

INTEGRATION OF SOFTWARE SAFETY ASSURANCE PRINCIPLES WITH AN
AGILE DEVELOPMENT METHOD

Osama Doss

MSc by Research

University of York
Computer Science

May 2016

Abstract

Agile software development has had success in different domains. However there is one area where the implementation of agile methods still needs significant development – that is in the field of agile and safety-critical system development. In this field, software engineering processes need to be justified against the requirements of software safety assurance standards (such as ISO 26262 in the automotive domain). It is therefore important that agile development processes can be justified to levels of assurance equivalent to that provided by traditional development approaches. While there is existing literature concerning the integration of agile methods with specific safety-critical system development standards and agile methods, the question of how fundamental software safety assurance principles can be addressed within agile methods has received little attention. In this thesis we describe the results of practitioner surveys that highlight the primary concerns regarding the use of agile methods within safety-critical development. In the context of this survey, and of existing work on software safety assurance principles, we then present an initial proposal as to how assurance could be addressed with an existing agile development method – Scrum. This proposal was submitted to practitioners for initial feedback and evaluation. The results of this evaluation are also presented.

Table of Contents

List of Figures	5
List of Tables	6
Acknowledgements	7
Author's Declaration	8
Chapter 1	9
Introduction	9
1.1 Research Problem	9
1.2 Research Questions	10
1.3 Research methods	12
1.4 Thesis outline	13
1.5 Summary	13
Chapter 2	15
Background and Related work	15
2.1 Introduction	15
2.2 Definitions of Safety Engineering Concepts	15
2.2.1 Safety	15
2.2.2 Hazard	16
2.2.3 Failures, Errors and Faults	17
2.2.4 Safety-critical Systems	17
2.2.5 Safety Case	18
2.2.6 Safety Case Patterns	19
2.2.7 Safety Standards	20
2.3 System Safety	20
2.4 System safety process	21
2.5 Software hazard analysis techniques	22
2.6 Agile Background	23
2.6.1 Agile Software Development	23
2.6.2 Agile manifesto and values	24
2.6.3 Agile Principles	25
2.7 Agile Methods	26
2.7.1 Extreme Programming	27
2.7.2 Scrum	28
2.7.3 Scrum Roles	29
2.7.4 SafeScrum	31
2.8 Existing work on the use of Agile Methods for Safety-Critical Systems	32
2.8.1 Agile Processes and Compliance with Standards	38
2.8.2 Integrating agile practices with an assurance case	41
2.9 Summary	43
Chapter 3	44
Survey	44
3.1 Introduction	44

3.2	Importance of research	44
3.3	Survey Design	44
3.3.1	Survey findings	45
3.4	Results and Discussion	64
Chapter 4		66
Integrating Software Safety Assurance Principles with Scrum		66
4.1	Introduction	66
4.2	The 4+1 challenges and recommendation	66
4.2.1	Software Safety Assurance Principle 1:.....	67
4.2.2	Software Safety Assurance Principle 2:.....	69
4.2.3	Software Safety Assurance Principle 3:.....	70
4.2.4	Software Safety Assurance Principle 4:.....	72
4.2.5	Software Safety Assurance Principle 4+1:	74
4.3	Summary and further work	76
Chapter 5		78
Feedback and Evaluation on the 4+1 Scrum Integration		78
5.1	Introduction	78
5.2	Aims of semi-structured interview process	78
5.3	Research questions and their motivations:	79
5.4	Participants and Interviews	79
5.6	Interview Findings	80
	OVERVIEW OF STUDY FINDINGS (Participant 1).....	81
	OVERVIEW OF STUDY FINDINGS (Participant 2).....	83
	OVERVIEW OF STUDY FINDINGS (Participant 3).....	86
	OVERVIEW OF STUDY FINDINGS (Participant 4).....	90
	OVERVIEW OF STUDY FINDINGS (Participant 5).....	92
	OVERVIEW OF STUDY FINDINGS (Participant 6).....	95
5.7	Identification of emerging themes	97
5.7.1	4+1 principles within agile and mapping agile to standards.....	97
5.7.2	Agile and Documentation	100
5.7.3	Safety Backlog	100
5.7.4	Safety Team Member.....	101
5.7.5	Hybrid agile approach and relationship to safety.....	102
5.7.6	Sprint Duration for Safety.....	102
5.7.7	Queries and Recommendations.....	103
5.8	Summary and Issues Arising	104
Chapter 6		106
Conclusion and Future Work		106
6.1	Introduction	106
6.2	Initial perceptions	107
6.3	Initial Survey	108
6.4	The development of the initial proposal for the integration of the 4+1 principles 108	
6.5	Semi-structured interviews	109
6.6	Limitations	109
6.7	Future Work	110
Abbreviations		112
References		113

List of Figures

FIGURE 1 RESEARCH METHODS	12
FIGURE 2: IMPACTED POPULATIONS WITHIN SCOPE OF SAFETY ADOPTED[5].....	14
FIGURE 3: CORE SYSTEM SAFETY PROCESS ADOPTED FORM[4].....	19
FIGURE 4 AGILE MANIFESTO VALUES [40].....	21
FIGURE 5:DIFFERENCE BETWEEN AGILE METHODS AND TRADITIONAL METHODS [8].....	23
FIGURE 6: XP DEVELOPMENT PROCESS[64].....	24
FIGURE 7:SCRUM SKELETON ADOPTED FROM [14].....	25
FIGURE 8: SCRUM DEVELOPMENT PROCESS[73].....	26
FIGURE 9: SAFESCRUM MODEL [45].....	28
FIGURE 10: SPRINT IMPLEMENTATION FLOW [56].....	35
FIGURE 11: R. SCRUM REGULATED IMPLEMENTATION AT QUMAS [50].....	36
FIGURE 12: THE AGILE SOFTWARE DEVELOPMENT PROCESS [58].....	37

List of Tables

TABLE 1: AGILE AND SAFETY CRITICAL SYSTEMS SOFTWARE [53].....33

Acknowledgements

There are so many people to thank for helping me during my study and for making my short stay in York a lot easier than I thought it was going to be. Writing this thesis has had a big impact on me. I would like to reflect on the people who have supported and helped me so much throughout this period.

First, I would like to thank my supervisor Professor Tim Kelly for his guidance, encouragement and patience. Thank you so much for teaching me new skills, to look at research and my work in different ways and for opening my mind. Your support was essential to my success here.

I would like to thank Dr Katrina Attwood for her support and patience, and Drs Richard Hawkins, Xiaocheng Ge and Thomas Richardson for support, friendship and conviviality.

Finally, I would like to thank my Lord for his mercy and justice. And also my parents, wife and daughters for their love and support.

Author's Declaration

'I declare that this thesis is a presentation of original work and I am the sole author. This work has not previously been presented for an award at this, or any other, University. All sources are acknowledged as References'.

Some of the material presented within this thesis has previously been published in the following papers:

- Osama Doss, Tim Kelly "Assurance Case Integration with An Agile Development Method" In: XP 2015, LNBIP 212, pp. 347–349, 2015.
 - This material has formed the basis of chapter 1
- O. Doss and T. P. Kelly. 2016. Challenges and Opportunities in Agile Development in Safety Critical Systems: A Survey. SIGSOFT Software Engineering. Notes 41, 2 (May 2016), 30-31.
 - This material has formed the basis of chapter 3
- Osama Doss, Tim Kelly "The 4+1 Principles of Software Safety Assurance and Their Implications for Scrum" (Eds.): XP 2016, LNBIP 251, pp. 1–5, 2016.
 - This material has formed the basis of chapter 5
- Osama Doss and Tim Kelly. 2016. Addressing the 4+1 Software Safety Assurance Principles within Scrum. In Proceedings of the Scientific Workshop Proceedings of XP2016 (XP '16 Workshops). ACM, New York, NY, USA, Article, 17, 5 pages. DOI: 10.1145/2962695.2962712
 - This material has formed the basis of chapter 4

Chapter 1

Introduction

Agile methods are known for being fast, efficient and adaptive, as well as for fostering discipline and good practices in engineers. It is claimed that the use of agile methods can support both quality and team productivity [26]. Agile methodologies have grown in popularity in software development since the presentation of the Agile Manifesto in 2001[40]. They are intended to produce software of higher quality and lower cost [41], while satisfying both employers and stakeholders.

Safety-critical systems are those where the system can either cause harm to humans or the environment, or is responsible for preventing such harm. There are many such systems, for example, in the railway, medical devices and automotive domains. Most safety-critical systems must be certified by a regulatory agency (or at least independently assessed) to ensure that they are safe for deployment, and that appropriate development and verification practices have been applied. It is therefore important that adherence to the objectives of the relevant standards can be demonstrated [26].

Evidence and experience concerning the integration of Agile in the field of safety-critical software development is limited. However, there are some published case studies and research on successes or failures/ problems in that field, (e.g. [42,43,44]). Since the development of safety-related software is generally governed by standards, we need to investigate whether it is possible to use agile methods that are flexible with respect to planning, documentation and specification while still being acceptable by standards [45]. In particular, we need to consider how a structured argument providing assurance of the safety of the system can be incorporated with a typical agile development method.

1.1 Research Problem

Whilst the use of agile methods is seen by some as attractive – for example, there is evidence of increasing use in safety-critical domain (e.g. in the railway domain for Train control systems, automatic control, Doppler radar, Axle counters) - there are still many safety experts who express concerns. For example, Redmill [46] raises concerns about

whether Agile incremental development would be a "good thing" in the safety-critical system domain, and states that evolutionary delivery would not be. He raises the question of how the importance of safety features could be distinguished if we cannot envisage the working system early in the analysis process? Hazard

identification and analysis cannot be carried out on a system in the absence of a design. As is clear from the literature reviewed in Chapter Two below, the question of how to integrate Agile methods and Safety Assurance is not new (see related work section). But there is one particular area of practice that remains neglected in the existing work – namely the integration of safety (assurance) case development with an agile approach.

A safety case is the argument and evidence that establishes the acceptable safety of safety-critical system [29]. It is normally prepared (by the developer) and assessed (by an independent assessor or regulator) as part of safety critical systems development. Safety cases are an increasingly widespread approach to the management of assurance [47]. Structured argumentation approaches (such as the use of the Goal Structuring Notation – GSN - [48]) have become popular as a means of explicitly representing the arguments (and links to evidence) contained within a safety case. In this thesis, we are concerned with the research problem of how assurance case development (including the incremental development of structured arguments) can best be integrated with a typical agile development method.

1.2 Research Questions

We propose the following two research questions (RQ) concerning the relationship between Agile Methods and Safety-Critical systems:

RQ1 What are the current concerns and opportunities voiced by safety-critical systems professionals regarding the use of agile development methods for safety-critical systems development?

RQ2 What changes are necessary to the Scrum Process in order to address the 4+1 Software Safety Assurance Principles?

As will be outlined in Chapter 2, there has been significant development in recent years towards integrated methods for agile and system safety engineering. One notable example is the Scrum methodology, which incorporates processes and concerns from the IEC Functional Safety Standard 61508. However, what is missing in this work is any mention or treatment of the safety case. It is our concern to examine how this can be integrated into a certification process. It is infeasible for us to establish an entirely new integrated

approach within the timescales of the Masters program. Therefore we intend to base our approach on extending the existing work of Scrum to address safety case development.

Our research will focus on investigating best practice evolution of GSN arguments as a means for safety case development as part of a Scrum process. We expect to develop guidance that will

support the development of a goal structure as an integral part of this process. Incremental, or phased, safety case development is already recognized as a useful activity within safety engineering. It can be advantageous to release the safety case in incremental stages throughout the project to gain early acceptance of the project safety approach [49]. In addition, past work has suggested that safety case argumentation notations (such as GSN) can help with providing a lightweight mechanism for safety case evolution. By addressing the two research questions defined above, we aim to examine how at least one aspect of software safety assurance and agile development methodology can be usefully aligned. The following hypothesis is therefore proposed:

It is possible and useful to successfully integrate software safety assurance case development within an existing agile development method (Scrum), in a way that can help address existing concerns.

- "Useful" - In this context this means that it means that it helps support accepted principles of safety assurance
- "Successfully" – Assurance is provided whilst still enabling the process to be agile
- "Existing concerns" - there are a number of concerns that people have expressed with using agile methods in safety-critical development (e.g. absence of necessary documentation). These are discussed further in Chapter 3.

Past work [48] has suggested that safety case argumentation notations (such as GSN) can help provide a lightweight mechanism for safety case evolution. Answering the two research questions defined above, we aim to examine how at least one aspect of software safety assurance and agile development methodology can be aligned within a GSN framework informed by the 4+1 principles. The principles are outlined in Chapter 4 below. We contend that this approach will address various existing concerns relating to the integration of safety and agile

Useful needs to be that it helps both in the attainment of agility (i.e. we can build better software more efficiently) and the achievement and assurance of safety (i.e. we can better address safety concerns as and when they arise). As well the 1-to-1 semi-structured is to investigate the success of the proposed framework for safety case development within Scrum

1.3 Research methods

In order to answer the research questions and substantiate the proposed hypothesis, a defined research method is necessary. Beyond the obligatory literature review, the research methods being used in this study are survey, peer review and interviews:

- We will conduct a survey to investigate the practical problems which practitioners perceive and experience in integrating the two disciplines. The answers collected from this survey will help to establish a framework to promote an iterative approach of building and evaluating artifacts, in the context of a safety case, as a part of an agile development methodology (Scrum).
- 1-to-1 semi-structured interviews will also be used with some of the respondents from our initial survey, the purpose of this interview study is to investigate the success of the proposed framework for safety case development within Scrum.

As described above, we will investigate existing concerns from safety critical systems engineering professionals by eliciting their opinions through a survey. The feasibility, and practicality of the proposed integration of safety case development with Scrum will be judged through peer review using structured interviews.

We have chosen these methods firstly because the lack of information from *practitioners* concerning the integration of Agile methodologies into safety-critical development. Secondly, whilst it is desirable to seek case study experience and evaluation of the proposed approach, it has not been possible to conduct a case study within the duration of a one year MSc by Research programme. Given the timescales of the work, we assert that having practitioners' views from the real world, scalable problems is perhaps more significant for problems of this kind than small-scale case study examples.

The results from the research methods, survey and semi-structured interview helped us provide a clear direction in terms of the importance of incremental hazard analysis, safety requirements development, and assurance case development.

We will gain from these methods offered one of the most feasible methods for gaining insight as to whether the proposed approach would be successful (i.e. support agility, not compromise safety) given that it was not feasible to run a trial software development during the timescales of the masters research programme, to evaluate how safety activities are currently being proposed within the Scrum method, and to help define a process model for how requirements

development, hazard analysis and assurance case development can be performed as incremental activities.

The figure below indicates the steps of our research methodology, and indicates what needs to be done in future work in this area.

1.4 Thesis outline

This thesis is organized as follows:

Chapter 1: This chapter introduces our work, research problem, questions, and methods.

Chapter 2: Introduces the literature relating to safety critical systems engineering and agile methods when dealing with software development. It aims to give an overview of the various terms, safety standards, Agile methodologies, including their values and principles. The chapter provides an overview of the existing literature that specifically addresses the use of agile methods in the safety critical domain.

Chapter 3: In this chapter we describe the results of our survey designed to elicit the opinions of safety practitioners as to the challenges and opportunities posed by the application of agile development methods in the field of safety critical systems development.

Chapter 4: In this chapter, we present an initial proposal as to how Scrum can be modified to address the 4+1 principles of safety assurance.

Chapter 5: In this chapter we describe the results of the interviews conducted to identify and address challenges associated with the integration of Agile methodologies into safety-critical systems development.

Chapter 6: concludes the thesis with a summary of the results, limitations and further work.

1.5 Summary

Ultimately, this research will develop and evaluate a process model of an adapted version of Scrum that clearly integrates the activities of software safety requirements evolution, software hazard analysis and software safety (assurance) case development. To support the assurance case development aspect of this process, the results from the survey and semi-structured interview have provided a clear direction in terms of the importance of incremental hazard analysis, safety requirements development, and assurance case development (i.e. they indicate clearly that these activities must be performed within an incremental, rather than simply being up-front or end-of-development activities).

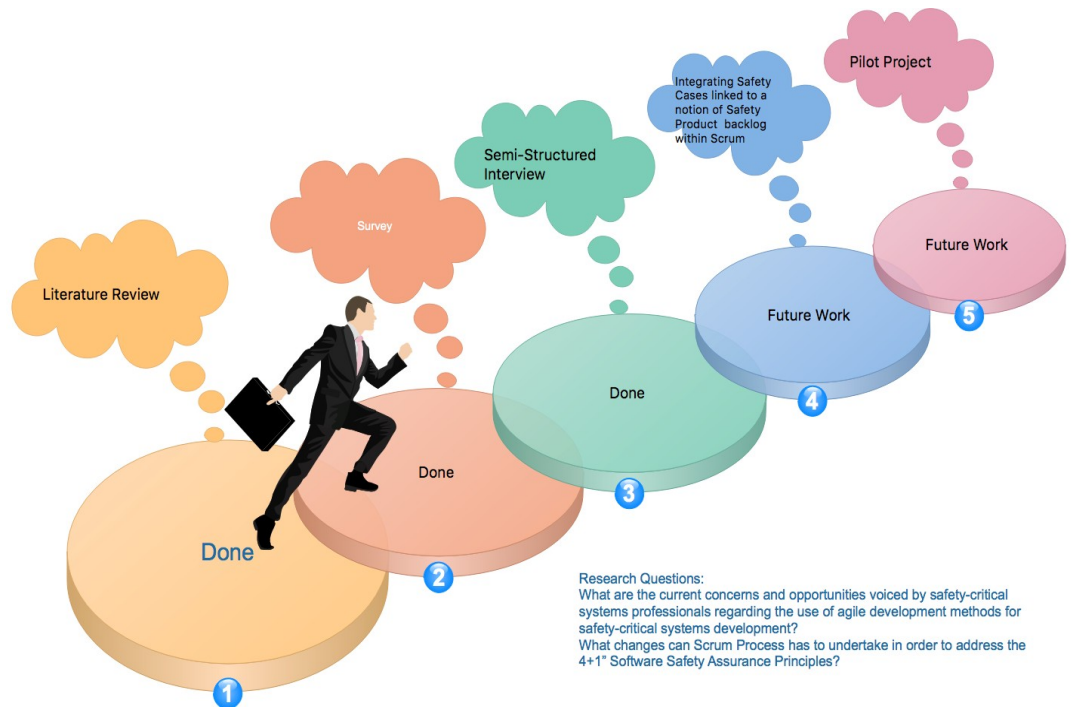


Figure 1 Research Methods

Chapter 2

Background and Related work

2.1 Introduction

This chapter introduces important concepts pertinent to the thesis. These concepts are important for understanding the thesis challenges and recommendations. Firstly, we define basic concepts in safety engineering and discuss standard processes and techniques from that field. We then address essential issues from agile methods, introducing the agile manifesto and core values and principles. These are then illustrated through three examples of agile methodologies: Scrum, Extreme Programming (XP) and Scrum. Finally, we provide a summary of previous work which identifies challenges and proposes methods for integrating agile methods within safety critical systems engineering.

2.2 Definitions of Safety Engineering Concepts

2.2.1 Safety

Safety has been defined in different ways in the literature and various domain-specific standards. Here, we illustrate some of the scientific terms about safety:

Safety is a state in which someone or something is secure from the possibility of death, injury, or loss [1]. Safety as defined in MIL- STD-882D [79], Standard Practice for System Safety, is “freedom from those conditions that can cause death, injury, occupational illness, damage to or loss of equipment or property, or damage to the environment”. In addition safety is described as freedom from accidents or losses, however, it should be pointed out that no system can be absolutely safe, the aim in design a system is to be as much as adequately safe. Figure 2 illustrates the scope to which the concept of safety can apply, in terms of potentially impacted people, equipment and the environment.

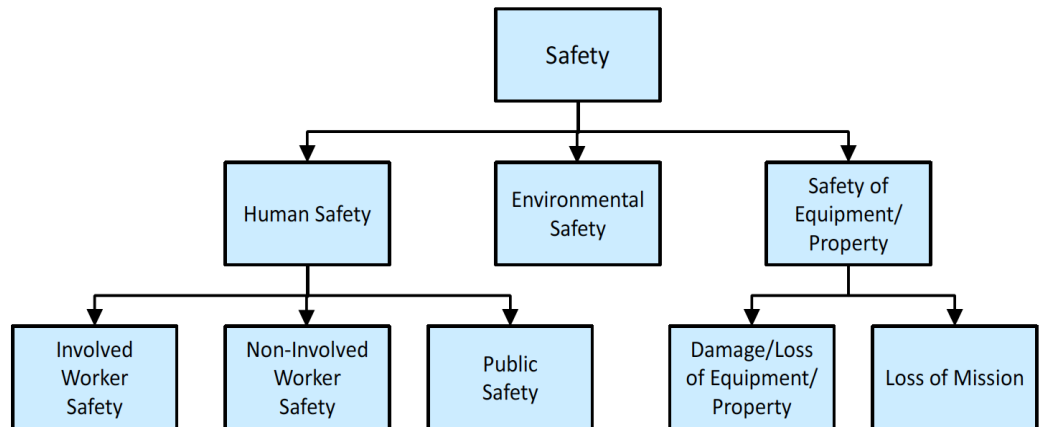


Figure 2: Impacted populations within scope of safety [5]

The concept of safety revolves around the related concepts of hazard, risk and mishap, which are closely entwined together [1]. These concepts are explored below. For the system to be made safe, the potential for mishaps must be cut down or eliminated. Any failure in any part of the system - whether mechanical, electronic, software or human - has the potential to cause injury or death and ultimately to lead to significant litigation [2]

2.2.2 Hazard

NPR 8715.3C defines a hazard as “a state or a set of conditions, internal or external to a system, that has the potential to cause harm” [5]. Hazards are caused by failures (discussed in Section 2.2.3 below). Examination of the effects of hazards effects is a way to determine system risks.

