A}$$

The score is a measure of the overlap that describes the number of occurrences of the pair (A, B) relative the total number of occurrences of code A. An overlap score of one means that code A is a true subset of B, i.e. A occurs only together with B.

We visualized the overlap scores for all possible code pairs in a "heat map", and identified clusters of pairs, i.e. groups with high pair-wise scores, using hierarchical clustering and manual inspection. In total, 7 clusters were identified, see Table 3. The clusters were used to select text segments for further analysis. A segment was selected if it was coded with a pair belonging to the cluster or if it had only one code and that code was in a pair belonging to the cluster.

By reading the segments thus selected, in-depth and in their context, the aim was to identify themes for each cluster. In total only four themes were found, partially due to content overlap between clusters and that some clusters were lacking in cohesiveness.

Summary of areas of interest. In this last step of the analysis the result of the previous step was summarized for each area of interest by contrasting different interviewees' statements. The result of this step is presented in Section 4.

3.5 Analysis of threats to validity

The validity of the research is discussed based on the design presented above, based on Runeson *et al* [11].

Construct validity. The construct validity concerns how well the researchers and the interviewees are able to communicate the real underlying phenomena under study. There is a risk of being misunderstood, e.g. if the interviewees did not have the same construct in mind as we researchers when talking about terms like 'safety', 'security', 'flexibility', etc. However, during the interviews we were aware of this risk, and much of the purpose of the interviews was to understand what the interviewed people and organizations mean with these constructs, and thus the threat is reduced.

Internal Validity. The internal validity threat is mainly concerned with causal relationships. Since this study is primarily descriptive, these threats are not applicable.

External Validity. The external validity threat concerns to what extent the results are valid in other contexts than the cases that were studied in the presented research. Since the study is carried out in a limited set of organizations it is not possible to widely generalize to other organizations without considering the differences. However, the selected organizations represent different domains that use different safety standards. In addition to this, the interviewees represent different roles in their organizations. Thus, the findings must be confirmed in larger studies, before generalizing them.

Table 4. Summary of company characteristics.

ID	Domain	Standard(s)	Product
C1	Robotics	ISO10218	Industrial robots and service robots. Off-the-shelf components for system integrators.
C2	Transportation	EN50128	Railway signal systems. System and component provider. Mainly government customers.
C3	Automation	IEC61508	Process controllers. Off-the-shelf components for system integrators.
C4	Aerospace	RTCA/DO-178B	Aircraft systems. End-user products. Government customers.

Reliability. This concerns how the analysis and the results depend on the researchers. We have followed strict protocols for the conduct and analysis, which are openly reported. Additionally, all findings that were derived by one of the researchers were reviewed by the other researchers, which we argue limits the reliability threat.

3.6 Case Description

In this section, the companies in the survey are described in more detail. Our classification of the companies and the presented characteristics are based on data collected during the interviews.

The four companies come from different industries: Robotics (C1), Transportation (C2), Automation (C3) and Aerospace (C4), and are subject to different certification standards. They employ different business models, some are market driven whereas others work in tight collaboration with governments and large organizations. See Table 4 for a summary of the characteristics of the four participating companies. In total, five interviews were held, one at each of the companies C1, C3 and C4, and two at C2.

C1 belongs to a division of a global company which has long experience developing industrial robotics and recent experience with service robotics. Applicable standards for industrial robots are ISO10218. One interview (I1) was held at this company with a system architect. Software is used extensively by the company for non safety-critical functionality, although not for safety-critical functions. The development can be seen as system integration of software and hardware components, either in-house or acquired from third-party sources. The company does not provide end user certificates for their products.

C2 belongs to a national branch of a global company. They develop train-signaling systems for several countries. Applicable and mandatory standards for railway signaling systems in the European Union is EN50128. At this company two interviews were held, (I2) and (I3). One with a safety manager and one with a software architect. The company has several years of

experience in developing safety-critical software, and they recently changed development process from a version diversity based process to a single version unit test based process. The company has customers in several countries all over the world, as a result their products must be able to comply with different legislation as well as specific customer needs.

C3 belongs to another division of the same corporation as C1 but develops automation solutions. The company is subject to the ISO61508 standard. We conducted one interview (I4) at this company with a safety manager who was responsible for contact with certification authorities and quality assurance of the development process. They have a strong focus on software, both non-critical and safety-critical. Their customers are end-users, which integrate the company's products into larger systems, e.g. processing plants.

