Universität Stuttgart

System-Theoretic Safety Analysis in Agile Software Development

Von der Fakultät für Informatik, Elektrotechnik und Informationstechnik der Universität Stuttgart zur Erlangung der Würde eines Doktors der Naturwissenschaften (Dr. rer. nat.) genehmigte Abhandlung

Vorgelegt von

Yang Wang

aus Shanghai, China

Hauptberichter: Prof. Dr. Stefan Wagner

Mitberichter: Prof. Dr. Casper Lassenius

Tag der mündlichen Prüfung: 2018-10-17

Institut für Softwaretechnologie

2018

Acknowledgement

First and foremost, I would like to express my sincerest gratitude to my advisor *Prof. Dr. Stefan Wagner*.

He gives me the opportunity to work with him and within his group. He helps me to apply a scholarship for supporting my study. He motivates, guides and supports me for taking my first step in research and this area. He opens my mind and stays be with me to find the way and pursue the success at each stage. He makes my dream to write this dissertation possible and my future with many more possibilities. I am grateful for his patience and understanding in these three years.

I would also like to express my deepest gratitude to:

Prof. Dr. Steffen Becker. I got known of him from the beginning of writing my dissertation. He helps me and gives me a lot of valuable suggestions. His suggestions make me writing this dissertation with a much clearer mind, especially when I am confused about the motivation as well as the final evaluation of this dissertation.

Dr. Ivan Bogicevic. He is the first colleague that I met at our group. During

these three years, he helped me through each daily work at the same office. Especially, we had a great cooperation with a publication in the year 2017 concerning the 5th chapter in this dissertation. In addition, he helps me reviewing the whole dissertation word-by-word in the end.

Dr. Daniel Graziotin. I really want to thank this Italien. Especially, we had a great cooperation when I am finishing my PhD concerning the 7th chapter in this dissertation. This cooperation helps me understanding more about academic writing and lighting my passion for the further research.

Dr. Geir Kjetil Hanssen. When I started my research, I noticed the rigorous work in this research area about "Safe Scrum" from him and his colleagues. I feel so lucky that we got known of each other in the year 2016. I wrote this dissertation following the guidance from him, as well as "Safe Scrum". We discussed each year in his workshop (ASCS) align with the conference agile software development (XP), which keeps me running in the right way until finishing this dissertation.

Prof. Dr. Casper Lassenius and Prof. Dr. Ralf Küster for their sincere and valuable suggestions and opinions during the reviewing process and the coming defence.

My warm thanks go to the XP conferences for these three years and a lot of friends that I met there, who lighted the ideas in my dissertation: *Kent Beck, Dr. Steven Fraser, Prof. Dr. Juan Garbajosa, Dr. Xiaofeng Wang, Prof. Dr. Laurie Williams etc.* In addition, many thanks to the friends, Prof. Dr. Nancy Leveson, Dr. John Thomas, Prof. Dr. Robert Jan de Boer, Martin Rejzek etc., in STAMP conferences, who provide me a deep understanding and touch with STPA.

I am grateful to the current researchers and employees in the software engineering group that I collaborate with them in daily work: *Dr. Ivan Bogicevic, Wolfgang Fechner, Dr. Daniel Graziotin, Verena Käfer, Daniel Kulesz,*

Kornelia Kuhle, Kai Mindermann, Rainer Niedermayr, Dr. Jasmin Ramadani, Horst Vöhringer, Marvin Wyrich, Erica Weilemann. Especially, I would like to thank Jonas Fritzsch for reviewing the german language part.

Special thanks to the researchers and employees at Robert Bosch GmbH, *Stefan Kriso etc., Hans-Leo Ross etc.,* and *Yun Jiang etc.,* for providing me the opportunity to work with them and help me to conduct research in industry.

Last but not least, my sincerest gratitude goes to my family, *Prof. Dr. Lei Wang, Changxia Ma, Chunlai Wang, Yuanxi Wang, and little Kasi Wang.* I am grateful for their assistance, encouragement and bringing me happiness in every second of my life.

This dissertation would not have been possible without the support from my family, advisors, colleagues, industrial researchers and friends.

All the names are listed by alphabetical order of the surnames without prioritisation.