In [3] Leveson indicates that recent technical advances in various fields such as science and industry have created new hazards. For example, developments that have occurred in the food industry, such as the increasing use of food additives, adversely affect public health. Similarly, large numbers of people may be harmed by unknown side effects of pharmaceutical products. Arguably, existing safety engineering strategies have limited impact on many of these hazards [3].

Hazard analysis identifies and lists potential hazards in the system design including subsystems, components, and interfaces for the intended use of the system in its intended operational context [62,63,7] and considers their effects.

2.2.3 Failures, Errors and Faults

A failure is defined by Leveson as “the non-performance or inability of the system or component to perform its intended function for a specified time under specified environmental conditions” [7].

There are many reasons why a system might fail. Often, a failure is the result of an inherent weakness of the design or implementation. Failures also arise in the transition between a correct and an incorrect service.

An error: is that part of the system state that may cause a subsequent failure [16].

A fault: is the adjudged or hypothesized cause of an error [16].

The above definition of fault and error indicates that faults are preconditions for errors: a failure happens when an error reaches the interface of a service and changes the service. In other words, an error may happen when a fault is active. Also, failure can be identified as a hazard. Finally, failures, faults, and errors are sometimes collectively referred to as defects [19].

2.2.4 Safety-critical Systems

The literature contains many definitions for the term ‘Safety-Critical Systems’. Essentially, they all share the intuitive notion that the system is concerned with the consequences of failure: if the failure could lead to danger or loss, then the system is system-critical [6]. The term ‘safety-critical’ therefore refers to systems that either can cause harm or are responsible for preventing harm [26]. In general, ‘harm’ refers to loss life, physical injury or danger, economic loss or environmental loss.

John C. Knight [6] classifies Safety-critical systems into traditional systems and non-traditional systems. Traditional systems are active in the areas that have long been considered as safety-critical systems e.g. nuclear power, medical care, airspace, and weapons. In these areas, any failure can lead to death or disaster, loss of equipment and so on. However, there are some new types of system that have the potential for very high consequences of failure, for example transportation control systems. Knight considers these systems as non-traditional safety-critical systems.

2.2.5 Safety Case

Safety cases are produced to provide argument and evidence to substantiate a claim that a system is acceptably safe to function.

The purpose of a safety case can be defined as follows:

“The best safety case should contain a clear, exhaustive and defensible argument that a system is acceptably safe in the designated frame.” U.K. Ministry of Defence Ship Safety Management System Handbook JSP430 [77]

“A safety case is a comprehensive and structured set of safety documentation which is aimed to ensure that the safety of a specific vessel or equipment can be demonstrated by reference to:

- Safety arrangements and organisation
- Safety analyses
- Compliance with standards and best practice
- Acceptance tests
- Audits
- Inspections
- Feedback
- Provision made for safe use including emergency arrangements[29]”

As we can see from the above definitions, the U.K. Ministry of Defence focuses on some important elements. Furthermore, as we shall see later in this chapter, some of these elements are also the main goals for agile methods.

The concept of the safety case has been introduced across many industries for example the

field of defence, aerospace and railway. So, in order to meet safety standards, it is necessary to provide a structured argument followed by evidence to support your claim, thus, argument and evidence are crucial elements of the safety case that must go hand-in-hand [29]. Documenting safety arguments is considered beneficial in improving safety assessment and safety case maintenance [31].

One approach to representing safety arguments is the Goal Structuring Notation (GSN). The purpose of a goal structure is “to show how goals are broken down into sub-goals, and eventually supported by evidence (solutions) whilst making clear the strategies adopted, the rationale for the approach (assumptions, justifications) and the context in which goals are stated.” [66]

2.2.6 Safety Case Patterns

Generic safety case patterns have been identified from the study of existing safety cases and safety standards, and from discussion with safety case practitioners. There are several published pattern catalogues, in which all of the patterns presented have been subjected to (at-least) peer review [48]. Software safety argument patterns present best practice in structuring and presenting software safety arguments.

Hawkins [33] demonstrates that software safety argument patterns may be combined in order to construct a software safety argument for the system under consideration. The following argument patterns are currently described in Hawkins’s Pattern Catalogue:

1. High-level software safety argument pattern – This pattern provides the high-level structure for a generic software safety argument. The pattern can be used to create the high level structure of a software safety argument either as a stand-alone argument or as part of a system safety argument.
2. Software contribution safety argument pattern - This pattern provides the generic structure for an argument that the contributions made by software to system hazards are acceptably managed. This pattern is based upon a generic ‘tiered’ development model in order to make it generally applicable to a broad range of development processes.
3. Derived Software Safety Requirements identification pattern - This pattern provides the generic structure for an argument that derived software safety requirements (DSSRs) are adequately captured at all levels of software development.
4. Hazardous contribution software safety argument pattern – This pattern provides the generic structure for an argument that the identified DSSRs at each level of software safety

development adequately address all identified potential hazardous failures.

5. Strategy justification software safety argument pattern - This pattern provides the generic structure for an argument that the strategy, which is adopted in a software safety argument, is acceptable given the confidence that is required to be achieved in the relevant claim.

Later on we will explain more on how to integrate these patterns in the research.

2.2.7 Safety Standards

Nowadays “safety” is an obsession, which is healthy. The formal interpretation of “safety” for systems within safety-critical domains is the role of standards. There are different standards that address safety-critical systems and regulations; each one of these standards has a different philosophy. The core concepts of each standard depend on arbitrary variances in criteria and assessment that come from the broad collection of stakeholders [35]. For system developers, this results in the challenge of having dozens of criteria to fulfill and various assessments to provide. In addition, some of these standards provide generic approaches, while some are designed for specific domains or industries. Hatcliff [35] identified that there are many applicable safety assurance standards such as IEC 61508 [38], ISO 26262 [39], EN 50128 [36], DO-178C [65], IEEE 7-4.3.2 [37] and IEC 62304[32].

McDermid [34] shows that safety standards do not normally determine the software development process; however they may recommend processes and practices to be used in order to achieve particular safety levels. If the standards do not explicitly define the software process, we are surely permitted to ask whether we could use agile methods as long as the safety goals can be achieved.

2.3 System Safety

System theory started from the 1930s and 1940s. Initially these theories were developed in response to the classical analysis techniques, to help cope with the increasingly complex systems starting to be built at that time [3]. The aim of the system safety is the reduction of losses to lives, systems and the environment. Therefore the main objective is to avoid hazards that can lead to loss or damage to the environment and to ensure the discovery of hazards to the fullest extent possible.

The US Air Force System Safety Handbook [78] provides the following definition of ‘system safety’: “The application of engineering and management principles, criteria, and techniques to optimize all aspects of safety within the constraints of operational effectiveness, time, and cost throughout all phases of the system lifecycle.”

System safety is defined in MIL-STD-882D [79] as follows: “The application of engineering and management principles, criteria, and techniques to achieve acceptable mishap risk, within the constraints of operational effectiveness and suitability, time, and cost, throughout all phases of the system life cycle” [4]. NASA general safety program requirements [80] define system safety as the “application of engineering and management principles, criteria, and techniques to optimize safety... within the constraints of operational effectiveness, time, and cost throughout all phases of the system life cycle [5].

System safety is thus a kind of process aim to manage the system, personnel, environment, and health accident risks that happen through the design and development of a safety-critical system. The goal of a system safety program is to reduce or eliminate hazards through design, engineering and management. That can be through the application of engineering, and management principles. Moreover, techniques are employed to improve safety within the constraints of operational effectiveness, time, and cost throughout all phases of the system life cycle. That leads us to the question: can the principles and processes defined for agile methods contribute in the field of “system safety” for safety-critical systems?

2.4 System safety process

MIL-STD-882D illustrates the essential aspects of the system safety process in eight principal steps, as presented in Figure 3:

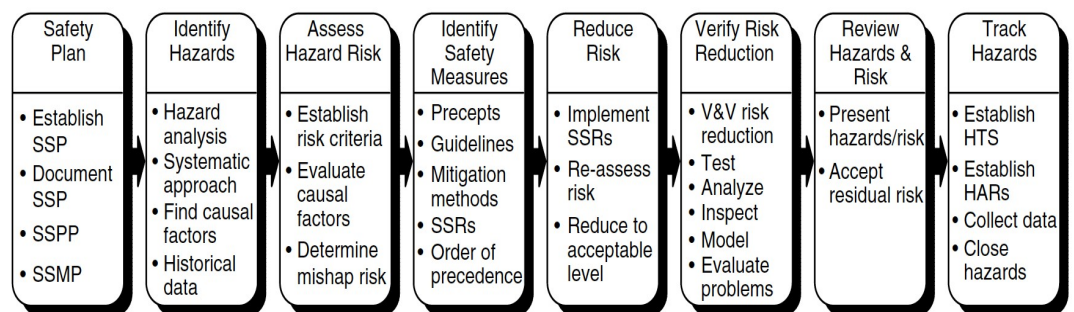


Figure 3: Core system safety process [4]

The system safety process is initially documented in the safety program plan, and is then carried through as all safety tasks are completed, including hazard analysis and reports. System safety is based on a lifecycle approach according on the idea that safety measures to manage risks and mishaps must be started as soon as possible in the life of the system.

Safety processes run parallel to and complement system development processes [28]. Safety process are structured around the notion of the hazard and its management: any changes with it can lead to danger or damage. For example, a System (airframe) - travelling down runway at circa 170 knots, without braking.

2.5 Software hazard analysis techniques

Regarding to (NIST 1993 National Institute of Standards and Technology)[81,67] Software hazard analysis “ eliminates or controls software hazards and hazards related to interfaces between the software and the system (including hardware and human components). It includes analyzing the requirements, design, code, user interfaces and changes.”

The IEEE Standard for Software Safety 1228-1994 [82] describes the relationship between the system safety analysis and the software safety analysis. If we are going to talk about software hazard analysis, we need to understand the role of the software on the performance of the system safety functions, as well the performance of the system on the monitoring and controlling functions and the impact of the software throughout the system. Therefore, the Software hazard analysis should be carried out within the context of the overall design of the system, for both attributes and assigned tasks of the software that contribute to the system's ability to perform its assigned tasks [67].

It is also important to note that Software Hazard Analysis is part of Software Design, the main goals of the hazard analysis program are to understand and correct deficit, as well to provide secure information. J. Dennis Lawrence [67] introduced four types of actions that may be appropriate, depending on the circumstances:

1. The system design may be changed to eliminate identified hazards which are affected by software or are not adequately handled by software, or to reduce the hazards to acceptable levels, or to adjust the system architecture so that identified hazards are compensated by defense-in-depth.
2. The software design may be changed to eliminate identified hazards, or reduce them to acceptable levels.

3. The quality of the software may be improved sufficiently to reduce the probability of a hazard to an acceptable level.
4. The application system may be rejected if it is considered too hazardous.

However, early hazard identification is the best approach. Thus it might be argued that one benefit of using agile methods is that they should be helpful in encouraging early and iterative evaluation that could reveal hazards and enable responsive introduction of risk reduction measures.

2.6 Agile Background

In this section, we introduce basic concepts from Agile Methods. This foundation is essential for comprehending the motivation of the thesis. Firstly, we introduce the Agile Manifesto and associated principles and values for agile methods. Then we describe three examples of Agile process methodologies: Scrum, Extreme Programming and SafeScrum.

2.6.1 Agile Software Development

Agile software development is an attempt to prioritise tasks in software development and to acknowledge that the user requirements change. Agile Methods can thus be defined as methods that try to focus on the primary goal of software development, i.e., the creation of working (defect-free) software [8]. The Agile philosophy also implies being both effective and sufficient for the current situation. Agile development is a broad term that combines to characterise a certain group of development methods. The core of Agile development itself is not a definitive methodology: rather, ‘Agile development’ is umbrella term that accommodates and describes various similar methodologies, as well as core principles and values.

The main idea of Agile was to create something “untraditional” and more effective, to improve product quality, and to reduce costs and the time to market [8]. Agile methodologies have increasingly gained respect in the software engineering field since the presentation of the Agile Manifesto in 2001 [40]. Agile Methods have started to be widely used as a development approach for software [32]. The majority of companies today use either traditional or agile approaches, or, increasingly, a combination of traditional and agile approaches [9].

2.6.2 Agile manifesto and values

In February 2001[8], the Agile Software Development alliance, which comprised seventeen independent representatives and several others considered to be leaders in the software industry, came together to introduce the “Agile Manifesto” [40,10]. The Manifesto they produced contained four core values, supported by twelve principles. These values and principles led to the evolution of lightweight methodologies, which were essentially people-oriented rather than process-oriented [27]. Figure 4 presents the Agile Manifesto values, which are discussed in more detail below:

**We are uncovering better ways of developing software by doing it and helping others do it.
Through this work we have come to value:**

Individuals and interactions over processes and tools

Working software over comprehensive documentation

Customer collaboration over contract negotiation

Responding to change over following a plan

That is, while there is value in the items on the right, we value the items on the left more.

Figure 4 Agile Manifesto Values [40]

Individuals and Interactions over Processes and Tools

Individuals and interactions is one of the most important aspects for high-performing teams. This principle motivates software developers to consider the people who are involved in the development process as high priority. Note that this does not mean that processes are not important, but that the priority is for “People rather than processes”.

Working software over comprehensive documentation

One significant difference between Agile development and more traditional software development philosophies is the concern with working software. Agile software developments depend on the development artifact itself and create only those documents that are needed at every stage [8,10]. Progress is made by delivering small releases of software to the customer at set intervals.

Customer Collaboration Over Contract Negotiation

Having a larger degree of customer involvement is vital, essential and logical. Logically, the customers have more knowledge about the product more than anybody. Therefore, the strategy in Agile Methods is to achieve customer satisfaction by regular engagement of customers rather than by considering the narrow confines of contracts. Moreover, it is recommended for customers to be involved in close daily interaction with development teams. In principle, such engagement should allow customers and developers to identify risks in the early stages of software development, which is what we seek in this research.

Responding to change over following a plan

Customers cannot generally predict all of their requirements a priori. Therefore, a gradual process, by which the requirements are incrementally understood by the customer and are delivered to and shared with the team, should be established [8,11]. Thus, Agile software development allows for changes in the development product, which derive from an increasingly clear understanding of the software.

2.6.3 Agile Principles

As we mentioned earlier, The Agile Manifesto introduces 12 principles which underpin the Agile philosophy: these principles are intended as a statement of what it means to be Agile, and to help people to identify and understand better what agile software development is centered around. As can be seen from the list below, the highest priority in Agile Manifesto is to satisfy the customer through early and continuous delivery of valuable software:

1. Highest priority is to satisfy the customer.
2. Welcome change.
3. Deliver working software frequently.
4. Business people and developers must work together daily.
5. Build projects around motivated individuals.

6. Face-to-face communication is best.
7. Working software is the primary measure of progress.
8. Promote sustainable development.
9. Continuous attention to technical excellence and good design enhances agility.
10. Simplicity – the art of maximising the amount of work not done – is essential.
11. The best architectures, requirements and design emerge from self-organising teams.
12. Introspection – teams should regularly review itself and its processes to try and improve. [8.17].

2.7 Agile Methods

Agile methodologies share the common essential idea that primarily concerned with the achievement of a working solution, in the context of changing user/stakeholder requirements. Although traditional development methods are, of course, also concerned with the development of a working solution, there is a fundamental difference concerning their flexibility in the light of requirements change [8]. Figure 5 illustrates this difference:

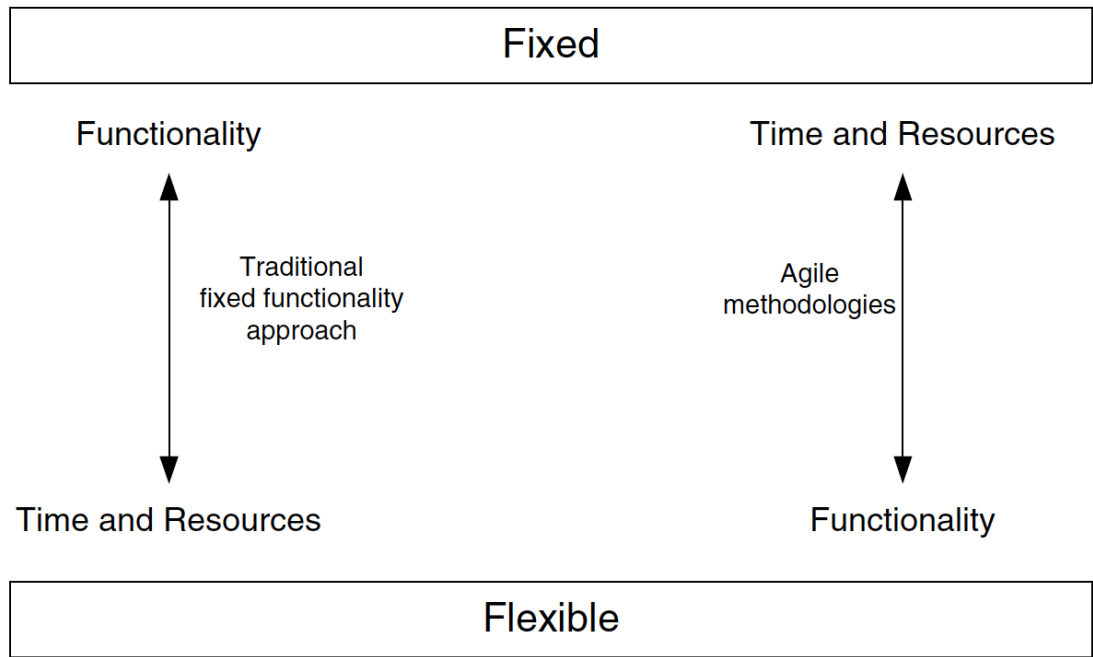


Figure 5: Difference Between Agile Methods And Traditional Methods [8]

There are a number of existing agile methods: for example, Extreme Programming (XP) [12], Scrum [14,30], Dynamic Systems Development Method (DSDM) [18], Adaptive Software Development (ASD) [23], Crystal [24] and Feature Driven Development (FDD) [25]. In this section, we consider three of these methodologies in detail, and discuss the potential for their integration in the development of safety-critical systems. The examples we consider are: Extreme Programming (XP), Scrum and SafeScrum

2.7.1 Extreme Programming

XP (Extreme Programming) is an incremental, lightweight methodology and discipline, that was conceived and developed to address the specific needs of software development [12]. It seeks to produce software which satisfies the customer through a set of working releases that focus on the timely delivery of specific software requirements. Further, XP defines basic principles to govern the development process: Communication, Simplicity, Feedback and Courage. XP has been one of the most widely adopted of the Agile approaches, and is widely seen as placing an emphasis on individuals and interactions over processes [13]. These four basic principles have led to the following key ideas presented within XP: (1) Code in pairs, (2) Stay in contact with the customer, (3) Create tests before coding then test heavily, (4) Short iterations, (5) Keep it simple, don't anticipate: code for current needs, and (6) Collective ownership [8]. These principles

have a positive impact on software development. For example, as discussed in the next chapter, we have found from responses to our survey that there is a perceived need for these practices in traditional safety processes.

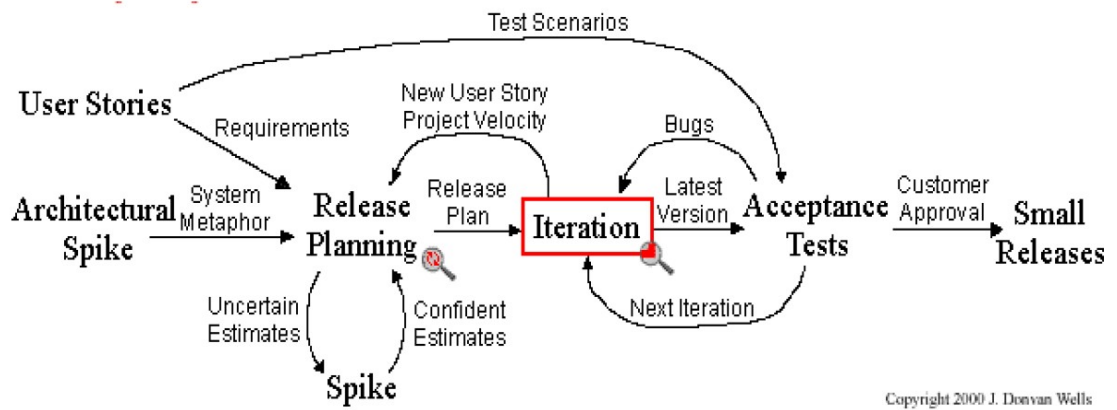


Figure 6: XP Development Process [64]

2.7.2 Scrum

Scrum is an iterative, incremental process, which can be captured as interlinked two circles, the lower circle and upper circle, as seen in Figure 7. The lower circle represents an iteration of daily development activities that occur in sequence. The output of each iteration is an increment of the software product. The upper circle represents the daily inspection process that occurs during each iteration [14].