C4 belongs to a multinational company in the defense & aerospace industry. The company base their development process around the RTCA/DO-178B standard. One interview (I5) was held at this company, with a safety manager responsible for the software development process. The company has a long history of developing avionics, i.e. aircraft software, and works in close collaboration with the national authorities responsible for certifying aircraft systems. They consider themselves system integrators and deliver a product that is intended to be used directly by end-users. All customers are governments.

4 Results and Discussion

This section presents the results from our study, as well as our interpretations of the result. The four themes presented in this section were identified during the transcript analysis.

For each theme, we first present our conclusions, then our analysis and finally the supporting evidence for our claims. References to RQs are given in the analysis, while the direct responses to RQs are summarized in Section 5.

4.1 Human Factors – Skills, Experiences and Attitudes

This theme covers skills and knowledge among developers and managers, as well as their attitudes towards safety critical development, and how this impacts the development and certification process. From a flexibility point of view, human factors affect team composition and the performance of employees.

Development of safety-critical systems is not about writing code. It is about understanding the problem that should be solved by the system and be aware of the special nature of safety-critical systems, i.e. that they can cause harm to people. For instance, several interviewees mentioned that the quality of testing was depending on good knowledge about the system, its intended functionality and about safety-critical development in general. This was exemplified by I2 stating: *if you do not ask the right questions you will not get the right answers.* Having employees with the right knowledge profile was explicitly considered by

two of the interviewees as the main asset of their company. I5 emphasized, that there is a big knowledge difference between developers doing normal application code and those that make safety-critical code. I3 considered that developers should work across the system-software boundary to get a better understanding of the product domain and intended system functionality. Agile team practices help achieving these goals (RQ1).

It is essential to document the knowledge in a way that can be accessed by the ones using it. Especially in an organization that change process model or reassigns experienced personnel to other projects and products. For instance, I1 pointed out that undocumented knowledge might be a challenge when going from one process model to another. This was further supported by I3 who thought that getting information out about novel development concepts and how to apply them posed a challenge. I5 strongly emphasized that information should be documented and be freely accessible by all concerned parties. This is both an advantage and a challenge of agile practices, which focus on communication, but not on documentation (RQ1). One key challenge that was identified by several of the interviewees was to keep the organizations' safety awareness high and how to improve it, i.e. that the people in the company should always be aware of that the product is safety-critical and have the potential of causing harm, and base their actions on this awareness (RQ2). It was considered to be both difficult and time consuming. For instance, interviewee I1 said that it cannot be forced upon the organization; it has to grow with time.

From a managerial perspective, safety awareness, or lack of it, will greatly impact the development. For instance in one of the case companies, management pushed for shortcuts and reduced staffing of key personnel, which in the short term might have led to some reductions in cost but in the long run adversely affected the company's ability to certify larger changes in their products. The reverse case, when management is too aware, is perhaps not ideal either. Although, a low level of safety awareness might lead to costly decisions, a too high level might result in fear of change, as was implied by I1. Though, the latter case would probably be better from a safety perspective, however, less so from a flexibility point of view (RQ2).

Although not directly connected with flexibility, human factors play an important role in safety-critical development. To be successful, a flexible process should consider these factors and provide adequate support for knowledge sharing. As identified in [2], agile practices (XP), seems to support developers motivational needs (RQ1). Their motivational needs included, among others: knowledge sharing, support for the less experienced and knowledge acquisition.

4.2 Requirements and Verification

This theme is related with testing, requirements engineering and how the requirements are elicited during the development process.

Good requirements and a complete requirement specification is the key to safety critical software development, or any kind of development for that matter. Without the right requirements, on the right level, verification and validation

becomes hard to do in any meaningful way. A complete set of requirement is perhaps not practical or even possible which means that the gaps must be filled by other means. In this regard, testing plays an important role by finding implementation and design errors. To get good quality test cases the testers need to be knowledgeable about the system, the problem domain and the development process. This view is strongly supported by both I2 and I3. For instance, I3 described how requirements specifications on the functional level were used to develop unit test cases, and, as a consequence because the testers needed more information to adequately test the units, additional data (requirements) were attached to the function specifications. The result was that testing became more occupied with finding discrepancies between the code and the specification rather than finding actual errors in the source code. Agile practices, with their focus on working code over rigorous documentation may counteract this (RQ1).

From our study we must conclude that formal methods are not used widely by industry. Only in C2 were formal methods used and only in a specific case. I2 expressed a wish to increase the use of formal methods because they had problems finding certain kinds of errors related to logic and combinatorics. When speaking about the software architecture and system properties, I4 explained that they did not use formal methods although they followed modeling guidelines when building their architecture model.