Stuttgart, October 2018
Yang Wang

List of Abbreviations

APS	*Autonomous Parking System*
ASD	*Agile Software Development*
ASIL	*Automotive Safety Integrity Level*
ATDD	*Acceptance Test-Driven Development*
BDD	*Behaviour-Driven Development*
CI	*Continuous Integration*
CSGS	*Crossroad Stop and Go System*
CTL	*Computation Tree Logic*
DSDM	*Dynamic System Development Method*
E/E/PE	*Electrical/Electronic/Programmable Electronic*
EUC	*Equipment Under Control*
FAA	*Federal Aviation Administration*
FASIC	*Firing Application Specific Integrated Circuit*

FET	*Field Effect Transistor*
FMEA	*Failure Mode and Effect Analysis*
FTA	*Fault Tree Analysis*
GSA	*Group Safety Analysis*
HIS	*High-Integrity Systems*
ICAO	*International Civil Aviation Organisation*
IDE	*Integrated Development Environment*
IF-FMEA	*Input-Focused FMEA*
IoT	*Internet of Things*
LTL	*Linear Temporal Logic*
MAD	*Median Absolute Deviation*
QA	*Quality Assurance*
RAMS	*Reliability, Availability, Maintainability and Safety*
R&D	*Research and Development*
RPN	*Risk Priory Number*
RUP	*Rational Unified Process*
SCS	*Safety-Critical Systems*
SIL	*Safety Integrity Level*
SMV	*Symbolic Model Verifier*
STPA	*System-Theoretic Process Analysis*
TDD	*Test-Driven Development*

UAT	*User Acceptance Testing*
UCA	*Unsafe Control Action*
XP	*Extreme Programming*

Zusammenfassung

Agile Software Entwicklung (ASD) hat seit vielen Jahren den Ruf erworben, höhere Kundenzufriedenheit, geringere Fehlerraten und schnellere Entwicklungszeiten zu erzielen und gilt als etablierte Vorgehensweise für sich schnell ändernde Anforderungen. Aufgrund sich permanent ändernder Märkte sowie kundenspezifischer Anforderungen weckt ASD daher aktuell das Interesse der Industrie im sicherheitskritischen Anwendungsbereich.

Die Nutzung von ASD zur Entwicklung sicherheitskritischer Systeme (SCS) wird jedoch kontrovers diskutiert. Die meisten Experten in diesem Bereich ziehen es in der Praxis vor, traditionelle Entwicklungsprozesse zusammen mit einem standardisierten Prozess für Sicherheitsaspekte zu verwenden, indem sie Normen wie die IEC 61508 erfüllen. Die bestehende Forschung strebt nach einer Konsistenz mit der Normen oder einem hybriden Modell zwischen ASD und den Normen. Die traditionellen Sicherheitstechniken können jedoch ohne eine solide Architektur des Systems nicht verlässlich funktionieren. ASD erlaubt eine sich ständig adaptierende Architektur, was die Integration traditioneller Sicherheitstechniken in ASD, insbesondere die Durchführung von Sicherheitsanalysen, zu einer Herausforderung macht.

In dieser Dissertation schlagen wir das Prozess-Modell S-Scrum vor, das hauptsächlich auf der Integration einer systemtheoretischen Prozessanalyse (STPA) basiert, um den sich ändernden Architekturen beim Einsatz von ASD

in der Entwicklung von SCS zu begegnen.

Die Forschungsstrategie dieser Dissertation lautet wie folgt: (1) Wir entwerfen eine theoretische Grundlage, ein vorläufiges S-Scrum Modell durch die Integration von STPA in Safe Scrum. Dieses vorläufige S-Scrum untersuchen wir anhand einer (in 2016 durchgeführten) Fallstudie basierend auf einem Studentenprojekt mit 14 Teilnehmern. (2) Wir verbessern die Sicherheitsverifikation, indem wir STPA-BDD vorschlagen (BDD steht für Behaviour-Driven Development). Durch ein kontrolliertes Experiment mit insgesamt 55 Teilnehmern validieren wir STPA-BDD. (3) Wir verbessern die Dokumentation, indem wir drei Dokumente für das vorläufige S-Scrum entwickeln und adaptieren. Diese Dokumente validieren wir anhand einer Fallstudie während eines einjährigen Studentenprojekts mit 14 Teilnehmern. (4) Wir verbessern die Kommunikation für das vorläufige S-Scrum. Es werden die bestehenden Kommunikationskanäle, ihre Ziele und Herausforderungen bei der Sicherheitsanalyse und -verifikation untersucht. Hierzu führen wir eine Fallstudie in sieben sicherheitskritischen Unternehmen mit 60 Experten durch. (5) Wir verbessern die Gruppenarbeit durch Vermeidung von Gruppendenken bei der Durchführung von Sicherheitsanalysen und -verifikationen im vorläufigen S-Scrum. Unsere Lösungen untersuchen wir im Rahmen der Durchführung einer industriellen Fallstudie in sieben sicherheitskritischen Unternehmen mit 39 Experten. (6) Resultierend schlagen wir ein finales S-Scrum Modell vor. Wir evaluieren dieses Modell durch informelles Review und Walkthrough mit weiteren 15 Befragungen von Sicherheitsexperten aus sechs großen sicherheitskritischen Unternehmen.