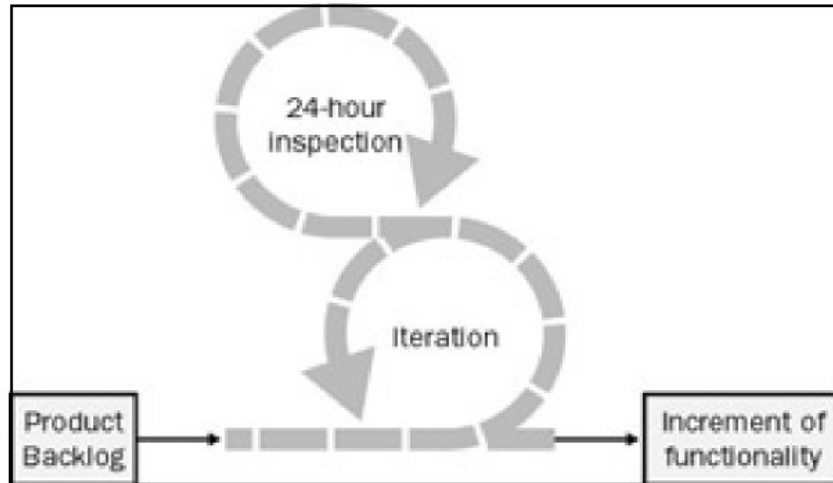


Figure 7: Scrum Skeleton [14]

Scrum is based on a small set of core values, principles, and practices [15]. In particular, the method defines three core team roles: the Product Owner, the Team, and the ScrumMaster [14].



Figure 8: Scrum Development Process [73]

2.7.3 Scrum Roles

Software development projects undertaken using the Scrum methodology are carried out by one or more scrum teams. Each Scrum team consists of three Scrum roles as follows: Product Owner, Scrum Master and Development Team. In this thesis, we have added one more element on top of the Scrum roles, i.e. Safety Member. This role is very important for projects which seek to use Scrum in the field of safety critical systems. We will discuss this addition in more detail later on.

The Product Owner

The Product Owner is responsible for ensuring that the most valuable functionality is produced inside the Product Backlog, and for turning the backlog items into developed features. Also, the Product Owner chooses to develop product functionality that solves critical business problems [14].

Scrum Master

The Scrum Master establishes what is necessary to help the team to be successful. In particular, the Scrum Master is responsible for teaching and championing Scrum and for helping the team to understand and embrace Scrum values and practices [15]. In addition, the Scrum Master acts like a lifeguard, protecting the team from outside interference and acting as a problem solver.

Development Team

The Scrum team is self-organized, which enables it to set up the best way for its members to achieve the team's goals, sprint goals, estimate effort and review the product backlog as determined by the Product Owner. The team typically consists of between five and nine people

[15]. Essentially, the team members must have good skills that match with the project goals and provide for the development of good quality software. The team members have collective responsibility for the success of each iteration and ultimately for the whole project [14].

Scrum Artifacts

Scrum Artifact is the collective term used for the resources which are used by the Scrum process: product backlog and sprint backlog.

Product Backlog

Product Backlog items initially describe features which are required to meet the Product Owner's vision [15]. The Product Owner is responsible for identifying, managing and

prioritizing the requirements and other contents that define the Product Backlog items, and for ensuring that the items are placed in the correct order. In addition, the Product Backlog is kept continuously variable, in order to meet the changing needs of the users/ customer.

Sprint Backlog

The Sprint Backlog describes the work as tasks: during a sprint, the Team selects tasks from the Product Backlog, potentially using it to develop product functionality for a given increment. Each sprint has a fixed start- and end- date/time. Each sprint should also deliver some identifiable, valuable, completed work to the customer or stakeholder – this is referred to as the ‘Sprint Release’.

2.7.4 SafeScrum

Stålhane et al. [45] propose “Safe Scrum”. This was 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 the safety standard IEC 61508. Stålhane illustrates two kinds of Product Backlog within the SafeScrum: the Functional Product Backlog and a Safety Product Backlog. These backlogs are linked to show dependencies between the functional requirements and the safety requirements. Whilst there are obvious potential opportunities to extend this work to include safety case development (e.g. linking the documentation and traceability of safety goals with an explicit safety argument development) there is no discussion of safety case development in this work.

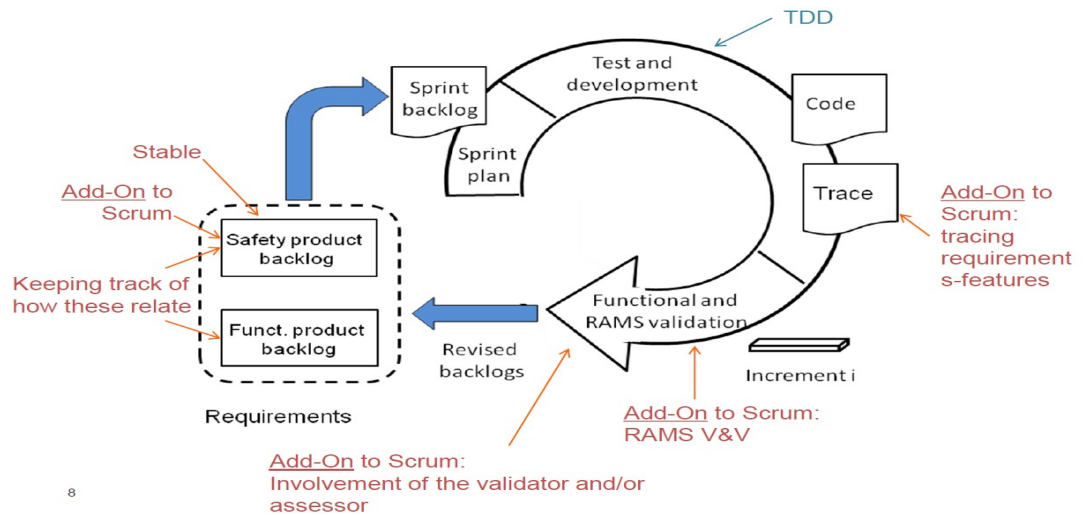


Figure 9: SafeScrum Model [45]

In addition, Safescrum has three main parts: The first part consists of the IEC 61508 steps needed for developing the environment description and the SSRS phases 1-4 (concept, overall scope definitions, hazard and risk analysis and overall safety requirements). The second part, which is the Scrum process and the last part, RAMS (Reliability, Availability, Maintainability, Safety) final validation in each iteration. SafeScrum focuses on compliance with IEC 61508. Perhaps. One possible criticism of this work is that, by focussing on one standard, it does not 'step back' and concentrate on the core intent of safety assurance and standards in general.

2.8 Existing work on the use of Agile Methods for Safety-Critical Systems

The purpose of this section is to give an overview of the existing literature that specifically addresses the use of agile methods in the safety critical domain. As well contains a selection of literature concerning the following:

1. **Safety Principles and Agile Principles, and their relationship**
2. **Integration of Assurance Processes and Agile Methodologies**
3. **Agile Processes and Compliance with Standards**
4. **Integrating Agile Practices with an Assurance Case.**

5. Safety Process and Agile

Ge, Paige, and McDermid [51] have published a paper on the iterative and agile development of safety critical software, in which they claim that there has hitherto been no tangible, documented deployment of agile methods on a real industrial safety-critical project. They mention a modular approach for building safety arguments incrementally using the Goal Structuring Notation (GSN)[48]. The process they propose “...precisely captures the notion of sufficient up-front design...” They also illustrate how to use safety patterns and introduce the notion of a modular safety argument to enable the iterative development of safety argument. However, their conclusion is that agile practices may not change the nature of the entire safety-critical development procedure model, but might improve the agility of the development. Furthermore, it is important to note that this paper is intended as a conceptual proposal, and is far from industrial practice.

Stephenson et al [52] propose an explicit representation of safety concerns in order to introduce Agile into the safety-critical development process: the Agile Health Model. Similar work has been done to introduce agility in security-critical systems, using an explicit representation of security concerns known as Iterative Security Architecture. They study the relationship between safety-critical and agile software development processes: this results in the Agile Health Model. Their main conclusion is that even though there are some problems with introducing agile methods in this area, there is nothing to prevent the use of agile methods in safety-critical development. They highlight that we need to add the certifying authority into the process. They also point to future work required to consolidate their findings.

Vuori [53] analysed the agile values and principles that are expressed in the Agile Manifesto, and considered what they would mean in the safety critical context and how they might be demonstrated. From the analysis, Vuori developed a generic, simple model to present the most prevalent agile features.

Agile value	Special meaning in a safety-critical development context	Practical principles to fulfil the meaning
[we value more]Individuals and interactions [than] Processes and tools.	Substance and understanding and sharing safety information is of the utmost importance.	Safety information should be discussed and not just be read in documents and analysis report
[we value more] Working software [than] Comprehensive documentation.	True safety is more important than fulfilling safety requirements (though the latter are mandatory)	We shall openly analyse safety and respond to real hazards first. Standards help in that, but we must not work only by standards. We need good systems, not systems that are documented as being good and safe.
[we value more] Customer collaboration [than] Contract negotiation.	While safety issues and features are important, they are always things that need collaboration so that we find practical, working solutions instead of non-robust ad-hoc solutions that cause more problems than they solve	Active collaboration with customers on safety-critical features.
[we value more] Responding to change [than] Following a plan.	When situation change, we need to assess the implication for safety immediately and not just blindly follow a project plan.	Every development must keep track of safety issues and respond to changes immediately
Principles	Special meaning in a safety-critical development context	Practical principles to fulfil the meaning
Our highest priority is to satisfy the customer	Early releases shall be safe and able to provide value	Safety of early releases needs to be validated;

through early and continuous delivery of valuable software.		hazard and analysis, safety assessment and testing needs to be an ongoing activity. This does not necessarily imply test automation, but an actively ongoing testing activity.
Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.	Safety requirement shall not hinder making sensible changes that provide value.	The product architecture needs to be flexible so as to encourage good changes that add value without compromising safety.
Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.	-----	Periodical releases shall not be compromised in their safety.
Business people and developers must work together daily throughout the project.	People who are responsible for safety or who are responsible for assuring it should work together with the development team.	Gradually, safety and reliability analysis tasks can be given to the development teams' tasks, yet independent analysis can be required to be carried out by an independent party.
Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.	People who participate in the development must be motivated by safety too and they need to be given resources and tools to use that motivation.	Participation in risk analysis is a very good way of sharing information to be documented properly.
The most efficient and effective method of	Safety information is shared in face-face meetings and not just	One metric of progress is how efficiently the process

<p>conveying information to and within a development team is face-to-face conversation.</p>	<p>assessment report. Safety issues need to be given place in meeting agendas.</p>	<p>can derive good, working software that is also safe to use and meets the safety requirements.</p> <p>Safety requirements need to be based on risk and safety assessment, not just standards.</p>
<p>Working software is the primary measure of progress.</p>	<p>True safety of the software is one measure of the progress; not just how designs and plans pass safety assessments.</p>	<p>Work on safety on safety features needs to be planned and resourced realistically, just like any other development activity</p>
<p>Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.</p>	<p>Working overtime exhausts developers and tester and makes them overlook essential factors. Tired people should not participate in safety-critical development any more than they should be exposed to risk in using machinery.</p> <p>Development of safe systems also benefits from experience, so it is not good if development if development change jobs due to exhaustion.</p>	<p>Work on solid safety architectures and elegant integration of safety features to the general architecture is essential.</p>

Continuous attention to technical excellence and good design enhances agility.	Safety, properly implemented, is technical excellence. Safety features need to be properly designed, not just add-ons. There must be absolutely no design flaws in safety systems.	Simple base architectures for safety features should be designed.
Simplicity--the art of maximizing the amount of work not done--is essential.	Simplicity and understandability of safety-critical features are essential qualities. Simple safety features are easy to adjust to the changing functionality.	
The best architectures, requirements, and designs emerge from self-organizing teams.	The team should have freedom for the design of safety features, but not safety requirements. Yet all decisions need to be on solid analysis and proven(or provable) techniques. The person who is appointed to be responsible for safety issues has responsible for safety issues has the final vote on all decisions, whether she/he is part of a team or not.	Present the safety requirements to the team clearly, let them understand what they mean and what their implication are and let the team do the designing as it best fits the whole system. Yet the results need to be validated in a sufficiently independent way
At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behaviour accordingly.	How well development of safety features has succeeded needs to be assessed as part of the team's self-assessment.	Reaching of safety goals as part of lessons learned-agendas and such.

Table 1: Agile And Safety Critical Systems Software [53]

Paige et al [54] addressed the tension between the needs of safety critical software development, and the principles of agile processes. They analyzed agile processes and their applicability in the domain of safety critical software, and presented a number of concrete recommendations for adapting agile processes to the domain. Paige et al suggested that cross-referencing could be used to link safety experience stories to user scenarios to illustrate how certain failure conditions affect the correct execution of code, and illustrated the use of simulator “pipelined iterations” in order to perform acceptance test, reiterated for a small V model. The authors also observed that the safety case was generally developed as an external activity to software. In our approach, we implement the safety case incrementally inside Scrum..

2.8.1 Agile Processes and Compliance with Standards

Most early theories talked about how to integrate agile methods in to the regulator environment [50]. Jonsson [55] illustrated the analysis of agile practices in regulated environment EN 50128 standard by mapping between EN 50128 requirements and agile practices. This mapping showed that all agile practices can be said to support some objective of the standard. Jonsson claimed that agile practices have the potential to lead to more efficient development, by reducing the gaps between customers, developer and tester. This can be achieved by short iterations, frequent integration and regular tracking of progress. Jonsson identified the fundamental challenge for the integration of agile processes into safety-critical product development as the need to find some way to produce all of the required documentation. He also stressed the importance of working with the quality assurance department and the safety assessor to ensure that the new practices are documented thoroughly.

Marques et al [56] presents a model for airborne software, including CRD-RM (Certifiable, Agile, Reusable, and Disciplined Reference Model). This is an attempt to define a generic model that can be instantiated on each airborne software project, in order to increase efficiency without interfering with DO-178C compliance. The authors argue that some of the compliance objectives are achieved during each sprint. However, the reference model is specified only for the domain of airborne software certification.

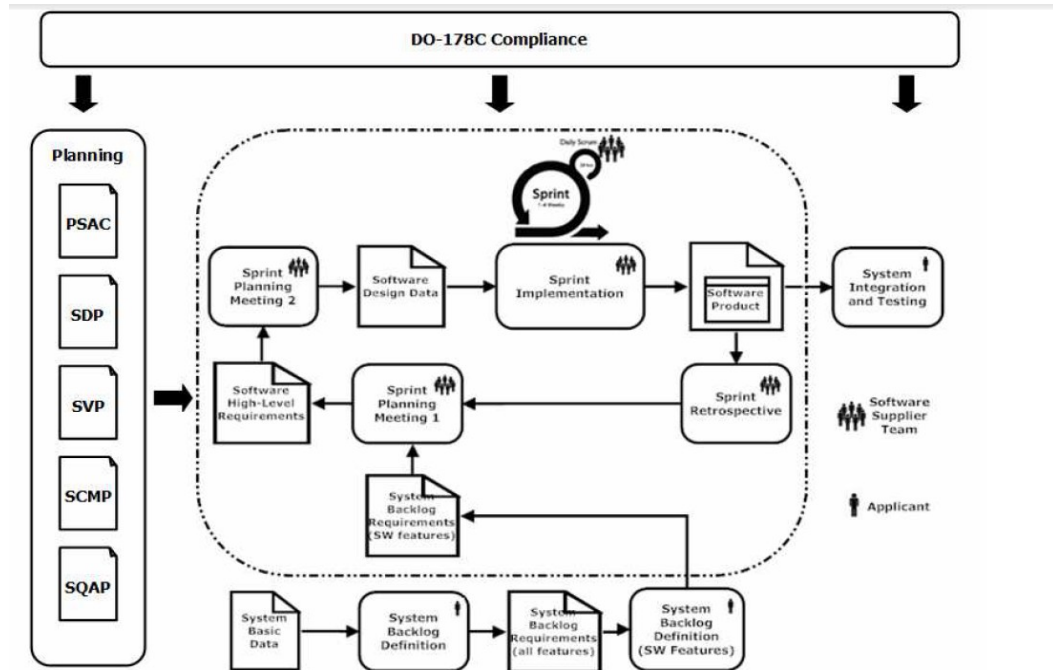


Figure 2. CARD-RM for Airborne Software

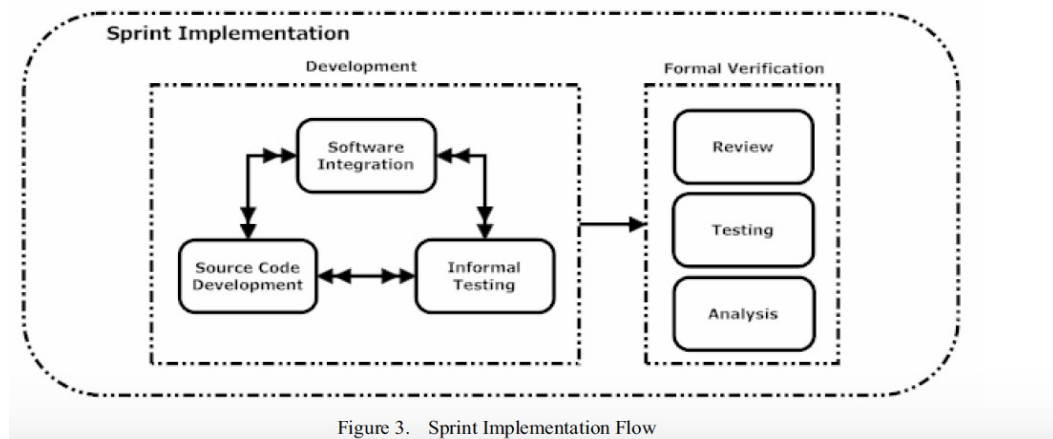


Figure 3. Sprint Implementation Flow

Figure 10: Sprint Implementation Flow [56]

In a similar vein, Fitzgerald et al [50] presented a case study to demonstrate that agile methods could be scaled to regulated environments. Their paper described a number of issues - quality assurance, safety and security, effectiveness, traceability, and verification and validation - and illustrated how they react with the model. The authors found that “living traceability” helps to establish compliance to standards and regulations, and concluded that agile processes can be augmented to work very well in regulated environments. The work also focused on how to implement an integrated model to achieve compatibility in terms of agility, safety/security, certification, and Quality Assurance. Fitzgerald et al made the important point that “the assumption of incompatibility between agile methods and regulated environments is more accidental than essential” [50].

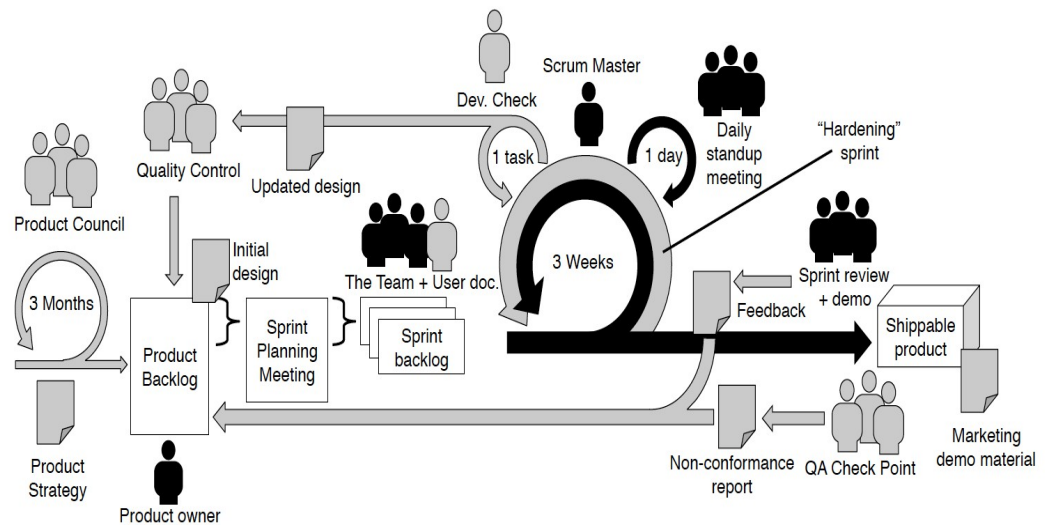


Fig. 4. R-Scrum: Regulated Scrum implementation at QUMAS.

Figure 11: R. Scrum Regulated Implementation At QUMAS (Management and Compliance Solutions) [50]

Coe and Kulick [57] examined potential sources of conflict between Agile principles and DO-178C certification requirements in the context of the inherent technical and procedural complexities in the aerospace domain. The Agile process emphasizes good communication and highlights the benefits of conveying information without rigorous process. However, DO-178C requires documentation to reflect organisations’ commitment to processes and tools. Furthermore, Agile processes recommend that the Working Software principle discussed in Section 2.6.3 above is preferable to documentation in terms of achieving requirements satisfaction. However, DO-178C requires documented evidence that process objectives relating to requirements have been satisfied. Coe and Kulick also observed that, whereas planning activities are part of Agile processes, the key element required for DO-178C is evidence for certification. Lotfi [58] reduced that conflict by preserving the Agile Values through the process. They introduced a Model-Based Agile process, which combined Agile development processes and model-based engineering methodologies to satisfy a FAA-mandated process [83]. The goal of the Model-Based Agile process is to help promote the use of Agile methods in the field of safety-critical systems.

Stålhane et al [45] proposed the “Safe Scrum” approach. This was motivated by the need to make it possible to use methods that are flexible with respect to planning, documentation and specification while still adhering to the safety standard IEC 61508. Stålhane illustrates two kinds of Product Backlog within SafeScrum: the Functional Product Backlog and Safety Product Backlog.

These Backlogs are linked to show dependencies between the functional requirements and the safety requirements. The work of Guo and Hirschmann [75] expresses similar ideas. The key difference concerns how the Product Backlog is managed. They claim that a Safety Manager will oversee safety assurance throughout the whole process. In addition, the Safety Manager will check the Product Backlog for certain safety goals and ensure proper ranking and arrangement in the Release Plan. Whilst there are obvious potential opportunities to extend this work to include safety case development (e.g. linking the documentation and traceability of safety goals with an explicit safety argument development) there is no discussion of safety case development in this work.