Safety related requirements are described, in all four cases, to be elicited through an iterative process, starting with a new design or change request. Then some form of risk analysis technique is applied, e.g. FMEA in the C1 case and HAZOP in the C2 and C3 cases.

Interviewee I4 explains that they find requirements engineering challenging because they are required by their standard to finalize the design before starting to write code. One explanation for this could be that the same engineers do both design and coding, as explained by I4, which is in sharp contrast to agile practices (RQ1). No other interviewee reported on having this difficulty.

A closely related topic is requirements traceability. Traceability is mandated by the safety standards and must be maintained for safety-critical systems (RQ2). I4 described traceability as a time consuming and labor intensive task that would probably not have been performed if the standard did not require it. In contrast, I5 gave the impression that traceability was a well integrated activity, but also agreed that it was time consuming. In both cases tool support existed as well as a functional specification that traced product requirements through system functions down to components, but in the C3 case the specification was seen as making tracing harder as opposed to the view of it as a helpful tool in the C4 case. One explanation of the different views might be that the level of detail in the functional specification is too high in the C3 case.

From a flexibility point of view requirements engineering and verification & validation are essential. If these activities are not aligned with the process or the architecture, flexible development or composition of system variants might not be feasible. For instance, one limiting factor today, that was reported by I4, is that the cost, in terms of certification overhead, of changing a safety-critical

system, i.e. make a new release, is proportional to the number of releases rather than the implementation effort of the releases (RQ2). This means that there is an incentive to have few releases with many changes, which is in conflict with the idea behind agile processes (RQ1).

4.3 Agile Development

This theme covers the trend towards more iterative and agile development processes and standard related challenges.

Agile processes, at least agile inspired processes, are being introduced or have been used for some time in safety-critical development. Although, only the C4 case explicitly followed an agile process model both the C2 and C3 cases made use of iterative development processes that seemed to share some characteristics of an agile process (RQ1). Interviewee I2 described their process as working back and forth in 14 days test release cycles. This agile interpretation is further supported by interviewee I3, who had previous experiences of working with Scrum, at another company, and did not think that the development process of C2 was any different from previous experiences. The C4 case used Scrum in their project teams.

Some challenges were identified by the interviewees. For instance interviewee I4 stated that there is a conflict between their need to adopt a more agile process and their certifier's insistence on conformance with the standard, which is means-prescriptive and prescribes a waterfall-based process (RQ2). Another challenge that was identified by interviewee I5 is maintaining independence in the development teams, i.e. a person is not allowed to produce and review the same artifact. This was resolved by keeping track of who did what.

A common belief is that agile processes are in conflict with the requirements of safety standards [12]. Our conclusion is that this might be the case when it comes to means-prescriptive standards, e.g. C3, but not for objective-prescriptive standards, e.g. C4. In fact, interviewee I5 saw Scrum primarily as a project management tool for work planning and did not seem to think that it was, in any way, in conflict with RTCA/DO-178 (RQ2).

4.4 Variants and Components

The last theme we identified covers system variants and how reusable components are handled by the case companies.

Reusable software components have the potential of reducing development time and the cost of certification, at least in the presence of system variants. In both case companies that had variants, C2 and C4, efforts were made to isolated software functions into reusable components or frameworks (RQ3). In the two other cases, C1 and C3, variants did not exist and little or no effort was put into creating reusable components.

There are different approaches towards creating reusable software components. For instance, interviewee I2 described their layered software architecture

that has a generic bottom layer and more specific adaptation layers at the top (RQ3). Each layer can be considered as a reusable component that can be certified and reused in other situations, although it would still be necessary to certify the integrated system finally delivered to the customer. A change to a higher layer does not force a re-certification of a lower level, however, the reverse is not true (RQ2). Interviewee I5 explained another approach, where they considered a solution with a main system containing all functions for all variants. A variant would then be an instance of the main system with some functions deactivated. They also considered a solution where common functions were put into components that were declared to a suitable criticality level. Sufficient evidence could then be collected for each component and be used repeatedly whenever the component was integrated into a system variant.

Reusable components seem to be one of the keys to enable flexibility in safety-critical development (RQ3). At least interviewee I5 states this directly but it is evident that in the C2 case the fact that they can reuse their layers in different variants enables them to be very flexible when it comes to system composition and to meet the needs of their customers. It is also interesting to note that component reuse is encourage by, at least, the RTCA/DO-178B standard (RQ2).