Die Ergebnisse dieser Dissertation sind: (1) Das vorläufige S-Scrum Modell verbessert die Möglichkeiten zur Gewährleistung der Sicherheit, die Agilität wird dabei jedoch leicht reduziert. Wir haben die großen Herausforderungen bei der Priorisierung, Kommunikation, Planung und Verifikation von Anforderungen untersucht. Hierfür schlagen wir erste Lösungsansätze vor, beispielsweise einen internen und externen Sicherheitsexperten. (2) Das STPA-BDD für die Sicherheitsanalyse und -verifikation stellt eine effektive Lösung hinsichtlich Produktivität, Testdurchgängigkeit, Fehlererkennungseffektivität und Kommunikationseffektivität dar. (3) Die verbesserte Dokumentation mit

Safety-Stories und Safety-Epics wirkt sich positiv auf die Kommunikation aus. Der agile Sicherheitsplan unterstützt hier bei der Planung und Zertifizierung. (4) Die vorrangig genutzten Kommunikationskanäle sind regelmäßige Meetings, die meisten von ihnen werden 1-4 mal pro Woche durchgeführt. Wir haben 28 Zielsetzungen wie beispielsweise Sicherheitsanforderungen für die Übertragung identifiziert, sowie die 10 größten Herausforderungen in diesem Kontext, wie sensible und vertrauliche Informationen überwachen. (6) Gruppendenken gibt es bei der Durchführung von Sicherheitsanalysen und Verifizierungen sowohl in unseren Forschungskontexten, als auch im vorläufigen S-Scrum Modell. Das Wichtigste der 10 gefundenen Phänomene des Gruppendenkens ist Manager planen zu optimistisch. Wir konnten Gründe hierfür identifizieren, z.B. hohe Kohäsion des Teams und entsprechende Lösungen z.B. Einladung externer Experten vorschlagen, um Gruppendenken zu vermeiden. (6) Das finale S-Scrum zeigt eine gute Fähigkeit zur Gewährleistung der Sicherheit nach ISO 26262. Nach unserer Analyse kann S-Scrum 97.3% der Anforderungen nach ISO 26262 mit ASIL A und ASIL B erreichen (ASIL steht für Automotive Safety Integrity Level).

Zusammenfassend ist S-Scrum eine mögliche Variante der ASD in der Entwicklung von SCS, sowohl für akademische als auch für industrielle Projekte der Klassen ASIL A und ASIL B.

Diese Dissertation bietet Hilfestellungen für Praktiker, z.B. (1) einen realisierbaren Weg zur Entwicklung von SCS im Rahmen des ASD, (2) die Verwendung von individuellen Techniken wie STPA und BDD in ASD, (3) die Unterstützung des Sicherheitsmanagements, z.B. durch eine verbesserte Formulierung der Dokumentation, sowie eine effektive Kommunikation und Gruppenarbeit während der Sicherheitsanalyse und -verifizierung. Für Forscher bedeutet diese Dissertation (1) die Erschließung neuer Wege, Techniken für die Sicherheit wie STPA und BDD innerhalb des ASD nutzbar zu machen, (2) eine mögliche Forschungsrichtung durch die Erprobung neuer Techniken im Gegensatz zur Verwendung traditioneller Vorgehensweisen auf Basis bestehender Normen, sowie (3) die gesonderte Berücksichtigung der Aspekte menschlicher Interaktion im Sicherheitsmanagement, z.B. Gruppendenken und Kommunikationskanäle.

Dennoch besitzt das S-Scrum Modell Einschränkungen: (1) S-Scrum wurde nicht in einem realen industriellen Projekt evaluiert. (2) Fahrzeugsysteme auf Basis der Risiko-Klassen ASIL C und ASIL D sollten die Verwendung von S-Scrum vermeiden (bei Verwendung von S-Scrum sollte eine Abkehr von ASIL in Betracht gezogen werden). (3) Hardwarenahe oder Embedded-Systeme mit einer hohen Hardware-Kohäsion sind für S-Scrum nicht geeignet. (4) Großangelegte oder verteilte Teams sollten bei der Einführung von S-Scrum mit Bedacht vorgehen oder dieses durch ggf. erforderliche Erweiterungen anpassen.

Für Folgearbeiten wird empfohlen, die Einschränkungen der in dieser Dissertation erforschten Konzepte zu reduzieren, z.B. durch die verstärkte Betrachtung der Umsetzung von S-Scrum im Automotive-Sektor. Um die heutigen Probleme bei der Entwicklung sicherheitskritischer Systeme zu bewältigen, könnten in diesem Zusammenhang auch andere kritische Aspekte wie Informationssicherheit Berücksichtigung finden.