2.8.2 Integrating agile practices with an assurance case

To date, there has not been any work which directly addresses the integration of Agile principles and methods in assurance cases for *safety* critical-systems. However, Lotfi, et al [58,59] illustrate a method for the iterative development of “*security*” features within the Scrum method which uses security assurance cases as a “*technique for ensuring fulfilment of the security requirements of the feature*”. Their findings are broadly positive: they claim that the approach helps to minimize the cost of assurance of software security, and reduces the costs of mitigating threads. However, they do indicate some barriers to adoption of the method: as presented, the technique is not sufficiently scalable, and is costly because of the rework it requires. Furthermore, the authors note that “*assumption limits the security reassurance of a new increment to the re-evaluation of a set of claims associated with the components*”. Lotfi et al conclude, “that the agile software development approach does not prevent ensuring the security of software increments produced at the end of each iteration”. The process they define is described in Figure 12:

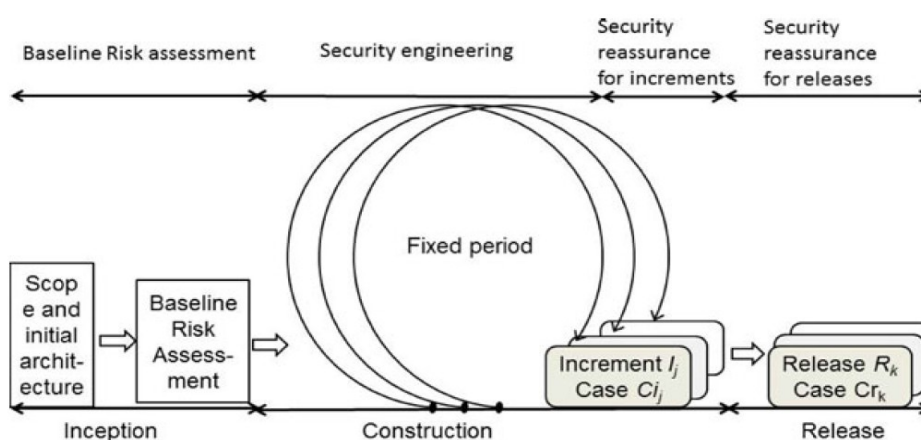


Figure 12: The Agile Software Development Process [58]

Other researchers have examined how conventional methods and techniques used for security assurance suit Agile methodologies. Beznosov and Kruchten [60] evaluated how well security assurance practices match the typical practices of Agile methods. For example, they identify that the informal review practices of agile methods match well, but that many agile approaches lack the external review and formal validation required for security. In addition, they identify that many security practices are independent of the adoption of agile practices – e.g. the use of security design principles. Similarly, Lotfi [59] (as described earlier in Section 2.8) found that some security assurance practices can be easily integrated. In particular, [53] begins to describe how security assurance cases can be used within Scrum to help iteratively develop security features.

Finnegan and McCaffery [61] introduced a work in the field of medical devices security assurance within the regulated software research center. They developed a framework to meet the requirements of this regulatory guidance through the use of assurance cases. They then used this framework to influence on a number of security related standards” six standards”, through the adaptation of Goal Structure Notation (GSN) argument pattern. This security argument pattern will facilitate to the regulators and healthcare organisations with a link between the security and risks, by mapping the six standards to 19 security capabilities to facilitate traceability for the regulators. Similar to Fitzgerald, et al [50] presented on how agile methods can be scaled to regulated environment.

Finnegan and McCaffery [61] introduced work in the field of security assurance for medical devices carried out at the Regulated Software Research Center. They developed a framework to meet the requirements of regulatory guidance in that domain through the use of assurance cases. They then used this framework to influence a number of security related standards (“six standards” ISO 27799, ISO/IEC 27002, IEC 62443-3, NIST SP 800-53, ISO/IEC, 15408-2, ISO/IEC 15408-3.).

Their assurance approach is documented through the use of a Goal Structuring Notation (GSN) argument pattern. This security argument pattern is intended to provide regulators and healthcare organisations with a means to link security measures and risks, by mapping the objectives of the six standards to 19 security capabilities to facilitate traceability for the regulators.

2.9 Summary

In this chapter we have explained some of the fundamental concepts and practices relating to safety assurance and Agile methods in order to provide context for the remainder of the thesis. Moreover, we have presented an overview of the literature relating to the integration of Agile within safety-critical systems, highlighting previous successes, challenges and obstacles faced by the researchers.

In the literature Stålhane et al [85] illustrate some of difficulties perceived in the integration of agile development and safety case. Stålhane describes the concern that safety case development does not directly contribute to the development of running software and only indirectly contributes to the test and verification activities. They state that, "Documentation produced simply for the sake of compliance or assurance is considered by some to run counter to the agile manifesto's idea of customer focus. Reuse of documents and use of document templates, however, can reduce the extra effort needed for building a safety case. Working with the safety case offers the potential to increase system understanding and will thus lead to a more efficient process" [85].

However our work focuses on the satisfaction of 4+1 principles within an agile framework "Scrum". Specially, the focus of the work is on assurance rather than being seen to comply with a specific safety standard.

Chapter 3

Survey

3.1 Introduction

In this chapter we describe the results of our survey designed to elicit the opinions of safety practitioners as to the challenges and opportunities posed by the application of agile development methods in the field of safety critical systems development. In particular, the survey explored the relationship between three key activities in safety engineering and an agile approach – namely, safety requirements development, hazard analysis, and safety case development. The results of this survey are presented together with brief discussion of the implications for integration.

3.2 Importance of research

As outlined in the literature survey in Chapter 2, despite progress in the use of agile development methods in safety critical systems development (e.g. [45,50,58], there are still those with doubts about the potential for successful integration. There are also reported experiences [43] that highlight the complementary nature of the iterative and incremental approach underlying many agile methods and recognised best practice in risk management in safety critical systems development. Rather than start with a theoretical evaluation of the compatibility of the principles of agile development with software safety assurance, we decided to draw out these experiences, opinions (and possibly preconceptions) by means of a practitioner survey. In particular, the survey attempted to draw out specific responses relating to the (possible) incremental and iterative nature of safety requirements development, hazard analysis and safety (assurance) case development.

3.3 Survey Design

The first section (8 questions) of the survey was designed to elicit information about the respondents: job role, knowledge and experience of safety critical systems engineering





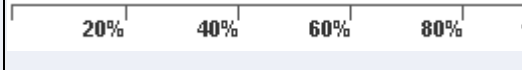
practice and standards, and knowledge and experience of agile methods. The main body of the survey – which is presented in the following section - was designed as a series of statements about the potential integration of safety engineering activities with agile development with respondents prompted for responses on a five-point Likert scale. To avoid biasing responses, both positive and negative statements about possible integration were included. In addition, some statements were designed with deliberate overlap and/or contradiction to check the consistency of the respondent’s answers.

3.3.1 Survey findings





This section displays and discussing the themes in the survey with graphs for illustration. At the end of this chapter, we will highlight the most important themes, which emerged from this part of the research.

Q9. Please rate the following statement: “Agile Methods can be integrated with Safety Critical Systems Development”.				
Responses	Count	Assigned Weight	%	Percentage of total respondents
Strongly disagree	1	1	3.23%	
Mildly disagree	2	2	6.45%	
Neither agree or disagree	3	3	9.68%	
Mildly agree	12	4	38.71%	
Strongly agree	13	5	41.94%	
Weighted Score : 4.10				
Total Responses	31			

Q9. We asked whether respondents thought that Agile methods could be integrated with safety- critical systems development. 41.94% of respondents Strongly Agreed, slightly followed by the number who Mildly Agreed 38.71%, these respondents indicated significant approval of integration between agile methods and safety critical systems. On the contrary, the figures for Nether Agree or Disagree 9.68%, Mildly Disagree 6.45%, Strongly Disagree 3.23% were very low. The result was an overwhelmingly positive response (80% Agree or Strongly Agree). That indicates an enthusiasm for integration.




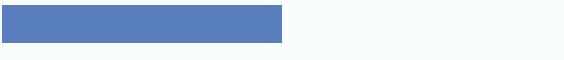

Q10. Please rate the following statement: "Hazard Analysis is an ongoing activity that must be applied throughout the lifecycle".				
Responses	Count	Assigned Weight	%	Percentage of total respondents
Strongly disagree	1	1	3.23%	
Mildly disagree	0	2	0%	
Neither agree or disagree	1	3	3.23%	
Mildly agree	8	4	25.81%	
Strongly agree	21	5	67.74%	
Weighted Score : 4.55				
Total Responses	31			

Q10. We asked whether respondents agree that hazard analysis is an ongoing activity that must be applied throughout the lifecycle. The majority of respondents Strongly Agree with that statement 67.74%, a little over quarter 25.81% Mildly Agree. Neither Agree or Disagree 3.23%, Strongly Disagree 3.23% and 0% Mildly Disagree. The final result strongly supported the notion that these activities are iterative and incremental, and need to be revisited throughout software development.

Q11. Please rate the following statement: "As well as making a safe product, it is necessary to provide assurance of safety (for others)".				
Responses	Count	Assigned Weight	%	Percentage of total respondents
Strongly disagree	0	1	0%	
Mildly disagree	0	2	0%	
Neither agree or disagree	2	3	6.45%	
Mildly agree	6	4	19.35%	
Strongly agree	23	5	74.19%	
Weighted Score : 4.68				
Total Responses	31			

Q11. "As well as making a safe product, it is necessary to provide assurance of safety (for others)"? Another interesting result is that 74.19% of the survey respondents Strongly Agree. 19.35% with this statement. The important point here is that there is strong recognition that the primary concern for safety engineering is not about just making a product safe, it's also about how we convince

others that the product is safe – i.e. assurance. The significance of this response is that it relates to the issue of demonstration of safety, e.g. with documentation, that others can independently review and evaluate. This response suggests that there is a need to establish an explicit safety case that be used to present the arguments and evidence of achieved safety to others.

Q12. Please rate the following statement: "Compliance with standards is important".				
Responses	Count	Assigned Weight	%	Percentage of total respondents
Strongly disagree	0	1	0%	
Mildly disagree	1	2	3.33%	
Neither agree or disagree	6	3	20.00%	
Mildly agree	8	4	26.67%	
Strongly agree	15	5	50.00%	
Weighted Score : 4.23				
Total Responses	30			

Q12. Half of the respondents 50% Strongly Agree that compliance with standards is important, followed by slightly more than quarter 26.67%, Neither Agree or Disagree 20% and Mildly Disagree 3.23%. Overall, there is an overwhelmingly positive response 77% Agree or Strongly Agree, indicating the practitioners acknowledge the importance of the standards that prevail over the development and assurance of safety critical systems.

Q13. Please rate the following statement: "Traceability (e.g. of safety requirements) is				
Responses	Count	Assigned Weight	%	Percentage of total respondents





Strongly disagree	0	1	0%	
Mildly disagree	0	2	0%	
Neither agree or disagree	3	3	9.68%	
Mildly agree	4	4	12.90%	
Strongly agree	24	5	77.42%	
Weighted Score: 4.68				
Total Responses	31			

Q13. The majority of our respondents Strongly Agreed that traceability for safety requirement is important for software development projects 77.42%, and Mildly Agree scored 12.90%. By contrast, only 9.68% of responses were on the Neither Agree nor Disagree survey scale and both Mildly Disagree and Strongly Disagree scored 0%. Again, this is an overwhelmingly positive response that recognizes the importance of traceability 90%. This is a specific illustration of the point about assurance that was raised in the response to Q11, in that traceability provides a means for, being able to demonstrate how requirements are decomposed and allocated.

Q14. Please rate the following statement: "Software safety assurance requires system level knowledge".				
Responses	Count	Assigned Weight	%	Percentage of total respondents
Strongly disagree	0	1	0%	
Mildly disagree	0	2	0%	
Neither agree or disagree	1	3	3.23%	
Mildly agree	8	4	25.81%	
Strongly agree	22	5	70.97%	
Weighted Score: 4.68				
Total Responses	31			

Q14. There is a clear convergence between this question and the previous one. Our respondents appear to be Strongly Agreed 70.97% that software safety assurance requires

system level knowledge. Mildly Agree scored 25.81%, and Neither Agree nor Disagree 3.23%. Again, the answers reveal overwhelming support for the statement that system level knowledge is essential for the safety assurance of software 96%. Whilst this is true, it should be noted that it relates to one of the biggest problems encountered in safety critical software development, that it is hard (and potentially unsafe) to develop software without system level safety knowledge.

Q15. Please rate the following statement: "Safety case development is an ongoing activity that must be applied throughout the lifecycle".				
Responses	Count	Assigned Weight	%	Percentage of total respondents
Strongly disagree	0	1	0%	
Mildly disagree	1	2	3.23%	
Neither agree or disagree	3	3	9.68%	
Mildly agree	9	4	29.03%	
Strongly agree		5	58.06%	
Weighted Score: 4.42				
Total Responses	31			

Q15. This is one of the most interesting set of responses in the survey. 58.06% of the responses to affirm the importance of safety case development as an ongoing activity which must be applied throughout the lifecycle. This is followed by Mildly Agree with a score of 29.03% and Neither Agree or Disagree with a score of 9.68%. The lowest score was for Mildly Disagree 3.23%. These answers are an indication of the importance of safety case development. The responses suggest approximately 90% overall support for this concept.





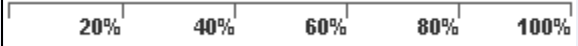
Q16. Please rate the following statement: "Regular evaluation of safety assurance / safety case progress should take place throughout the lifecycle".				
Responses	Count	Assigned Weight	%	Percentage of total respondents
Strongly disagree	0	1	0%	

Mildly disagree	2	2	6.45%	
Neither agree or disagree	1	3	3.23%	
Mildly agree	8	4	25.81%	
Strongly agree	20	5	64.52%	
Weighted Score: 4.48				
Total Responses	31			


Q16. Another affirmation of the importance of safety assurance and the safety case. We asked whether regular evaluation of safety assurance and safety case progress should take place throughout the lifecycle. Around 64.52% of the respondents consider that regular evaluation of safety assurance and the safety case need to take place throughout the lifecycle, as demonstrated by their support for Strongly Agree on the survey scale. 25.81% of the respondent Mildly Agree, 6.45% Mildly Disagree, and finally a small percentage 3.23% Neither Agree or Disagree with the statement. The most obvious position in the system lifecycle for review of the assurance case position is “preoperational” – i.e. just prior to the system being approved to enter service. However, staged safety case review (alongside staged production of the assurance case, as advocated in [1] and [2]) is far less risky (in project risk terms). If there are problems with the arguments and evidence being offered up by the assurance case it is desirable to find this out as early as possible in the lifecycle.




Q17. Please rate the following statement: "Safety critical systems development should involve regular contact with the regulator / acceptance authority".				
Responses	Count	Assigned Weight	%	Percentage of total
1 - 1	1	1	3.23%	
2 - 2	1	2	3.23%	
3 - 3	7	3	22.58%	
4 - 4	6	4	19.35%	
5 - 5	16	5	51.61%	
Weighted Score : 4.13				
Total Responses	31			

Q17. A little over half of the respondents 51.61% Strongly Agree with the notion that safety critical systems development should involve regular contact with the regulator / acceptance authority. 22.58% of the respondents Neither Agree or Disagree, followed by 19.35% who Mildly Agree. Finally Mildly Disagree and Strongly Disagree each score the same 3.23%.





Q18. Please rate the following statement: "Safety assurance and certification difficulties discovered late in the lifecycle (e.g. failure to establish an acceptable safety case) have resulted in significant cost and time overruns".				
Responses	Count	Assigned Weight	%	Percentage of total respondents
1 - 1	0	1	0%	
2 - 2	1	2	3.23%	
3 - 3	1	3	3.23%	
4 - 4	4	4	12.90%	
5 - 5	25	5	80.65%	
Weighted Score : 4.71				
Total Responses	31			

Q18. An overwhelming majority 80.65% of the respondents Strongly Agree with the following statement: "Safety assurance and certification difficulties discovered late in the lifecycle (e.g. failure to establish an acceptable safety case) have resulted in significant cost and time overruns". 12.90% Mildly Agree, and Mildly Disagree and Strongly Disagree each score 3.23%. The results therefore indicate strong overall support for this statement (>90%). This highlights the importance of “early and often” approaches to getting feedback on the development of an acceptable safety case.





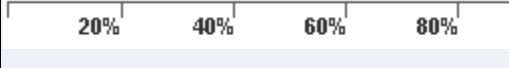
Q19. Please rate the following statement: "Software Safety Requirements can be defined once, at the beginning of the project, then used (unchanged) through the software development and assurance activity".				
Responses	Count	Assigned Weight	%	Percentage of total respondents
Strongly disagree	20	1	64.52%	

Mildly disagree	9	2	29.03%	
Neither agree or disagree	0	3	0%	
Mildly agree	2	4	6.45%	
Strongly agree	0	5	0%	
Weighted Score: 1.48				
Total Responses	31			





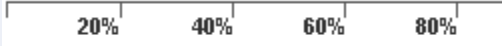
Q19. The most interesting part of this survey is that 64.52% of our respondents Strongly Disagree with the idea that Software Safety Requirements can be defined once, at the beginning of the project, then used (unchanged) through the software development and assurance activity. 29.03% mildly disagree, 0% Neither Agree or Disagree and 0% Strongly Agree. 6.45% of respondents Mildly Agree with the statement. The strong disagreement with this statement challenges any view of safety requirements that is static through the process. It challenges the view that requirements can be defined (principle 1) to cover hazards but then unaltered. In addition, the disagreement here is consistent with the strong agreement with the statement proposed in Q10 (Hazard Analysis is an ongoing activity that must be applied throughout the lifecycle). The response here highlights the fact that we cannot simply establish a safety requirements backlog at the beginning of software development and expect this to be unchanged through the software development process.

Q20. Please rate the following statement: "Compliance with standards is the primary objective of software safety assurance".				
Responses	Count	Assigned Weight	%	Percentage of total respondents
Strongly disagree	15	1	50.00%	
Mildly disagree	7	2	23.33%	
Neither agree or disagree	5	3	16.67%	
Mildly agree	3	4	10.00%	
Strongly agree	0	5	0%	
Weighted Score: 1.87				
Total Responses	30			






Q20. 50% of the respondents Strongly Disagree that compliance with standards is the primary objective of software safety assurance, 23.33% Mildly Disagree, 16.67% Neither Agree or Disagree, 10% Mildly Agree and, unexpectedly, 0% Strongly Agree. The strong disagreement with this statement highlights the view that whilst standards are important (note the strong positive response to Q12) they are not the primary objective of safety critical systems development, which could (for example) be argued to be concerned with building a safe product. This suggests that our approach to integrating safety critical systems development and Agile should be aware of, and demonstrate compliance with standards (such as IEC 61508). We must, more importantly, address how integrated processes help us establish compelling arguments (and evidence) of safety.

Q21. Please rate the following statement: "The interface and interaction between software engineers and safety engineers is				
Responses	Count	Assigned Weight	%	Percentage of total respondents
Strongly disagree	6	1	19.35%	
Mildly disagree	13	2	41.94%	
Neither agree or disagree	11	3	35.48%	
Mildly agree	1	4	3.23%	
Strongly agree	0	5	0%	
Weighted Score : 2.23				
Total Responses	31			

Q21. As we can see from the results for this question, 41.94% Mildly Disagree with the statement that the interaction between software engineers and safety engineers is not well managed, followed by 35.48% who Neither Agree or Disagree and 19.35% who Strongly Disagree. By contrast, Mildly Agree scores 3.23% and Strongly Agree. The responses are therefore basically neutral to negative, showing that this is an area for improvement. If we get the ‘customer’ stakeholder involvement part of Agile right then this is an opportunity to improve on this experience.

Q22. Please rate the following statement: "Safety case development only needs to be considered at the end of the lifecycle after development is complete".				
Responses	Count	Assigned Weight	%	Percentage of total respondents
Strongly disagree	24	1	77.42%	
Mildly disagree	5	2	16.13%	
Neither agree or disagree	1	3	3.23%	
Mildly agree	0	4	0%	
Strongly agree	1	5	3.23%	
Weighted Score: 1.35				
Total Responses	31			

Q22. 77.42% of respondents Strongly Disagree with the statement that safety case development only needs to be considered at the end of the lifecycle after development is complete. 16.13% Mildly Disagree, 3.23% Neither Agree or Disagree, 3.23% Strongly Agree and 0% Mildly Agree. This answer is the opposite of that indicated for Q15 (Safety case development is an ongoing activity that must be applied throughout the lifecycle), which emphasizes the perceived importance of safety case development throughout the lifecycle

Q23. Please rate the following statement: "Safety problems, including safety assurance problems, are identified and managed early in the lifecycle as they arise".				
Responses	Count	Assigned Weight	%	Percentage of total respondents
Strongly disagree	1	1	3.23%	
Mildly disagree	11	2	35.48%	
Neither agree or disagree	5	3	16.13%	
Mildly agree	9	4	29.03%	
Strongly agree	5	5	16.13%	
Weighted Score : 3.19				

Total Responses	31	20%	40%	60%	80%
-----------------	----	-----	-----	-----	-----

Q23. Responses to this statement are much more spread – a variety of experiences is clearly being hinted at. Just over a third of the participants 35.48% Mildly Disagree with the statement here, as compared with just under a third 29.03% who Mildly Agree. Two categories (Strongly Agree and Neither Agree or Disagree) have the same figure 16.13%, and 3.23% Strongly Disagree. So there is clearly some recognition that safety problems aren't always identified early in the lifecycle. The fact that there are some positive indications and some negative ones from the spread of responses implies that there is room for improvement in the early identification of safety and assurance problems. Incremental development of assurance cases offers one potential means to supply this improvement.