5 Conclusions and Future Work

We launched a qualitative survey of companies developing safety-critical systems with software, in four different industry domains. Although governed by different standards, the characteristics are very similar across the domains. Based on interviews with five practitioners (safety managers, system and software architects) we conclude that human factors and quality of requirements are central for safety-critical systems, as they are for non-critical systems.

We conclude that issues related to *agile methods* (RQ1) include aspects, which are in line with safety goals, for example, their focus on communication, teamwork across boundaries, developer motivation and good code. Challenging issues of combining agility and safety include less focus on documentation, tight collaboration between development and test, in contrast to independent test teams, and many releases, which conflicts with certification procedures required for each release. Practice demonstrated the feasibility of combining agile and safety. One of the surveyed companies, C4, used agile processes, while C2 and C3 cases used iterative development processes.

Regarding system composition and *safety certification* (RQ2) we observe that safety awareness is a key human aspect. However, a challenge is that it may lead to fear of change, hindering flexibility. Traceability, as mandated by the safety standards, may support flexibility since it helps identifying dependencies to handle during evolution. The cost of maintaining traceability is high, as is the costs for safety certification, although both being a necessary condition for making safety-critical systems. The use of agile processes is possible to combine with safety standards, although some implicitly assume waterfall processes.

The role of the *software and system architectures* (RQ3) is primarily to harness reusable components, which is a key strategy to make safety-critical systems development and composition more efficient. A layered architecture may also help isolate changes, and thus the need for re-certification.

We conclude that agile development and flexibility may co-exist with safety-critical software development, although there are specific challenges to address by research and industry practice. Future work includes e.g. designing modeling concepts that may reduce the certification overhead for software changes, under kept standard compliance [10].

Acknowledgements. The work in this paper was funded by the Swedish Foundation for Strategic Research under a grant to Lund University for ENGROSS-ENabling GROwing Software Systems.

References

1. Baufreton, P., Blanquart, J.P., Boulanger, J.L., Delseny, H., Derrien, J.C., Gassino, J., Ladier, G., Ledinot, E., Leeman, M., Quéré, P., Ricque, B.: Multi-domain comparison of safety standards. In: Proceedings of the 5th International Conference on Embedded Real Time Software and Systems (ERTS2), Toulouse, France (2010)
2. Beecham, S., Sharp, H., Baddoo, N., Hall, T., Robinson, H.: Does the XP environment meet the motivational needs of the software developer? An empirical study. In: AGILE 2007. pp. 37–48. IEEE CS (2007)
3. Flink, A.: The survey handbook. SAGE Publications, 2nd edn. (2003)
4. Hawkins, R., Clegg, K., Alexander, R., Kelly, T.: Using a software safety argument pattern catalogue: Two case studies. In: Proc. SAFECOMP. pp. 185–98. Springer (2011)
5. Jansen, H.: The logic of qualitative survey research and its position in the field of social research methods. Forum Qualitative Sozialforschung / Forum: Qualitative Social Research 11(2) (2010)
6. Kornecki, A., Zalewski, J.: Software certification for safety-critical systems: A status report. In: International Multiconference on Computer Science and Information Technology. vol. 3, pp. 665–672. IEEE Computer Society (2008)
7. Land, R., Akerholm, M., Carlson, J.: Efficient software component reuse in safety-critical systems – an empirical study. In: Proc. SAFECOMP. LNCS, vol. 7612 LNCS, pp. 388–399. Springer (2012)
8. Leveson, N.: Engineering a safer world: systems thinking applied to safety. MIT Press, Cambridge, Mass. (2011)
9. McHugh, M., McCaffery, F., Casey, V.: Barriers to adopting agile practices when developing medical device software. In: 12th Int. Conf. on Software Process Impr. and Capability Determination. CCIS, vol. 290, pp. 141–147. Springer (2012)
10. Pederson Notander, J., Runeson, P., Höst, M.: A model-based framework for flexible safety-critical software development – a design study. In: Proceedings 28th Symposium On Applied Computing. Coimbra, Portugal (2013)
11. Runeson, P., Höst, M., Rainer, A., Regnell, B.: Case Study Research in Software Engineering – Guidelines and Examples. Wiley (2012)
12. Turk, D., France, R., Rumpe, B.: Assumptions underlying agile software development processes. Journal of Database Management 16(4), 62–87 (2005)