Q24. Please rate the following statement: "The level of effort spent in the design and assurance of a safety-critical system should be commensurate (in balance) with the level of risk posed by that system".				
Responses	Count	Assigned Weight	%	Percentage of total respondents
Strongly disagree	1	1	3.23%	
Mildly disagree	1	2	3.23%	
Neither agree or disagree	3	3	9.68%	
Mildly agree	12	4	38.71%	
Strongly agree	14	5	45.16%	
Weighted Score : 4.19				
Total Responses	31			20% 40% 60% 80% 100%

Q24. As we see, 45.16% of respondents Strongly Agree with the statement, followed 38.71% who Mildly Agree and 9.68% who Neither Agree or Disagree. By contrast, Mildly Disagree and Strongly Disagree each score 3.23%. Prioritisation of assurance effort (according to risk) was thus recognized as important by 84% of the respondents. Together with the observations regarding evaluation, this suggests that incremental safety case development may be useful in helping guide an incremental software development process






Q25. Which of the following are recognised as features of agile development?			
Responses	Count	%	Percentage of total respondents
Active customer involvement is imperative	26	83.87%	
The development team must be empowered to make decisions	28	90.32%	
Requirements evolve but the timescale is fixed	15	48.39%	
Capture requirements at a high level; lightweight & visual	8	25.81%	
Develop small, incremental releases and iterate	31	100.00%	
Focus on frequent delivery of products	21	67.74%	
Complete each feature before moving on to the next	11	35.48%	
Testing is integrated throughout the project lifecycle – test early and often	29	93.55%	
A collaborative & cooperative approach between all stakeholders is essential	28	90.32%	
Apply the 80/20 rule	4	12.90%	
None	0	0%	
Total Responses	201		

Multiple answers per participant possible. Percentages added may exceed 100 since a participant may select more than one answer for this question.

Q25.This question was included specifically to check how much experience and knowledge participants had of Agile methods, specifically practices and principles. As can be seen from the table above, participants have some knowledge on agile practices, but knowledge is estimated through exercise and experience.






Q26. Please rate the following statement: "The process of software development is less important than the finished software product".				
Responses	Count	Assigned Weight	%	Percentage of total respondents
Strongly disagree	5	1	16.13%	
Mildly disagree	7	2	22.58%	
Neither agree or disagree	7	3	22.58%	
Mildly agree	6	4	19.35%	
Strongly agree	6	5	19.35%	
Weighted Score : 3.03				
Total Responses	31			

Q26. As we can see from the table above, that the participants' answers indicate a spread of different opinions here. This indicates that there is no unified opinion as to whether the software development process is less important than the finished software product.

Q27. Please rate the following statement: "Documentation should be minimised during software development".				
Responses	Count	Assigned Weight	%	Percentage of total respondents
Strongly disagree	5	1	16.13%	
Mildly disagree	10	2	32.26%	
Neither agree or disagree	7	3	22.58%	
Mildly agree	4	4	12.90%	
Strongly agree	5	5	16.13%	
Weighted Score : 2.81				










Total Responses	31	
-----------------	----	--

Q27. More than a third 32.26% Mildly Disagree with the idea of minimized documentation during software development posed in Q27. After that comes Neither Agree nor Disagree 22.58% followed by both Strongly Agree and Strongly Disagree, each of which has 16.13%. Mildly Agree, comes in last at 12.90% This indicates the importance of documentation.






Q28. Please rate the following statement: "Requirements should be addressed according to their priority".				
Responses	Count	Assigned Weight	%	Percentage of total respondents
Strongly disagree	0	1	0%	
Mildly disagree	3	2	10.00%	
Neither agree or disagree	4	3	13.33%	
Mildly agree	15	4	50.00%	
Strongly agree	8	5	26.67%	
Weighted Score : 3.93				
Total Responses	30			







Q28. Half of the participants 50.00% Mildly Agree and 26.67% Strongly Agree with the statement that, requirements should be addressed according to their priority. In total, 77% of respondents indicate broad agreement with the principle of requirements prioritization, which is a fundamental principle of Agile methods: developers are given space and freedom in the choice of tasks. 13.33% of respondents Neither Agree or Disagree with the statement and 10.00% Mildly Disagree. As we can see above the practitioners reject the idea that documentation should be minimised during software development.

Q29. Which of the following practices of agile development can contribute safety-critical systems development and assurance?			
Responses	Count	%	Percentage of total respondents
None (Please leave a comment)	0	0%	


All above	6	33.33%	
User stories	7	38.89%	
Small/short releases	8	44.44%	
Simple design	12	66.67%	
Refactoring	6	33.33%	
Continuous Integration	11	61.11%	
Pair Programming	9	50.00%	
Release Planning	10	55.56%	
Other (please specify)	0	0%	
Total Responses	69		
Multiple answers per participant possible. Percentages added may exceed 100 since a participant may select more than one answer for this question.			

Q29. The responses indicate clear support for the need for agile practices and principles, at the forefront came Simple Design with score of 66.67%. Continuous Integration scored 61.11%, followed by Release Planning with 55.56%. Half of the participants (50.00%) prefer to use Pair Programming, less than half 44.44% indicate support for Small/short releases, 38.98% for User Stories 38.89%. 33.33% agreed that all of the principles and practices listed are important, and the same proportion support Refactoring.

Q30. Which one of the following features of agile development could be in conflict with safety critical systems? (If you chose one or more please leave a comment)			
Responses	Count	%	Percentage of total respondents
Active user involvement is imperative	3	9.68%	
The team must be empowered to make decisions	5	16.13%	
Requirements evolve but the timescale is fixed	13	41.94%	
Capture requirements at a high level; lightweight & visual	10	32.26%	
Develop small, incremental releases and iterate	4	12.90%	

Focus on frequent delivery of products	10	32.26%	
Complete each feature before moving on to the next	7	22.58%	
Testing is integrated throughout the project lifecycle – test early and often	0	0%	
A collaborative & cooperative approach between all stakeholders is essential	2	6.45%	
Apply the 80/20 rule	12	38.71%	
None	4	12.90%	
Other (please specify)	9	29.03%	
Total Responses	79		
Multiple answers per participant possible. Percentages added may exceed 100 since a participant may select more than one answer for this question.			





Q30. We asked which Agile features could be in conflict with safety-critical systems. The participants' answers indicate their concern with fixed timescales 41.94%, with applying the 80/20 rule 38.71%. Around a third of responses express concern about requirements at a high level, and the focus on frequent delivery of products (32.26% for each of these) 22.58% of responses indicate concern with the need to complete each feature before moving on to the next. Some of the participants agreed that the notion that the team must be empowered to make decisions is potentially in conflict with safety critical systems principles. 12.90% feel that developing in small, incremental releases and iterating these is in conflict with safety critical systems principles. However, 12.90% believe that none of factors indicated in the question necessarily conflicts with safety-critical systems principles. Finally, 9.68% believed that active user involvement in development as a principle conflicts with safety critical systems methodologies.

Q31. Which one of these practices could be beneficial for safety-critical systems development and assurance?			
Responses	Count	%	Percentage of total respondents
Active user involvement is imperative	21	67.74%	

The team must be empowered to make decisions	17	54.84%	
Requirements evolve but the timescale is fixed	10	32.26%	
Capture requirements at a high level; lightweight & visual	9	29.03%	
Develop small, incremental releases and iterate	18	58.06%	
Focus on frequent delivery of products	14	45.16%	
Complete each feature before moving on to the next	7	22.58%	
Testing is integrated throughout the project lifecycle – test early and often	29	93.55%	
A collaborative & cooperative approach between all stakeholders is essential	22	70.97%	
Apply the 80/20 rule	1	3.23%	
None	0	0%	
Total Responses	148		
Multiple answers per participant possible. Percentages added may exceed 100 since a participant may select more than one answer for this question.			

Q31. The question here is the exact opposite of the previous question. Here, we found that the participants indicated that some Agile practices could potentially contribute to and be beneficial for safety-critical systems. 93.55% agreed that the principle of testing being integrated throughout the project lifecycle – test early and often - could be beneficial for safety-critical systems development and assurance. 70.97% agreed that a collaborative & cooperative approach between all stakeholders is essential for safety-critical systems. 58.06% felt that developing in small, incremental releases and iterating these was potentially beneficial.

Q32. Which of the following statements best represents your overall opinion regarding agile development methods and safety?			
Responses	Count	%	Percentage of total respondents

Agile development methods are in conflict with safety critical systems development and assurance	2	6.67%	
Agile development methods can sometimes be in conflict, and sometimes in harmony with safety critical systems development and assurance (add comment below)	14	46.67%	
Agile development methods can be in harmony with safety critical systems development and assurance	14	46.67%	
Other (please specify)	12	40.00%	
(Did not answer)	0	0%	
Total Responses	42		
Multiple answers per participant possible. Percentages added may exceed 100 since a participant may select more than one answer for this question.			

Q32. The results indicate a spread of opinion here: 46.67% of participants were agreed that Agile development methods can be in conflict, and occasionally in harmony with safety critical systems development and assurance, and the same proportion 46.67% agreed that Agile development methods can be in harmony with safety critical systems development and assurance, 6.67% agreed that Agile development methods are in conflict with safety critical systems development and assurance. 40.00% of responses indicated an “Other” response. There were a variety of additional comments made by those stating ‘Other’, including the following:

“Depending on how you apply agile method. Must still provide documentation as evidence.”

“I believe they can be in harmony if one considers the essential complexity of the problem space. Agile methods help us address complexity, but they should not be applying dogmatically in the same manner as other non safety-critical domains. Likewise, they should not be discounted due to dogmatic thinking in w.r.t. [with respect to] traditional SE [Software Engineering] methods - that one has to do it only the same way it has been done before”

“Work is needed to put them into harmony, but it can be done.”

“The standard agile practices are fine but must be supplemented with additional practices for safety critical systems”

“Only when project management acknowledges iterative software and requirements evolution, the typical conflict between safety and software development can be avoided.”

“Agile is an attitude, not an ideology”

“It depends on the complexity of the project/product/service and also the ability of the teams/stakeholders and also timescales.”

“The agile culture encourages people to hack. Agile methods adapted to safety critical projects often just look like normal monolithic methods which are mostly what has to be done”

“Depends on the external assessor and the PoC [Points of Contact] he's provided”

“If they seem in conflict its through thinking too literally about popular branded agile methodologies and a shallow understanding.”

“The mindset must be adjusted as to how/what to test and form documentation can take”

“Agile can support building safety critical systems and ease to build them - compared to using other methods. Because of the ongoing attention to it.”

3.4 Results and Discussion

31 people completed the questionnaire. A broad range of industrial sectors was represented. We selected the participants from different roles, to get as broad a representation of practitioners as possible with respondents from health, aerospace automotive, railway, and the process sector. The majority of respondents represented organisations involving 150 people or more. 87% had practical experience of safety critical system development. 77% had practical experience of the application of agile development methods. A broad range of agile methods were cited as the basis for this practical experience, with the most popular method (76%) being Scrum. The limited case study sample meant that not enough data was collected from only 31 practitioners, and more than 31 practitioners are needed. We had considerable difficulty finding practitioners who were sufficiently expert in both Agile and safety. The majority of our participants were based in the Germany, the USA Sweden, Norway, UK, and Italy.

At the outset of the survey 80% either agreed or strongly agreed that agile methods can be integrated with safety critical systems development. Answers to the final survey question shows a more nuanced response with 48% believing agile and safety methods to be in harmony, and 45% believing that some agile and safety practices are in harmony whilst some are in conflict practices are in harmony whilst some are in conflict. The following themes emerge from the responses to individual questions:

The response to statements regarding the processes associated safety requirements elicitation, hazard analysis, assurance case development and evaluation strongly supported the notion that these activities are iterative and incremental, and need to be revisited throughout software development. This challenges the view that safety requirements and hazard analysis can largely be performed outside of the sprint / incremental cycle

- The need to provide assurance of product safety, as well as making the product safe was strongly recognized (>95%). This indicates the importance of establishing the safety case alongside development.
- Whilst compliance with standards was recognized as important (50%) it was also recognized (>73%) as not being the primary objective of software safety assurance.
- 75% recognized that software safety assurance should involve system level knowledge and engineers. (70%) also recognized the importance of involving regulators and evaluators during software development. This helps inform the definition of the traditional agile ‘customer’ role when applied in a safety critical

systems development context.

- There was recognition (50%) that safety problems aren't always identified early in the lifecycle. There was strong support (90%) for regular evaluation of safety case progress during development. Again, this suggests safety case development should be treated as an 'in-increment' activity.
- Prioritisation of requirements (77%) and assurance effort (84%) (according to risk) was recognized as important. Together with the observations regarding evaluation, this suggests that incremental safety case development may be useful in helping guide an incremental software development process.

Chapter 4

Integrating Software Safety Assurance Principles with Scrum

4.1 Introduction

In this chapter, we present an initial proposal as to how Scrum can be modified to address the 4+1 principles of [84] safety assurance. We discuss the 4+1 principles in detail in section 4.2, and provide suggestions for the modification of Scrum targeted to each principle in turn. The following ‘design criteria’ motivated these modifications: change as little as necessary, keep agility as far as possible, do not compromise on safety assurance. We believe that the use of Agile in the field of safety critical systems is likely to present a number of challenges. These challenges, and our recommendations for how to address them, are also discussed in this chapter.

Agile development traditionally promotes the view that the delivery of working software to the customer is the primary measure of progress [40]. However, for safety critical systems development it needs to be clear that the definition of ‘working’ software implies safety assured software. If the safety of the software isn’t assured, it cannot be considered as working software, because it cannot be used. Therefore our primary measures of progress (for safety critical software) must take into account both the status of the software and the status of the software’s safety assurance. As discussed in Chapter 2, Scrum is an agile development approach that uses an iterative, incremental process, and is based on a small set of core values, principles, and practices [15]. Scrum does not explicitly address software safety assurance. e software safety assurance into Scrum.

4.2 The 4+1 challenges and recommendation

The 4+1 Principles were developed by Kelly et al [76]. The value of using the 4+1 principles as the basis for this project is that the principles capture the spirit, or intention, of the standards, which provides us with a more abstract basis for attempting a modification / integration with Scrum than would a specific safety standard. These principles are constant across domains and across projects, and can be regarded as the immutable core of any software safety justification. The principles also help maintain

understanding of the ‘big picture’ of software safety issues whilst examining and negotiating the detail of individual standards, and provide a reference model for cross-sector certification.

4.2.1 Software Safety Assurance Principle 1:

Software safety requirements shall be defined to address the software contribution to system hazards

Kelly provides the following description of Principle 1 [76]:

“Software by itself cannot be safe or unsafe. It is only when placed in a system context that the ‘safety’ of software can be judged by considering the contribution that the software could make to system level hazards. For example, software can play a role as the initiator of a causal chain of events leading to a system level hazard (and eventual accident). Software (e.g. when placed in the role of a protection system) can also play a role by failing to mitigate failures of other (non- software) elements. The first challenge of software safety assurance is to identify all of the ways in which software can contribute to system level hazards and to capture the necessary behaviour of the software in relation to these contributions in terms of a clearly defined set of software safety requirements at the system software boundary. (Note these behavioural software safety requirements are distinct from the assurance requirements of software safety standards, e.g. to maintain traceability.)”

Challenges posed

- For assurance (e.g. certification), software safety requirements (and their treatment) must be easily identifiable within the total set of requirements being addressed.
- Safety requirements can be positive functional requirements in their own right, constraints to be satisfied (e.g. timing) when addressing other requirements, or negative requirements (i.e. behavior to not be exhibited)
- Safety requirements are therefore often associated and interrelated with core product requirements (i.e. requirements in the product backlog)
- Safety requirements should not be addressed as an ‘after thought’ following

implementation of core product requirements.

- Relevant (related) safety requirements need to be highlighted when addressing core requirements from the Product Backlog (so that they can be addressed as an integral of the implementation of these requirements).

Recommendations for Scrum

- A second Product Backlog – the Safety Backlog – should be maintained alongside the Product Backlog to explicitly store, manage and prioritise software safety requirements.
- The Safety Backlog should record traceability between safety requirements and system hazards
- The Safety Backlog should record traceability between safety requirements and ‘core’ Product Backlog requirements (e.g. between a safety constraint and the function to which it relates)
- The Safety Backlog is initially populated as a result of (deductive, ‘top down’) system level safety analysis performed prior to the software development commencing (i.e. analysis that identifies potential software contributions to system level hazards).
- The Safety Backlog should also be populated through performing (inductive, ‘bottom up’) hazard analysis (e.g. functional failure analysis) on the core Product Backlog requirements.
- Both the system level safety analysis, and the software hazard analysis should involve the following stakeholders: the system team (as ‘customer’ to the software team), the system customer (e.g. system operator), competent safety engineers capable of carrying out the analyses, the safety regulator (as safety ‘customer’) potentially supported by additional independent and internal safety reviewers (as proxies for the regulator).
- Maintenance of the Safety Backlog is the responsibility of a nominated safety team member from within the software team. This team member must ensure that they fully understand the source and rationale behind each of the requirements in the

safety backlog. (This may involve discussion with the stakeholders involved in the original safety and hazard analyses.)

- When product requirements are chosen for a given sprint, the related safety requirements from the Safety Product Backlog should be identified, discussed and explained (by the nominated safety team member) at the beginning of the sprint.
- One team member should be explicitly responsible for maintaining and explaining the safety product backlog.

4.2.2 Software Safety Assurance Principle 2:

The intent of the software safety requirements shall be maintained throughout requirements decomposition

Kelly provides the following description of Principle 2 [76].

As the software development lifecycle progresses, the requirements and design are progressively developed and a more detailed software design is created. Having established complete and correct software safety requirements at the highest (most abstract) level of design, it must be possible to demonstrate that the intent of those requirements is maintained as the software safety requirements are developed, reinterpreted, allocated and decomposed.

Challenges posed

- If software safety requirements are reinterpreted, allocated or decomposed during development, the new safety requirements that emerge should be documented
- If software safety requirements are reinterpreted, allocated or decomposed during development, traceability must be maintained between the original safety requirement and the new requirements or design that emerges.
- When new requirements or design emerge as a response to the original requirement they must be reviewed to check whether they fully address the intent of the original safety requirement.

- Justification must be maintained as to how the new requirements and address the original safety requirement.

Recommendations for Scrum

- If software safety requirements are reinterpreted, allocated or decomposed during a sprint, the new safety requirements that emerge should be documented in the Safety Product Backlog, with traceability to the original safety requirement
- As part of the Sprint review, those responsible for developing the original safety requirement must present the design (or refined requirements) solution proposed and new safety requirements that have emerged.
- The Sprint review should attempt to check whether the proposed solution fully addresses the intent of the original safety requirement.
- Independent stakeholders are required as part of this review. It should involve the nominated safety team member and potentially some or all of the following: the system team (as ‘customer’ to the software team), the system customer (e.g. system operator), a suitable internal safety reviewer (as proxy for the regulator’s position).
- If it is determined that more time is required to assess the adequacy of the proposed solution (than the sprint review allows) then a more thorough review will need to be allocated as an activity to the following Sprint. Whilst this activity is taking place, ideally further implementation of the solution should be avoided (i.e. other requirements from the Backlogs should be prioritized)
- The decomposition of the original safety requirement into the new safety requirements, and the justification as to the adequacy of this decomposition should be documented in the assurance case by the nominated safety team member. This should take place in the Sprint immediately following the review and will be presented as during the following Sprint review).

4.2.3 Software Safety Assurance Principle 3:

Software safety requirements shall be satisfied

Kelly provides the following description of Principle 3 [76].

Ultimately, it is necessary to demonstrate that any safety requirements allocated to software have been satisfied (i.e. present evidence to establish whether the required behaviour will occur in operation). It is important to present evidence that shows the satisfaction of safety requirements under anticipated operating conditions. For example, this requires presenting evidence that addresses satisfaction under both normal and abnormal (fault) conditions.

Challenges posed

- Verification evidence must be presented to demonstrate the satisfaction of any safety requirements claimed to be implemented.
- Explicit traceability must be maintained between verification evidence and implemented safety requirements.
- Verification evidence generated must be reviewed to check whether it fully addresses the intent of the safety requirement.
- Justification must be maintained in order to indicate how the verification evidence demonstrates satisfaction of the safety requirement.

Recommendations for Scrum

- Daily Scrum meetings can be used to provide an early indicator for monitoring progress towards safety requirements satisfaction.
- As part of Sprint review, those responsible for the implementation of a particular safety requirement must present the solution and explain how the intent of the safety requirement is fully satisfied.
- The Sprint review should attempt to check whether the implementation fully addresses the intent of the original safety requirement, and should review the verification evidence presented.
- Independent stakeholders are required as part of this review. It should involve the nominated safety team member and potentially some or all of the following: the system team (as ‘customer’ to the software team), the system customer (e.g. system operator), a suitable internal safety reviewer (as proxy for the regulator’s position).
- After initial inspection in the Sprint review if it is determined that more

verification evidence is required (e.g. to satisfy the verification criteria of a safety standard) then further verification will need to be allocated as an activity to the following Sprint. It may be necessary for this verification activity to be performed by independent team members or teams (according to Principle 4+1).

- The traceability of safety requirements to the verification evidence, and the justification as to the adequacy of the evidence set should be documented in the assurance case by the nominated safety team member. This should take place in the Sprint immediately following the review and will be presented as during the following Sprint review).

4.2.4 Software Safety Assurance Principle 4:

Hazardous behaviour of the software has been identified and mitigated

Kelly provides the following description of Principle 4 [76].

Whereas Principle 2 is concerned with maintaining the intent of safety requirements in the presence of increasing design commitment, Principle 4 is concerned with the potential undesirable and unintended consequences of design and implementation decisions. Principle 2 is concerned with whether lower levels of requirements and implementation do what is required (intended). Principle 4 is concerned with whether the design and implementation does anything else that is considered unsafe. These potentially emergent hazardous behaviours could firstly result from design decisions that have unintended hazardous side effects. Secondly, they can also result from implementation (process execution) errors during the software development process – e.g. modelling errors, coding errors, and tool-use errors. It is necessary to ensure that assurance effort has been targeted at attempting to reveal both of these sources of errors.

Challenges posed:

- Unintended hazardous behaviour can emerge as a result of the design and implementation of both safety and core requirements and therefore has to be identified in both cases.
- Where new hazardous behaviour is identified new safety requirements must be defined to manage this behaviour

- Identifying hazardous behaviour can require system-level domain expertise from outside of the software development team.

Recommendations for Scrum

- At the planning stage of a Scrum development the chosen modelling approach, implementation language, development environment and assessment tools should be documented. Hazard analysis should be conducted on the chosen languages, processes and tools to identify the potential for the introduction of implementation errors. Where the potential for error is identified it is necessary to provide evidence for non-introduction of error (e.g. tool qualification), controls for the minimization of error introduction (e.g. process guides) or means of demonstrating the absence of introduced errors (e.g. static analysis tools). Justification of the chosen strategies must be documented in the case.
- Daily Scrum meetings can be used to provide an early opportunity to identify unintended side-effects emerging from a chosen implementation approach.
- (Inductive, ‘bottom up’) Hazard analysis should be performed as part of Sprint review in order to identify hazardous side effects from the design and implementation activity undertaken during the Sprint. This hazard analysis should include a check of the implementation of the chosen strategy for dealing with implementation error introduction. This analysis should be performed in the context of the existing identified system level- hazards, but be open to the possibility of identifying new hazards.
- Independent stakeholders are required as part of this hazard analysis. It should involve the nominated safety team member and potentially some or all of the following: the system team (as ‘customer’ to the software team), the system customer (e.g. system operator), a suitable internal safety reviewer (as proxy for the regulator’s position).
- If it is determined that more time is required for the hazard analysis (than the Sprint review allows) then a more thorough review will need to be allocated as an activity to the following Sprint. Whilst this activity is taking place, ideally further implementation of the solution should be avoided (i.e. other requirements from the Backlogs should be prioritized)
- Where new hazardous behaviours (i.e. new software level contributions to system-

level hazards) are identified, new safety requirements (to manage these behaviours) must be added to the Safety Backlog.

- Where the Sprint activity and subsequent review relates to the implementation of safety requirements, the decomposition of the original safety requirement into the new safety requirements, and the justification of the identification and mitigation (through new safety requirements) of hazardous behavior should be documented in the assurance case by the nominated safety team member. This should take place in the Sprint immediately following the review and will be presented as during the following Sprint review).
- Where the Sprint activity and subsequent review relates to the implementation of core (i.e. non-safety specific) requirements, and new hazardous behavior is identified, and new safety requirements defined, the assurance case needs to be extended to include the new safety requirement.

4.2.5 Software Safety Assurance Principle 4+1:

The confidence established in addressing the software safety principles shall be commensurate to the contribution of the software to system risk

Kelly provides the following description of Principle 4+1 [76].

This principle is expressed as '+1', rather than principle number 5 because it underlies the implementation of the first 4 principles. Perfect assurance of the other four principles is unachievable. For example, it is impossible to prove that all hazards have been identified, and that all the necessary corresponding safety requirements have been identified. Consequently, we have to consider how much effort to expend in addressing the first four principles, and how much evidence to generate. We have to decide upon a sufficient level of evidence to present. This principle states that the level of evidence needs to be proportional to the level of risk associated with the software in question.

For a highly critical software-intensive system, the level of confidence in addressing the first four principles needs to be high. For a lower criticality system, the level of confidence can be lower.

Challenges posed

- To implement this principle it is necessary to be able to identify the criticality of any software safety requirement
- Implementing this principle will affect the amount of effort required, and the level of argument and evidence presented in the assurance case, to provide assurance of the first four principles. This can affect the length of Sprints, whether assurance activities take place alongside implementation activities within Sprints, or need their own Sprint Cycles to complete. This can also affect the type of stakeholder involvement required (e.g. whether independent verification or assessment is required, or when customer stakeholder involvement is sought). It will also influence the level of rigour in modelling and implementation techniques, and the level of qualification required in tools used within the process.
- Different standards have different approaches to defining required levels of assurance: IEC61508 sets a Safety Integrity Level according to the probability delta in risk reduction; DO-178C places more focus on severity; ISO 26262 incorporates the concept of controllability of the vehicle. In addition, there are many differences in the recommended techniques and processes at different levels of criticality. If conformance to these standards is sought, the specific requirements will need to be identified and a conformance argument established as part of the assurance case.

Recommendations for Scrum

- The Safety Backlog should record the criticality associated with each safety requirement. This could be expressed using the language of a particular safety standards (e.g. Safety Integrity Levels – SILs) or through risk attributes (e.g. the severity of the hazard associated with a requirement, and the degree of contribution of the requirement to the hazard).
- The criticality of Safety Backlog items can be used within Sprint planning to help manage project risk. For example, satisfactory implementation of high criticality safety requirements may be considered a project risk that should not be left until late Sprint releases.

- Before the Scrum development commences, it is necessary to decide how the assurance strategy will be moderated according to criticality. In particular, how the chosen approach to representing criticality will influence the following:
 - The amount and type of effort, stakeholder engagement, and assurance evidence required for the initial population of the Safety Backlog
 - Whether a dedicated safety team member is required within the software development team
 - The extent to which (system, regulator) customers are expected to be involved in the activities described in the first four principles.
 - The amount and type of effort, stakeholder engagement, and assurance evidence required in reviewing safety requirements development during Sprints (i.e. checking safety requirements validity)
 - The amount and type of effort, stakeholder engagement, and assurance evidence required in demonstrating satisfaction of safety requirements
 - The amount and type of effort, stakeholder engagement, and assurance evidence required in checking for implementation introduced errors and performing hazard analysis as part of the Sprint Cycles.
- The criticality of chosen requirements for the Sprint (or requirements related to those chosen) should be highlighted at the beginning of the Sprint. In particular, it should be highlighted where independence cannot be demonstrated between the implementation of requirements of differing criticality, the higher criticality should be used.

4.3 Summary and further work

This chapter has summarized the 4+1 principles of software safety assurance and discussed the challenges posed when applying 4+1 principles to Scrum. We have also made initial recommendations as to how the principles can be accommodated within a Scrum development. We believe that feedback on the proposed recommendation from safety and agile practitioners will help us to understand further the effective use of 4+1 within Scrum.

The purpose of the Safety Backlog is to keep all of the safety requirements in one place so that they can be explicitly identified and traced throughout the project development.

As well, the safety backlog is the focus of safety assurance and a safety case, whereas items in the normal backlog (concerning the normal functioning of the system) are not expected to need the same attention. We believe it is easier to track the relationship between 'core' requirements and associated safety requirements by maintaining two separate (but interrelated) backlogs. Also it's possible to examine the consistency of all safety requirements contained in the safety backlog to examine the safety requirements for potential inconsistency (e.g. conflicting requirements for safe behaviour).

In addition, the Safety Backlog forces a separation between safety requirements and other requirements. Safety requirements are usually stable (according to Stålhane [45]) but functional requirements will change to introduce new risks. When changes arise in the functional requirements, especially if changes are made to some aspect of the system which provides a safety barrier, it may be possible and desirable to update the safety requirements. Since safety requirements can introduce barriers to the system, it is wise to have them separately.

In the next chapter, we describe the results of semi-structured interviews with both Agile and safety practitioners to review the challenges and recommendations presented above. Beyond these interviews, it will be necessary to demonstrate how these recommendations can be applied through illustrative case study examples, however this lies outside the scope of a one-year Master's thesis.

Chapter 5

Feedback and Evaluation on the 4+1 Scrum Integration

5.1 Introduction

The primary focus of our work is to identify and address challenges associated with the integration of Agile methodologies into safety-critical systems development. The recommendations in the previous chapter seek to assist and encourage Scrum users in using Scrum in the field of safety critical systems project, while maintaining harmony with the core philosophy of the Agile approach. In this chapter, we report on work to gain feedback on the proposals made in the previous chapter.

In order to gain a deeper insight into the challenges identified for the integration of safety assurance into Scrum, and into the practicality of the recommendations made above, we conducted semi-structured interviews with safety engineers and Agile developers. It was decided that we should perform this validation step at this relatively early point in the development of the approach, because it was feasible given the short timescale of the Masters degree.

Participants were presented with an overview of the challenges we identified as being associated with applying the 4+1 software safety assurance principles to Scrum, together with our initial recommendations as to how the principles can be accommodated within a Scrum development. Participants were taken through a series of questions designed to gain feedback on the feasibility of the approach, and were asked for an assessment as to whether the 4+1 principles can be addressed without compromising the core principles of agility.

5.2 Aims of semi-structured interview process

This study is part of the research under the High Integrity System Engineering Group, Computer Science Department, of the University of York. This introduces the 4+1 Principles of Software Safety Assurance and their implications for Scrum, specifically, the

impact on the processes, roles and artifacts associated with Scrum development. Historically, there has been a reluctance to adopt agile methods within safety-critical systems development. However, feedback from our initial research in this area suggests that there are benefits to be gained from the application of agile methods to safety critical systems [74].

Following this feedback we have done further work to assess how the 4+1 principles of software safety assurance can be integrated with Scrum, and have developed an initial proposal for how Scrum could be modified to better address the principles.

5.3 Research questions and their motivations:

This study specifically furthers our investigation into RQ1 and RQ 2 in Chapter 1. We now have initial proposals for modifications to Scrum to accommodate safety assurance concerns, and a fuller description of how assurance case development can be integrated with Scrum. In these structured interviews, we are concerned to preliminary evaluation the credibility, feasibility and efficacy of these proposals with practitioners.

5.4 Participants and Interviews

We interviewed 6 participants, from the academic and industrial domains in order to use their experience and insight to gain feedback on our proposed approach. All of the participants have been involved with Safety Critical-Systems, Agile methods, or both. We approached the participants because of their skills, experiences, and the extent of their field link to our research. The questions we asked aimed to explore a wide range of concerns, challenges and feedback on the integration of 4+1 principles into Scrum, by giving the practitioner's the liberty to provide detailed responses. Our participants were based in the UK, Sweden, the USA, and Norway. The limited semi-structured interviews sample meant that not enough data was collected from only 6 practitioners, and more practitioners are needed. The table below shows participant qualifications:

PARTICIPANTS AND JOB TITLES

DATE	INTERVIEWEE JOB TITLES
16/12/2015	Agile Coach and Certified Scrum Trainer
24/05/2016	Kernel Developer
15/03/2016	President and Managing Partner Development Practice

12/04/2016	Software Engineering Professor
25/04/2016	Chief Evangelist
14/05/2016	Senior Implementation Project Manager

Interviewees were briefed with an introduction that explained the overall aims of the work, i.e. to explore perceptions around the 4+1 Principles of Software Safety Assurance and their implications for Scrum. Then the 4+1 principles were explained, along with an outline of our proposal for integrating these principles within a Scrum development. After the introduction, we asked for feedback relating to the proposal – picking out specific features one-by-one (e.g. our recommendations for team composition). The questions were designed to address two concerns: a) whether the proposed approach challenges agility and b) whether the proposed approach challenges safety assurance. The majority of the interviews were conducted over phone, email or via Skype.

5.6 Interview Findings

This section illustrates the responses from the feedback sessions that have been conducted during this empirical study. The responses are organized as follows. Each participant's response is presented in a separate table. Each table is organized according to the proposed modifications of Scrum (i.e. as presented in the previous chapter). Responses are presented alongside the proposed modification.

OVERVIEW OF STUDY FINDINGS (Participant 1)

Recommendation	Response from participants
<p>A second Product Backlog – the Safety Backlog</p>	<p><i>OK – already in SafeScrum. As an aside, we would like to point out the difference between the Safety Backlog and the SSRS. In our opinion, the safety backlog is a realization / concretization of the (SSRS).</i></p>
<p>New role in Scrum: Safety Team Member for maintenance of the Safety Backlog and ensure that they fully understand the source and rationale behind each of the requirements in the Safety Backlog.</p>	<p><i>Interesting idea. This implies that we need a link from a functional requirement to the related safety requirements – i.e. safety requirements that are included because the function could otherwise lead to a hazard. This could be part of the hazard log but is currently not considered in SafeScrum.</i></p>
<p>Daily Scrum meetings can be used to provide an early indicator for monitoring progress towards safety requirements satisfaction</p>	<p><i>Why do we need early indicators? How can we identify a reasonable set of indicators and how can we use them? In our opinion, this is a lot of work with a doubtful effect. It might, however, be worthwhile if the proposed indicators could be collected and analysed automatically.</i></p>

<p>The Safety Backlog should also be populated through performing (inductive, ‘bottom up’) hazard analysis (e.g. functional failure analysis) on the core product backlog requirements.</p>	<p><i>This is a standard approach based on e.g. user stories and part of the Hazard Log.</i></p>
<p>What do they think about the idea of documenting an assurance case ‘as they go’ during the Sprints?</p>	<p><i>Decomposition of safety requirements is performed in CIA (Change impact analysis)</i></p>
<p>Sprint review should attempt to check whether the implementation fully addresses the intent of the original safety requirement, and review the verification evidence presented. Further verification will need to be allocated as an activity to the following Sprint.</p>	<p><i>This goes for all requirements and should be considered together with ‘normal’ requirements. This will involve the liaison between SafeScrum and RAMS (reliability, availability, maintainability and safety),</i></p>
<p>Hazard analysis should be performed as part of Sprint review.</p>	<p><i>Verification of safety requirements (RAMS) are outside the current version of SafeScrum. This might change for a future version, especially since the SafeScrum team has more system knowledge. At the present, only V&V of functional requirements are done inside the sprint team. We should move SafeScrum in a direction where part of the safety V&V is moved inside the SafeScrum team – if necessary together with the SafeScrum RAMS liaison</i></p>
<p>Daily Scrum meetings can be used to provide an early opportunity to identify unintended side-effects emerging from a chosen implementation approach.</p>	<p><i>Is this necessary? Seems like a lot of work with doubtful benefits. In our opinion, this is best done through the daily stand-ups and by having a safety</i></p>

<p>The criticality of chosen requirements for the sprint (or requirements related to those chosen) should be highlighted at the beginning of the sprint. In particular, it should be highlighted where independence cannot be demonstrated between the implementation of requirements of differing criticality; the higher criticality should be used.</p> <p>Where new hazardous behaviours (i.e. new software level contributions to system-level hazards) are identified, new safety requirements (to manage these behaviours) must be added to the safety backlog</p>	<p><i>culture in the SafeScrum team.</i></p> <p><i>Even though it is possible to assign criticality to individual requirements or functions, In my honest opinion it is more reasonable to assign a criticality to the system and use this criticality throughout the project. Operating with different SIL-values for different functions is not reasonable.</i></p> <p><i>OK. Done in CIA</i></p>
---	---

OVERVIEW OF STUDY FINDINGS (Participant 2)

Recommendation	Response from participants
<p>The Safety Backlog should record traceability between safety requirements and system hazards</p>	<p><i>I'm not sure if I entirely agree. I think the user stories need to include safety steps, issues, and concepts. As the team develops the software for the user stories, functional, non-functional, and safety requirements ought to be developed together. I'm not sure this will be easy to coordinate with two distinct Backlogs</i></p>

<p>A Second Product Backlog – the Safety Backlog</p>	<p><i>To me a "Safety Backlog" could easily be a database view on the principal Backlog (much like a team-specific backlog on a multi-team project can be a database view on the one master backlog -- show me only those stories from the master Backlog that team one will work on)... BTW, I am guessing that safety is in part a non-functional requirement that affects most other stories, which would lessen the need for a separate Backlog.</i></p>
<p>(Inductive, ‘bottom up’) Hazard analysis should be performed as part of Sprint review in order to identify hazardous side effects from the design and implementation activity undertaken during the sprint.</p>	<p><i>Strictly speaking, this is reliability analysis since the impact cannot be assessed without understand how the other engineering disciplines contribute to safety in the scenario and context. But the idea of doing on-going hazard and safety analysis is a good one. I think that an explicit role (Harmony process calls it "Safety Czar") allows for this independence of perspective.</i></p>
<p>Justification must be maintained as to how the new requirements and design addresses the original safety requirement</p>	<p><i>What about how the SW safety requirements will be verified?</i></p>
<p>Independent stakeholders are required as part of this review. It should involve the nominated safety team member and potentially some or all of the following: the system team (as ‘customer’ to the software</p>	<p><i>Independence is particularly important for the developers of the verification procedures (testing, formal methods, etc). Surely they should participate in the review as well?</i></p>

<p>team), the system customer (e.g. system operator), a suitable internal safety reviewer (as proxy for the regulator's position).</p> <p>The traceability of safety requirements to the verification evidence, and the justification as to the adequacy of the evidence set should be documented in the assurance case by the nominated safety team member. This should take place in the Sprint immediately following the review and will be presented during the following Sprint review</p> <p><u>Other Comment Generated</u></p> <p>4+1 Fundamental Principles of Software Safety.</p> <p>It is necessary to ensure that assurance effort has been targeted at attempting to reveal modelling errors, coding errors, and tool-use error</p>	<p><i>It is very unusual for a Scrum team to have a separate V&V team but it is required by a number of safety standards. Probably should be called out here</i></p> <p><i>I know it may not be part of the 4+1 framework but I'd also add "Software safety shall be ensured throughout all phases and activities of the software development process". Too many people think it's sometime addressed in only a single step or phase.</i></p> <p><i>Good to have this here.</i></p>
---	---

OVERVIEW OF STUDY FINDINGS (Participant 3)

Recommendation	Response from participants
<p>Does Agile have a culture of documentation? Especially if we developing safety critical system projects?</p>	<p><i>That is the great mess: that there is no documentation, that we encounter less and less about documentation, that is misunderstood what agile is. I would expect that skill sets of the people would be appropriate to deal with safety related concerns. For example one of my clients for medical devices, needs to follow some kind of regulatory documentation; sub-creating that documentation is the another output that the team create – just as we have the team support software so we have the team support documentation</i></p>
<p>A second product backlog – the Safety backlog</p>	<p><i>My first impression is that we do not want the Safety Backlog, In Scrum the Product Backlog is the stream of the work the team produces. I think safety related issues can be represented inside the normal Backlog: The product backlog is used to log user stories. These stories may involve functionality that is safety related to safety critical. Safety experts can review the stories with the stakeholders. The other well-known, well- used thing in Scrum is the use of accepting criteria to clarify the definition of “done”. The accepting criteria define what it means to be “done” with a particular user story. [Safety] considerations probably</i></p>

<p>Maintenance of the Safety Backlog is the responsibility of a nominated safety team member from within the software team. This team member must ensure that they fully understand the source and rationale behind each of the requirements</p> <p>Daily scrum meetings can be used to provide an early indicator for monitoring progress towards safety requirements satisfaction</p>	<p><i>represent acceptance criteria.</i></p> <p><i>I agree with the idea of having Safety team member and one or more people safety related experts. I think that it is vital that if we doing safety critical systems ether the product owner is an experts in the field of safety</i></p> <p><i>The daily scrum is where the team essentially creates the plan for that day. We do that because new things are typically coming up every day. It is really appropriate for the safety experts to be present and to bring out their concerns. The biggest thing is to identify which members of the development team are working particularly with sensitive requirement: may be the safety experts need to spend some time after the daily scrum meeting to ensure that [these requirements are] understood. There is one more very common meeting in Scrum - it is an official activity - it is called the Backlog Refinement Meeting. This lasts for one hour, the purpose is for the stakeholder and the product owner to work with the development team to refine Backlog items.</i></p> <p><i>Understanding the upcoming stores not the one we are doing in the</i></p>
---	--

<p>Is a good idea to have the Sprint longer than 2 weeks</p>	<p><i>sprint new, we make sure that we identified and redefine and we documented the important acceptance criteria or those story, that is the incredible time for safety experts to be present and see that safety issues does get captured</i></p> <p><i>The reason is, the shorter the Sprint the shorter the feedback (the Sprint review). The stakeholder community We share with all of the stakeholder community what the team has produced, we show them the new results, and the stakeholders understand it, and we see their reactions. We force contracting in terms of whether we are building the right product, specifically for safety critical systems</i></p>
<p>Could the retrospective activity in Scrum help the safety issues?</p>	<p><i>That is an interesting question, the retrospective activity is really the time for the Scrum team to be a better team; that meeting is not really about the product at all. The safety related staff [are concerned with] all aspects about the product. May be there are safety related things that might come up in the retrospective in terms of how the team is working. What I mean by tha ist if the team have ideas for</i></p>

<p><i><u>Other Comment Generated</u></i></p> <p>What about having Scrum and XP at the same time? Could that help safety critical projects?</p> <p>Is it a good or bad idea to have two scrum teams, the first working with the Safety Backlog and the second with the normal Backlog?</p>	<p><i>work process change - like how they can do the work - that might improve the chances of discovering safety-related issues</i></p> <p><i>I totally agree. My experience is that very frequently Scrum teams are highly functional teams - eventually they almost adopt XP practices. For example, pair programming is a better way to create the code - with two people you are much more likely to find errors early. These may be safety related or not.</i></p> <p><i>Bad idea, because every one who building safety critical system need to be aware of safety issues, and need be examining safety related issues, my be the another team do not think much regard for safety, and the anther safety team need to modify that to make it safe, it is much better to work on having aware of safety issues and what are the safety critical behavior need to be maintained.</i></p> <p><i>I think that Scrum is well suited for</i></p>
--	---

<p>What do you think about adapting Agile to safety critical systems, especially Scrum?</p>	<p><i>software. It required complex work and particularly requires a number of experts that have different specialties to all be coordinated in order to get the final desire and products</i></p>
---	--

OVERVIEW OF STUDY FINDINGS (Participant 4)

Recommendation	Response from participants
<p>As part of the Sprint review, those responsible for implementation of a safety requirement must present the solution and explain how the intent of the safety requirement is fully satisfied. The Sprint review should attempt to check whether the implementation fully addresses the intent of the original safety requirement, and review the verification evidence presented.</p> <p>When product requirements are chosen for a given Sprint, the related safety requirements from the safety product backlog should be identified, discussed and explained (by the nominated safety team member) at the beginning of the Sprint.</p>	<p><i>Does this imply that the whole of a safety requirement will be met in one Sprint? What about cross- Sprint safety requirements?</i></p> <p><i>Is it permitted to break a safety requirement – satisfied in one Sprint? He highlighted some of the challenges that we have not addressed.</i></p>

<p>The Safety Backlog should record traceability between safety requirements and system hazards</p> <p>It is necessary to ensure that assurance effort has been targeted at attempting to reveal modelling errors, coding errors, and tool-use errors.</p> <p>To implement this principle it is necessary to be able to identify the criticality of any software safety requirement</p> <p><u>Other Comment Generated</u></p>	<p><i>Generally we trace requirements not to hazards, but to the risk. The mitigation of each risk becomes a safety requirement</i></p> <p><i>Do you include the introduction of race conditions in this? We find that one of the most difficult things to catch.</i></p> <p><i>Is this possible in isolation? Can you address the criticality of any one safety requirement?</i></p> <p><i>Have you considered Scrum as a good way to produce a prototype (as recommended in 26262, 61508)?</i></p> <p><i>I wondered if it is worth mentioning an existence proof: QNX'S (QNX help build products that enhance their brand characteristics – innovative, high- quality, dependable) kernel has been certified to 61508 SIL3 and 26262 ASIL-D it was developed by Scrum.</i></p>
--	--

OVERVIEW OF STUDY FINDINGS (Participant 5)

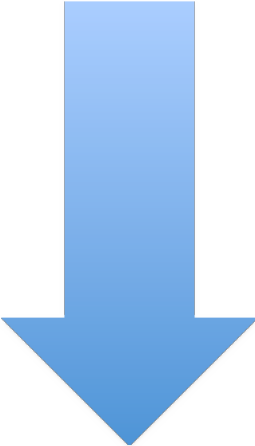
Recommendation	Response from participants
<p>A second Product Backlog – the Safety Backlog</p>	<p><i>Safety Backlog -- I coach teams to have only one Backlog for the product, though it often divides into sections to feed multiple teams. Still I agree with your idea that some Backlog items need to address safety. Also agree that there should be a bottom-up analysis of hazards. Also agrees that we should have haz analysis in the cycle</i></p>
<p>What is your opinion regarding principle two?</p>	<p><i>Intent of Safety Requirements Maintained - Keeping the epics and stories compact, and linked, goes a long way toward this. Traceability via tools has a place in this too. I would add that having a Sprint focused on stories that all focus on one major feature is a big help in 2 ways - less likely to overlook a risk or mitigation test, and productivity is higher because focus is concentrated in related areas of the codebase. In my team, we'd sometimes tell the Product Owner that we could lower the points estimate on a group of stories if they could all be done in the same Sprint. Supporting principle two and saying that there is practice within agile to keep backlog items - e.g. user stories linked.</i></p> <p><i>Practitioner also highlighted the fact that it's ideal to tackle 'linked' items in one</i></p>

<p>Practitioner wants to highlight on some issues</p>	<p><i>Sprint. This might relate to us in terms of trying to make sure that requirements and their corresponding safety requirements are tackled in the same Sprint.</i></p>
<p>Practitioner agrees but makes a recommendation</p>	<p><i>Safety Requirements shall be Satisfied -- One of the weakest traditional practices in my opinion has always been demonstrating safety through analysis. The rationale was that certain situations were too difficult to test explicitly, so a walkthrough of the source code is allowed. I can't recall having to resort to that even once in the 3-year Agile project my team did back when Agile was first emerging. We could test our embedded application on the target hardware or on a desktop PC on top of Windows. We could also break out any task and run it solo on the target hardware or on Windows. And we could inject error situations too.</i></p> <p><i>You have this: "The traceability of safety requirements to the verification evidence, and the justification as to the adequacy of the evidence set should be documented in the assurance case by the nominated safety team member. This should take place in the Sprint immediately following the review and will be presented as during the following Sprint review)." I'd aim to do all this in the Sprint that implements the story, to avoid the errors and waste associated with context switching for team members</i></p>

<p>Practitioner agreeing but making a recommendation</p>	<p><i>You have this: "The traceability of safety requirements to the verification evidence, and the justification as to the adequacy of the evidence set should be documented in the assurance case by the nominated safety team member. This should take place in the sprint immediately following the review and will be presented as during the following sprint review)." I'd aim to do all this in the sprint that implements the story, to avoid the errors and waste associated with context switching for team members.</i></p>
<p>What do you think about the identification and mitigation of Hazardous behaviour?</p>	<p><i>Hazardous behaviour of the software has been identified and mitigated - Yes! Very much needed. I'd do the hazard analysis during the early sprints rather than at a sprint review. I coach teams to use one or more of the early iterations to deliver knowledge rather than product increments (or a combination of both), especially for projects where there is much technical uncertainty. Also, as part of loading the Backlog, I'd use hazard analysis periodically to load new safety stories into the Backlog. You have a bullet point saying essentially the same.</i></p>
<p>Practitioner raises an interesting question</p>	<p><i>The confidence established in addressing the software safety principles shall be commensurate to the contribution of the software to system risk. -- Agree with all you've said here except that I wouldn't move the work to a later Sprint (as already discussed). Better to split the story to fit the</i></p>

<p><u><i>Other Comment Generated</i></u></p>	<p><i>Sprint length, and have it disabled in the partial releases till enough of it is present to really release.</i></p> <p><i>General comment - I think the role you call out as safety team member can operate alongside the rest of the Agile team in the same Sprints. I believe that's in line with what you're saying</i></p>
--	--

OVERVIEW OF STUDY FINDINGS (Participant 6)

Recommendation	Response from participants
<p>Practitioner supportive of our proposal. “From the Recommendation Script“</p> 	<p><i>You capture just about all of the safety practices I advocate:</i></p> <ul style="list-style-type: none"> <i>- Assessing potential hazards "top down" early in design (this is where Fault Tree Analysis is helpful).</i> <i>- Including specific "safety stories" in the Backlog.</i> <i>- Establishing and maintaining traceability of the "safety stories" to the hazards which were identified</i> <i>- Making sure that safety features don't get lost or removed (this is why the features need to be clearly</i>

	<p><i>commented as such)</i></p> <ul style="list-style-type: none">- <i>Directly checking that safety stories have been implemented effectively, that is, that the mitigations work</i>- <i>Regularly reviewing status of safety issues to look for (a) new hazards arising from design, and (b) hazards as a result of errors</i>- <i>Conducting "bottom up" safety analysis as design emerges (FMEA, FMECA can be useful here)</i>- <i>Involving independent reviewers with specific product knowledge to look at safety controls</i>- <i>Tempering the safety efforts by the overall product risk level.</i> <p><i>The one point Osama makes that I'm not sure I agree with is that maintaining the safety items should be made the responsibility of a single team member. This is similar to the classic problem with quality</i></p> <ul style="list-style-type: none">- <i>"QA is not our job." My feeling is that in industries where hazards to life and limb are present (e.g. aviation, rail transportation, nuclear power, medical devices), awareness of and attention to safety need to be everyone's job.</i>
--	--

<p><u>Other Comment Generated</u></p>	<p><i>It might be helpful to you to review the old AAMI TIR 32 – (the medical device software risk guidance) which was superseded by IEC TIR 80002-1.</i></p> <p><i>TIR 32 talks about many of the issues you discuss, and in addition describes such concepts of "first point of software control" and "last point of software control." Since TIR 32 is now obsolete, I don't see any problem with sharing that.</i></p>
--	--

5.7 Identification of emerging themes

In this section, we highlight some of the important findings from the 6 semi-structured interviews summarized in the preceding section. There were also some answers that were not included on the interview scripts, so we also report here on some of the potentially interesting issues raised by the participants.

5.7.1 4+1 principles within agile and mapping agile to standards

This section addresses differences in the practitioners' opinions regarding the interaction of the 4+1 principles into agile and the integration of agile methods within one of the standards, and also addresses queries raised in the interviews.

We should discuss one of the practitioner's answers in more detail. His responses were often formulated with respect to the specific obligations of standards (i.e. his benchmark was whether the practice is *compliant*, in this case with IEC 61508).

We quote some of our recommendations and the practitioners' answers (R means 'Recommendation'; the practitioners' responses are quoted in italics):

R. The Safety Backlog should record traceability between safety requirements and 'core' product Backlog requirements (e.g. between a safety constraint and the function to relates).

“OK. This info is found in the Hazard Log which is a part of IEC 61508”

R. The traceability of safety requirements to the verification evidence, and the justification as to the adequacy of the evidence set should be documented in the assurance case by the nominated safety team member. This should take place in the sprint immediately following the review and will be presented as during the following Sprint review).

“OK. Traceability is already required in IEC 61508. The adequacy of evidence is not a part of IEC 61508. In addition, we see no good measure for adequacy. This will be a problem if we try to apply such a measure.”

R. If it is determined that more time is required to assess the adequacy of the proposed solution (than the Sprint Review allows) then a more thorough review will need to be allocated as an activity to the following Sprint. Whilst this activity is taking place, ideally further implementation of the solution should be avoided (i.e. other requirements from the backlogs should be prioritized)

“The V&V of safety requirements are taken care of in the RAMS [Reliability, Availability, Maintainability, and Safety] process[es] – see diagram on separation of concerns – and is outside current SafeScrum. ... implementations that fail here are returned to phases 1 – 4 in IEC 61508 and will go through CIA [Change Impact Analysis] phase 1”¹

R. At the planning stage of a Scrum development the chosen modelling approach, implementation language, development environment and assessment tools should be documented. Hazard analysis should be conducted to on the chosen languages, processes and tools to identify the potential for the introduction of implementation errors. Where the potential for error is identified, it is necessary to provide evidence for non-introduction of error (e.g. tool qualification), controls for the minimization of error introduction (e.g. process guides) or means of demonstrating the absence of introduced errors (e.g. static analysis tools). Justification of the chosen strategies must be documented in the assurance case.

“OK. This is already done for all projects ran according to IEC 61508. The process is defined via the SIL value. We have two questions related to this issue. What do you mean by

a. “Conducted to on the chosen language”?

b. “Potential for errors” and how can this be identified?”

Another participant made a good point about the need to think beyond the framework of standards. In this respect, working to the 4+1 principles – which capture the general, rather than the specific intent of standards – could be more valuable: *“More generic 4+1 principles as you do in this thesis, rather than restrict the work to one of the many standards ”*

The main criticism that has been made about SafeScrum so far is that the approach focuses on compliance with IEC 61508; people do not step back from individual standard and concentrate on the spirit of the standards (and of safety) in general.

“More generic 4+1 principles as you do in this thesis, rather than restrict the work to one of the many standards ”

¹ Here the respondent is referring to elements of the SafeScrum process

R. When product requirements are chosen for a given sprint, the related safety requirements from the safety product backlog should be identified, discussed and explained (by the nominated safety team member) at the beginning of the Sprint

Interesting idea. This implies that we need a link from a functional requirement to the related safety requirements – i.e. safety requirements that are included because the function could otherwise lead to a hazard. This could be part of the hazard log but is currently not considered in SafeScrum.

In the literature review, we identified several challenges to the integration of Agile Methods to standards-based safety engineering (see, particularly, section 2.8 above). However, the empirical study carried out for this thesis aimed to reduce confusion in this area, by collecting more information about the integration of agile methods into the field of safety critical systems and by suggesting an approach to this integration which goes beyond “the letter” of a given standard. Our experience in the study provided some reassuring answers and motivation to continue to find better research results.

5.7.2 Agile and Documentation

Another practitioner provided the following comment on the lack of understanding as to how Agile methodologies deal with documentation:

“Like the team support software, the same the team supports documentation”

This practitioner commented that one of the biggest misconceptions of agile methods is that they are not willing to support documentation in the process. However, for example, in XP documentation is recognised as part of the development team’s responsibility [70].

5.7.3 Safety Backlog

There are major differences of opinion among our respondents concerning our proposal to introduce a second Product Backlog, the Safety Backlog, to track safety-critical concerns through the Scrum-based process. As will be seen from the extracts quoted below, the practitioners’ most commonly-held opinion was that the Safety Backlog was non-essential. However, some of the literature promotes the importance of a Security Backlog [21], to help to deal with the security “safety” issues in Scrum.

The following extracts from our interviews indicate the practitioners’ differing views:

“Until I see really compelling evidence that there should be a separate backlog there is only ONE backlog that includes all of the work that needs to be completed. If safety is so critical to the effort then the development team should have on it people with deep safety expertise. The safety people might collectively form a safety community of practices but the individuals can and likely should be fully contributing members of Scrum development teams.”

“Safety Backlog -- I coach teams to have only one backlog for the product, though it often divides into sections to feed multiple teams. Still I agree with your idea that some backlog items need to address safety. Also agree that there should be a bottom-up analysis of hazards.”

“A safety requirement is a requirement derived from the initial system level safety analysis, but to the development team they are just a requirement the same as any other requirement that may have been developed from the system level. In some Agile approaches there is a separate ‘safety’ backlog (see Thor Myklebust [45]), in other cases within the overall product backlog those requirements that are safety related are just ‘tagged’ to indicate they are safety related”

The general view that we discern is that addition of the Safety Backlog may be a good idea, but that we need to conduct more investigation to see if that is true.

5.7.4 Safety Team Member

Almost all of the interviewees felt that our proposed addition of a team member with specific responsibility for safety issues was a sensible idea. We present extracts from the interviews below:

“I agree with the idea of having a Safety team member and one or more safety related experts. I think that it is vital that if we doing safety critical systems the Product Owner is an expert in the field of safety.”

I think the role you call out as safety team member can operate alongside the rest of the Agile team in the same Sprints. I believe that's in line with what you're saying.

The Safety Team Member is our proposal, to improve communication between the development team and the independent assessor, to ensure that the safety requirements, safety criticality, and safety case will meet the customer requirements, and to ensure that the development team has fully understood the safety requirements.

5.7.5 Hybrid agile approach and relationship to safety

R. Hybrid approaches that combine two Agile practices within the area of safety critical systems should be developed, in order to achieve better safety results. There is some evidence from literature and the practitioners to support this recommendation, for example the Certified Software Development Process based on XP@Scrum for ISO9001:2000 [72] .

I totally agree, in my experience is that very frequently scrum team are highly functional team, eventually adopt almost the XP practices, for example pair programming is the better way to create the code with 2 people you much likely to find errors early, my be safety related or not

Some features of XP practices are likely to benefit the treatment of safety issues. For example Paige et al [20] suggest the use of Pair Programming which is likely to help in catching errors, identifying problematic code (that may need refactoring vs rewriting) and providing instant feedback on ideas.

The challenges and observations identified in our semi-structured interviews, suggest that there may be considerable advantages in adopting a hybrid Scrum-XP approach for safety-related projects, in order to take the benefits of both Scrum and XP approaches.

5.7.6 Sprint Duration for Safety

Agile practitioners generally recommend a Sprint length of one to two weeks. However, some of our interviewees argue that one to two weeks are not enough to satisfy safety requirements, and suggest that it is better to extend Sprints for safety-critical projects.

Here we list some of the different opinions about the Sprint duration from the practitioners we interviewed:

“The shorter the Sprint, the shorter the feedback”

“We're generally concerned by the time it takes to assess safety issues as part of sprint planning. If the "Stakeholder consortium" is large (i.e. more than about 2 people), then how often does it meet and make decisions? (The Scrum approach seems to assume a single "product owner" who can make decisions _really_ fast - will this work if all safety-related planning and decisions have to be taken by some big committee??)”

“In Agile the timescale takes precedence so if functionality cannot be completed in time it is removed from the current backlog. Consequently until an increment has been completed you cannot be certain what functionality will actually have been completed.”

5.7.7 Queries and Recommendations

In our initial interviews, we captured practitioners recommendations and queries concerning the integration of safety assurance into Agile methodologies, and also elicited their feedback concerning the challenges posed by this approach. These challenges and recommendations will be checked and examined in the next phase of our work, and appropriate ones will be implemented in future.

Below, we have listed some of the participants’ queries and recommendations:

Under "Principle 1", you talk of the Safety Backlog being "initially populated" by both "top down" and "bottom up" analysis. Totally agree, of course, but that seems to infer that you already have enough design and architecture to give you something to "iterate over" ... so you need to have done

enough architecture and design to do the safety analysis. This seems contradictory to Agile's "minimal design" mantra, but seems (at least to us) to be the crux issue - how much "up front design" is "just right" for a particular system?

In P1 - you say that "when a product requirement is chosen for a given Sprint, the related safety requirements from the safety product backlog should be identified, discussed, and explained." Why not go further? Why not require that the safety requirements are implemented in the _same_ Sprint as the "product requirement". Put another way - would you ever want to implement a "feature" in a particular Sprint, but NOT implement its related safety requirements in the same Sprint?

In Principle 4 - you talk about "justification of chosen strategies" being documented. I would go further and require justification of the _rejected_ strategies too. This is important in very long- lived projects, where you want future maintainers (years in the future) to know what you rejected and why...

You don't talk about Refactoring. This is important - the Agile people say a refactoring is "Done" when "all the tests pass..." Big deal! In the context of maintaining a safety argument, how would you define "when a refactoring is done"?

We have discussed and summarized the practitioner's opinions regarding our recommended approach and our initial analysis of the challenges it presents. These findings will be taken into consideration for any future work.

5.8 Summary and Issues Arising

In this chapter, we have reported on the results of 6 semi-structured interviews which were conducted in order to gain actual practitioners' reactions to our approach to integrating the 4+1 safety assurance principles within Scrum and our initial assessment of the challenges associated with the approach. Although the research sample is limited, the study has benefitted from the introduction of a more pragmatic perspective, which complements our rather theoretical viewpoint.

We summarise the results in forms of answers to the two research questions we posed at the beginning of the study:

RQ1 What are the current concerns and opportunities voiced by safety-critical systems professionals regarding the use of agile development methods for safety-critical systems development?

We encountered some difficulties during the interviews. For example, one aspect that we briefly touched on was misunderstanding and lack of knowledge or awareness from both the Agile and the safety practitioners concerning each others' outlooks and work. Nonetheless, the semi-structured interviews have motivated us to propose further work involving interviews on a much larger scale in order to achieve better results. We need to move from the basic questions that we asked during the current research – i.e. “is it feasible to integrate safety into Agile methods?” – towards more specific questions, such as the following:

- Is it permitted to break a safety requirement which has been satisfied in one Sprint?
- Safety backlog - Yes or No?
- Safety team member - Yes or No?
- Hybrid agile approach - Yes or No?

- Sprint duration: 1 to 2 weeks? Or longer?
- Independence is particularly important for the developers of the verification procedures (testing, formal methods, etc). Surely they should participate in the review as well?
- What about how the software safety requirements will be verified?
- Would it be desirable/possible to implement a "feature" in a particular Sprint, but NOT to implement its related safety requirements in the same Sprint?
- Is Scrum a good way to produce a prototype (as recommended in 26262, 61508)?

RQ2 What changes are necessary to the Scrum Process in order to address the 4+1 Software Safety Assurance Principles?

The findings indicate clear support for the recommendations that we propose to integrate the 4+1 safety assurance principles into the Scrum process in order to help demonstrate compliance with safety standards, and with our initial survey of the challenges presented by such an approach. Our recommendations stem from the use of the 4+1 principles to build on the strengths of the Scrum process to improve management of safety issues in system development. Finally the numbers of practitioners were not adequate for a good credibility, but more is needed.

Chapter 6

Conclusion and Future Work

6.1 Introduction

- This chapter summarizes all of the previous chapters, from the literature review through to the evaluation. In the literature review, we examined previous work in the area of safety assurance and Agile methods, to identify potential challenges and barriers for the integration of safety into Agile methodologies. This was followed by empirical research to collect a snapshot of current practitioner opinions. Our initial survey gave us a number of vital points that influenced the second stage of our research, the semi-structured interviews. For example, we retrieved the following substantial points from our initial survey:
- The need to provide assurance of product safety, as well as making the product safe was strongly recognized (>95%). This indicates the importance of establishing the safety case alongside development.
- There was recognition (50%) that safety problems aren't always identified early in the lifecycle. There was strong support (90%) for regular evaluation of safety case progress during development. Again, this suggests safety case development should be treated as an 'in-increment' activity.
- Prioritisation of requirements (77%) and assurance effort (84%) (according to risk) was recognized as important. Together with the observations regarding evaluation, this suggests that incremental safety case development may be useful in helping guide an incremental software development process.

After that, we presented an preliminary analysis of challenges, which stemmed from the literature review and responses to the initial survey. We then made a series of recommendations as to how Scrum could be adapted to allow for the incorporation of safety assurance concerns. We subjected our analysis and recommendations to evaluation,

by means of semi-structured interviews with practitioners.

The results from the semi-structured interviews were encouraging, and opinion was broadly supportive of our approach. That is indicated by the following key points from the semi-structured interviews:

- One of the practitioners stated his support for Principal 2, saying that there is an established practice within Agile to keep Backlog items - e.g. user stories - linked. He also highlighted the point that it is considered best practice to tackle 'linked' items in one Sprint. This provides support for our attempt to ensure that requirements and their corresponding safety requirements are tackled in the same sprint.
- Support for the inclusion of hazard analysis in the Sprint.
- Support for the idea of a separate safety team member (the new role we proposed in this work).
- Inclusion of specific "safety stories" in the Backlog.
- Establishment and maintenance of traceability between the "safety stories" and the hazards which were identified.
- Support for the observation that it is necessary to ensure that assurance effort has been targeted at attempting to reveal modelling errors, coding errors, and tool-use errors.

6.2 Initial perceptions

Our research began with a literature review that explored the theoretical implications of our topic, from both the safety assurance and the Agile perspectives. Some gaps and needs in existing approaches became clear during the review. In particular, some felt that agile development insufficiently recognized the need to provide assurance of product safety, as well as making the product. Furthermore, we found that there are only a few surveys and empirical studies concerning the possibility of integrating Agile Methods into the field of safety critical systems engineering.

Moreover, we used the preliminary results from the literature to support our hypothesis. Therefore, we conducted our initial survey, followed thereafter with semi-structured interviews to strengthen the evidence required to support the hypothesis. In conclusion, the reaction to the proposal to integrate the 4+1 safety assurance principles into Scrum was generally positive, however a number of concerns emerged from the evaluation. These key concerns were discussed in chapter 5.

6.3 Initial Survey

The initial survey attempted to draw out specific responses relating to the (possible) incremental and iterative nature of safety requirements development, hazard analysis and safety (assurance) case development.

The survey successfully targeted practitioners with experience of safety critical systems development and agile development methods. The responses received from the survey indicated that the practitioners have a largely positive view on the integration of Agile and safety methods. By addressing specific practices in safety assurance, the survey responses also help inform the features of future integration attempts (e.g. concerning the placement of safety activities inside and outside of the increment ‘cycle’).

6.4 The development of the initial proposal for the integration of the 4+1 principles

Some researchers have attempted to tailor agile methods to comply with specific standards (e.g. SafeScrum and IEC61508). However, this risks over-configuring the agile method in such a way as to make difficult to apply to another safety standard. Our approach sought to look at the problems of addressing the more *fundamental* principles of safety assurance by adopting the emerging “4+1” safety principles [76] and investigating how a Scrum process challenges, and can be adapted to support, these principles. Our aim was to suggest the minimum of changes necessary to make the Scrum process support the assurance principles. By adopting a principle-based approach, as well as getting to the ‘heart’ of the problem of safety assurance, it also provided us with greater flexibility in configuring the development and assessment process.

6.5 Semi-structured interviews

We conducted semi-structured interviews with participants to gain feedback on our proposed approach. More specifically, the semi-structured interviews were designed to explore the general feasibility of the approach, and to provide an assessment as to whether the 4+1 principles can be addressed without compromising agility.

The results of our 1-to1 semi-structured interviews gave strong indication that the practitioners felt that there is a significant potential for successful integration of the 4+1 principles within Scrum. As discussed above, there were some issues where practitioners were concerned to focus only on one safety standard, and it was also the case that neither the Agile practitioners nor the safety practitioners had a clear understanding of the outlook and work of the other group. However, we used these issues to inform a further set of questions.

6.6 Limitations

Our study suffered from some limitations which should be addressed in future work:

- a) Large scale empirical evaluation is simply not possible in the timescale of a Masters programme;
- b) We had considerable difficulty finding practitioners who were sufficiently expert in both Agile and safety – in the end, we were able only to interview those with an interest in the integration of agile and safety;
- c) The limited research sample meant that not enough data was collected from the practitioners;
- d) We needed to establish specific criteria in order to avoid deviation from the interview script.

6.7 Future Work

In this section we will illustrate a number of potential work need to be achieved:

As outlined above, there has been very little research to date concerning the integration of safety assurance concerns in Agile methodologies. Within the confines of the Masters degree, we have been able to conduct an initial evaluative survey and to propose a potential approach. It must be noted, however, that this represents only the preliminary stage of a line of research in this area. In this section we will illustrate a number of potential ways in which the initial work presented in this thesis could be expanded and validated in future.

- It would be desirable for future researchers to conduct a pilot project. This should be formulated in such a way as to address the particular themes that emerge from our survey: for example, the pilot project could evaluate how difficult it was to establish safety requirements at the outset, and how much they change during the project.
- Our initial survey in this area highlighted some areas of interest in the role of the safety case. Further work is required to explore how GSN safety cases could be linked to a notion of safety Product backlog within Scrum. The research indicates that existing assurance case activities need to be adjusted.
- Software safety argument patterns provide a way of capturing good practice in software safety arguments. Future research could develop a pattern-based approach to integrating software safety cases, Scrum's Safety Product Backlog, risk-based planning, and requirements-based evaluation. Software safety argument patterns describe the nature of the argument and safety claims that would be expected for any software safety case.
- Peer review (through structured questionnaire) of our proposed approach as applied to a worked case study example should be conducted.
- It would be useful for future researchers to engage in a larger-scale interview-based evaluation of an approach for safety case development within Scrum. In particular, research should address the development of pragmatic techniques to ensure that evidence to validate the safety case is developed and collected in all incremental

(Sprint) processes.

- A realistic case study should be developed, to investigate where there are opportunities to build up a safety case as a part of an Agile development, to determine the risks and conflicts associated with this approach and how these risks could be mitigated.

Abbreviations

ACM	Association for Computing Machinery
ASD	Adaptive Software Development
CIA	Change impact analysis
CMMI	Capability Maturity Model Integration
CRD-RM	Certifiable, Agile, Reusable, and Disciplined Reference Model
DSSRs	Derived Software Safety Requirements
DSDM	Dynamic Systems Development Method
FDD	Feature Driven Development
FAA	Federal Aviation Administration
GSN	Goal Structuring Notation
IEEE	Institute of Electrical and Electronics Engineers
IEC	Functional Safety Standard
ISO	International Organisation for Standardisation
MIL-STD	Military Standard
NIST	National Institute of Standards and Technology
NASA	National Aeronautics and Space Administration
QUMAS	Quality Management and Compliance Solutions
RAMS	Reliability, Availability, Maintainability and Safety
RQ	Research Questions
SIL	Safety Integrity Level
SSRS	System Safety Requirements Specification
XP	Extreme Programming

References

- [1] Ericson, C.A. 'Concise encyclopedia of system safety' definition of terms and concepts (Wiley, 2011)
- [2] Paul, F.G., B.Bruce, D.Ben, A. 'Model-Driven Development for Safety-Critical Projects in Intelligent Energy', 28–30 October 2013
- [3] Leveson, N. 'Engineering a safer world : systems thinking applied to safety' (MIT Press, 2011.
- [4] Ericson, C.A. 'Hazard analysis techniques for system safety' (Wiley-Interscience, 2005.
- [5] 'NASA system safety handbook' (National Aeronautics and Space Administration, NASA headquarters, 2011, Version 1.0. edn. 2011)
- [6] Knight, J.C. 'Safety critical systems: challenges and directions', in Editor (Ed.)^(Eds.): 'Book Safety critical systems: challenges and directions' (2002, edn.), pp. 547-550
- [7] Leveson, N. 'SafeWare : system safety and computers' (Addison Wesley, 1995)
- [8] Hunt, J. 'Agile software construction' (Springer, 2006)
- [9] McCaffery, F., Pikkarainen, M., and Richardson, I. 'Ahaa --agile, hybrid assessment method for automotive, safety critical smes', in Editor (Ed.)^(Eds.): 'Book Ahaa --agile, hybrid assessment method for automotive, safety critical smes' (2008, edn.), pp. 551-560
- [10] Hazzan, O., and Dubinsky, Y. 'Agile software engineering' (2008)
- [11] Hazzan, O., & Dubinsky, Yael. ' Essays on agile projects and beyond (SpringerBriefs in computer science).'
- [12] Beck, K. (2004). 'Extreme Programming Explained'. Embrace Change (2nd Edition ed.). Boston, MA: Addison Wesley PublishingCo., Inc.
- [13] Š mite, D., Moe, N.B., and Ågerfalk, P.r.J. 'Agility across time space : implementing agile methods in global software projects' (Springer, 2010)
- [14] Schwaber, K. 'Agile project management with Scrum' (Microsoft Press 2004)
- [15] Rubin, K.S. 'Essential Scrum', a practical guide to the most popular agile process (Addison-Wesley, 2012)
- [16] Avizienis, A., J.-C. Laprie, and B. Randell. 'Fundamental Concepts of

- Dependability'. In Third Information Survivability Workshop (ISW-2000). 2000. Cambridge, Massachusetts, USA: IEEE Computer Society Press
- [17] Beck, K. (2000). Manifesto for Agile Software Development.
- [18] Stapelton, J. 'The Agile Software Development Series' Business Focused Development 2nd (second) Edition 2003.
- [19] Weihang Wu. 'Architectural Reasoning for Safety-Critical Applications' In Degree of Doctor of Philosophy, The University of York, September 2007
- [20] Richard F. Paige, Howard Chivers, John A. McDermid, and Zoë R. Stephenson. 'High-integrity extreme programming'. In Proceedings of the 2005 ACM symposium on Applied computing (SAC '05), Lorie M. Liebrock (Ed.). ACM, New York, NY, USA, 1518-1523.
- [21] Z. Azham, I. Ghani and N. Ithnin, 'Security backlog in Scrum security practices' Software Engineering (MySEC), 5th Malaysian Conference in, Johor Bahru, 2011, pp. 414-417.
- [22] R. Dardar, B. Gallina, A. Johnsen, K. Lundqvist and M. Nyberg, 'Industrial Experiences of Building a Safety Case in Compliance with ISO 26262' Software Reliability Engineering Workshops (ISSREW), 2012 IEEE 23rd International Symposium on, Dallas, TX, 2012, pp. 349-354.
- [23] Highsmith, J. (2000). 'Adaptive Software Development'. A Collaborative Approach to Managing Complex Systems. Dorset House.
- [24] Cockburn, A. (2002). 'Agile Software Development' AddisonWesley.
- [25] Palmer, S. and J. Felsing. 'A Practical Guide to Feature Development'. Prentice Hall 2002.
- [26] Bruce, D. 19 February 2013 (Agile analysis practices for safety-critical <http://www.ibm.com/developerworks/rational/library/agile-analysis-practices-safety-critical-development/>).
- [27] Fowler, M. (2005, December). The New Methodology
- [28] John, McDermid.: 'Software Hazard and Safety Analysis' in 7th International Symposium, FTRTFT 2002 Co-sponsored by IFIP WG 2.2 Oldenburg, Germany, September 9–12, 2002 Proceedings.
- [29] T P Kelly.: 'A Systematic Approach to Safety Case Management' in Proceedings of SAE 2004 World Congress, Detroit, March 2004 (Proceedings published by the Society for Automotive Engineers).
- [30] Chris Sims and Hillary Johnson "Scrum: a Breathtakingly Brief Agile

Introduction" – April 3, 2012

- [31] Kelly, T.P., 'Arguing Safety - A Systematic Approach to Safety Case Management' In PhD Thesis, in Department of Computer Science. 1999, University of York.
- [32] VersionOne: State of Agile Survey 2012 (7th Annual). 2013. Available: <http://www.versionone.com/pdf/7th-Annual-State-of-Agile-Development-Survey.pdf> [Accessed on 13th April 2013].
- [33] Richard, H. Tim, K. Journal of System Safety, Volume 46, No. 4, pp 25-33, System Safety Society Inc., July 2010.
- [34] John A McDermid. 'Software safety: where's the evidence? '. In Proceedings of the Sixth Australian workshop on Safety critical systems and software - Volume 3 (SCS '01), Peter Lindsay (Ed.), Vol. 3. Australian Computer Society, Inc., Darlinghurst, Australia, Australia, 1-6 2001.
- [35] John Hatcliff, Alan Wassying, Tim Kelly, Cyrille Comar, and Paul Jones 2014. Certifiably safe software-dependent systems: challenges and directions. In Proceedings of the on Future of Software Engineering (FOSE 2014). ACM, New York, NY, USA, 182-200.
- [36] European Committee for Electrical Standardization (CENELEC). Railway applications - Communication, signalling and processing systems - Software for railway control and protection systems. CENELEC Standard 50128, 2011.
- [37] Institute of Electrical and Electronics Engineers (IEEE). IEEE Standard Criteria for Digital Computers in Safety Systems of Nuclear Power Generating Stations . IEEE Standard 7-4.3.2, 2010
- [38] International Electrotechnical Commission. Functional Safety of Electrical/Electronic/Programmable Electronic Safety-related Systems. IEC Standard 61508 edition 2.0, 2010.
- [39] International Organization for Standardization. Road Vehicles – Safety. ISO Standard 26262, 2011. [60] RTCA. Software Considerations in Airborne Systems and Equipment Certification. RTCA Standard DO-178C, 2012.
- [40] Agile manifesto (2001). Available: <http://agilemanifesto.org>.
- [41] Ambler, S.: Quality in an Agile World. SQP References 34 SQP VOL. 7, NO. 4/©, ASQ (2005).
- [42] Hawkins, R., Habli, I., Kelly, T., McDermid, J. Assurance cases prescriptive

software safety certification: A comparative study. Volume 59, Pages 55–71
November (2013).

- [43] Bedoll, R. A Tail of Two Projects: How ‘Agile’ Methods Succeeded after ‘Traditional’ Methods Had Failed in a Critical System-Development Project. In: F. Maurer and D. Wells (Eds.): XP/Agile Universe, LNCS 2753, pp. 25–34. Springer-Verlag Berlin Heidelberg (2003).
- [44] Bowers, J., May, J., Melander, E., Baarman, M., Ayoob, A.: Tailoring XP for Large System Mission Critical Software Development. In: Wells, D., Williams, L., (Eds.): XP/Agile Universe, LNCS 2418, pp. 100–111. © Springer-Verlag Berlin Heidelberg (2002).
- [45] Stålhane, T., Myklebust, T., Hanssen, G. The application of Scrum IEC-61508 certifiable software, Unpublished, (2011).
- [46] Felix, R. Software Projects: Evolutionary v Big-bang Delivery (Wiley Series in Software Engineering Practice) Hardcover – 30 Jan (1997).
- [47] +SAFE, V1.2. A Safety Extension to CMMI-DEV. Defence Materiel Organization, Australian Department of Defence, Version 1.2 (CMMI-DEV, V1.2) SEI March (2007)
- [48] Kelly, T.; Weaver, R.: The Goal Structuring Notation – A Safety Argument Notation. In: Proceedings of the International Conference on Dependable Systems and Networks – Workshops on Assurance Cases, Florence, Italy, (2004)
- [49] Paige R., Charalambous R., Ge X., Brooke P.: Towards Agile Engineering of High- Integrity Systems. Proc. In: 27th International Conference on Computer Safety, Reliability and Security (SAFECOMP), September (2008)
- [50] B. Fitzgerald, K. J. Stol, R. O. Sullivan and D. O. Brien, "Scaling agile methods to regulated environments: An industry case study," 2013 35th International Conference on Software Engineering (ICSE), San Francisco, CA, 2013, pp. 863-872.
- [51] Ge, X., Paige, R. F., McDermid, J. A.: An Iterative Approach for Development of Safety-Critical Software and Safety Arguments. Agile Conference, Orlando, Florida, p. 35 – 43, (2010)
- [52] Z.R. Stephenson et al.: Health modelling for agility in safety-critical systems development, The 1st Institution of Engineering and Technology International Conference on Systems Safety, 2006
- [53] Matti Vuori.: Agile Development of safety-critical Software, Tampere University

of Technology. Report 14 Department of software systems Tampere 2011. ISBN 978-952-15-2595-7

- [54] Richard F. Paige, Andy Galloway, Ramon Charalambous, Xiaocheng Ge, and Phillip J. Brooke. 2011. High-integrity agile processes for the development of safety critical software. *Int. J. Crit. Comput.-Based Syst.* 2, 2 (July 2011), 181-216.
- [55] H. Jonsson, S. Larsson and S. Punnekkat, "Agile Practices in Regulated Railway Software Development," *Software Reliability Engineering Workshops (ISSREW)*, 2012 IEEE 23rd International Symposium on, Dallas, TX, 2012, pp. 355-360.
- [56] J. C. Marques, S. M. H. Yelisetty, A. M. Da Cunha and L. A. V. Dias, "CARD-RM: A Reference Model for Airborne Software," *Information Technology: New Generations (ITNG)*, 2013 Tenth International Conference on, Las Vegas, NV, 2013, pp. 273-279.
- [57] David J. Coe and Jeffrey H. Kulick.: "A Model-Based Agile Process for DO-178C Certification" Department of Electrical and Computer Engineering University of Alabama in Huntsville, Huntsville, Alabama, USA
- [58] L. b. Othmane, P. Angin, H. Weffers and B. Bhargava, "Extending the Agile Development Process to Develop Acceptably Secure Software," in *IEEE Transactions on Dependable and Secure Computing*, vol. 11, no. 6, pp. 497-509, Nov.-Dec. 2014.
- [59] L. b. Othmane, P. Angin and B. Bhargava, "Using Assurance Cases to Develop Iteratively Security Features Using Scrum," *Availability, Reliability and Security (ARES)*, 2014 Ninth International Conference on, Fribourg, 2014, pp. 490-497.
- [60] Konstantin Beznosov and Philippe Kruchten. 2004. Towards agile security assurance. In *Proceedings of the 2004 workshop on New security paradigms (NSPW '04)*. ACM, New York, NY, USA, 47-54.
- [61] A. Finnegan and F. McCaffery, "A Security Argument Pattern for Medical Device Assurance Cases," *Software Reliability Engineering Workshops (ISSREW)*, 2014 IEEE International Symposium on, Naples, 2014, pp. 220-225.
- [62] Bruce Douglass. "Agile Systems Engineering2015.
- [63] Bruce Douglass. "Real-Time Agility June 2009
- [64] Wells, D. (2009, September). *Extreme Programming: A Gentle Introduction*.
- [65] RTCA. *Software Considerations in Airborne Systems and Certification*. RTCA

- Standard DO-178C, 2012.
- [66] R.D. Hawkins, T.P. Kelly.: A Systematic Approach for Developing Software Safety Arguments. Department of Computer Science, The University of York, York, YO10 5DD, UK,
 - [67] J . Dennis Lawrence. “Software Safety Hazard Analysis”, Prepared for U.S. Nuclear Regulatory Commission, Manuscript date: October 1995.
 - [68] G. Melnik and F. Maurer, “Comparative analysis of job satisfaction in agile and non-agile software development teams,” in Extreme Program- ming and Agile Processes in Software Engineering. Springer, 2006, pp. 32–42.
 - [69] L. Gren, R. Torkar and R. Feldt, "Work Motivational Challenges Regarding the Interface between Agile Teams and a Non-Agile Surrounding Organization: A Case Study," Agile Conference (AGILE), 2014, Kissimmee, FL, 2014, pp. 11-15.
 - [70] Robinson, H., and Sharp, H. (2004) The characteristics of XP teams, in Proceedings of XP2004 Germany, June, pp 139 -147
 - [71] Nicoll, Dav, 23-24 September 2008, “Use of Agile Techniques in the Development of a Safety- Critical Rail Application”, Agile Business Conference, London.
 - [72] C. Vriens, "Certifying for CMM Level 2 and ISO9001 with XP@Scrum," Agile Development Conference, 2003. ADC 2003. Proceedings of the, 2003, pp. 120-124. doi: 10.1109/ADC.2003.1231461
 - [73] Mike Cohn . “Scrum Overview for Agile Software Development” <https://www.mountaingoatsoftware.com/agile/scrum/overview>
 - [74] O. Doss and T. P. Kelly. 2016. Challenges and Opportunities in Agile Development in Safety Critical Systems: A Survey. SIGSOFT Softw. Eng. Notes 41, 2 (May 2016), 30-31.
 - [75] Z. Guo, C.Hirschmann “An Integrated Process for Developing Safety-critical Systems using Agile Development Methods” ICSEA 2012 The Seventh International Conference on Software Engineering Advances
 - [76] Kelly, Tim. "Software Certification: Where is Confidence Won and Lost?." Addressing Systems Safety Challenges, T. Anderson, C. Dale (Eds), Safety Critical Systems Club (2014)
 - [77] U.K. Ministry of Defence, “JSP 430 - Ship Safety Management System Handbook,” Ministry of Defence January 1996.

- [78] Department of Defense: Standard Practice – System Safety (MIL-STD-882E). 2012.
- [79] Department of Defense.MIL-STD-882D: standard practice for system safety. U.S. Department of Defense January 2000.
- [80] NASA. NPR 8715.3C, NASA General Safety Program Requirements, Washington, DC. 2008.
- [81] NIST 1993. Review of Software Hazard Analyses. National Institutes of Standards and Technology. Draft (June 4).
- [82] IEEE: Standard for Software Safety Plans, IEEE STD 1228-1994. 17 logical p. of 23 physical pages.
- [83] Federal Aviation Administration, Job Aid-Conducting Software Reviews Prior to Certification, FAA, 2004.
- [84] Osama Doss, Tim Kelly "Addressing the 4+1 Software Safety Assurance Principles within Scrum" In: Second International Workshop on Agile Development of Safety-critical Software Workshop, XP 2016, Edinburgh, UK, 2016
- [85] T. Stålhane ,T. Myklebust.” The Agile Safety Case” SAFECOMP 2016 Workshops, LNCS 9923, pp. 5–16, 2016. DOI: 10.1007/978-3-319 1_1