Abstract

Agile software development (ASD) has gained a good reputation for a number of years due to its higher customer satisfaction, lower defect rates, faster development times and as a solution to rapidly changing requirements. Thus, ASD arouses interests from safety-critical industries due to a fast changing market and upcoming customised requirements.

However, applying ASD to develop safety-critical systems (SCS) is controversial. Most of practitioners in SCS prefer using traditional development processes together with a standardised safety assurance process by satisfying the norms, such as IEC 61508. Existing research is striving for a consistency or a hybrid model between ASD and norms. However, the traditional safety assurance cannot work well without a stable architecture. ASD has a constantly changing architecture, which makes the integration of traditional safety assurance in ASD a bottleneck, especially the execution of safety analysis.

In this dissertation, we aim to propose a process model called *S-Scrum*, which is mainly based on integrating a *System-Theoretic Process Analysis (STPA)* to face the changing architectures when using ASD for developing SCS.

The *research strategy* of this dissertation is: (1) We build a theoretical foundation, a preliminary S-Scrum, through integrating STPA in Safe Scrum.

We explore the preliminary S-Scrum by conducting a case study in a one-year student project with 14 participants. (2) We improve the safety verification by proposing STPA-BDD. We validate STPA-BDD by conducting controlled experiments with overall 55 participants. (3) We improve the documentation. We adapt and develop three documents for the preliminary S-Scrum. We validate them in a one-year student project by conducting a case study with 14 participants. (4) We improve the communication. We investigate the existing communication channels, their purposes and challenges during safety analysis and verification by conducting a case study in 7 safety-critical companies with 60 experts. (5) We improve group work through avoiding groupthink when performing safety analysis and verification in ASD. We investigate the solutions by conducting an industrial case study in the same 7 safety-critical companies with 39 experts. (6) To this end, we generate a final S-Scrum. We evaluate it by conducting one round of informal review, one round of walkthrough, together with 15 interviews with senior-level safety experts from 6 large safety-critical companies.

The *results* of this dissertation are: (1) The preliminary S-Scrum enhances the possibilities to ensure safety, yet the agility is slightly reduced. We explored major challenges in *requirements prioritisation, communication, planning and verification.* We propose initial solutions, such as adding an internal and external safety expert. (2) STPA-BDD for safety analysis and verification seems effective on productivity, test thoroughness, fault detection effectiveness and communication effectiveness. (3) The improved documentation, namely safety story and safety epic, have a positive effect on communication. Agile safety plan supports more on planning and certification. (4) The most popular communication channel is "formal meetings". Most of them happen 1-4 times per week. We found 28 purposes like "transfer safety requirements" and the Top 10 challenges like "monitor sensitive and confidential information". (5) Groupthink does exist when performing safety analysis and verification in our research contexts as well as in S-Scrum. The Top 10 phenomena of groupthink include "managers are too optimistic on the plan". We investigate reasons like "high cohesiveness of the team" and propose solutions like "inviting external expert" to avoid groupthink.

(6) The final S-Scrum shows a good capability on safety assurance for developing SCS according to ISO 26262. S-Scrum can achieve 97.3% of the requirements from the ISO 26262 for the SCS with ASIL A and ASIL B.

In *conclusion*, to some extent, S-Scrum is a possible ASD for developing academic and automotive industrial projects with ASIL A and ASIL B.

This dissertation provides *implications* for practitioners, such as (1) a feasible ASD development process for developing SCS, (2) the use of individual techniques, such as STPA and BDD in ASD, (3) the support for safety management, such as the formulation of documentation, facilitating an effective communication and group work during safety analysis and verification. For researchers, this dissertation implies (1) new techniques including STPA and BDD for safety assurance in ASD, (2) a possible research direction through investigating new techniques rather than purely using the traditional techniques from norms, (3) the specific consideration of human aspects in safety-critical system's organisation management, such as groupthink and communication channels.

Nevertheless, S-Scrum has *limitations*: (1) S-Scrum has not been evaluated in a real-world industrial project. (2) Automotive systems with ASIL C and ASIL D should avoid adopting S-Scrum (when using S-Scrum, a degradation of ASIL should be considered). (3) Hardware systems or embedded systems with a high hardware cohesion are not suitable for S-Scrum. (4) Large-scale or distributed teams should take special care or extend S-Scrum with possible variations when adopting S-Scrum.

In the future, the immediate work is to narrow down the limitations in this dissertation, such as *implementing S-Scrum in a (more) automotive project(s)*. On the other hand, to face today's SCS, other critical aspects can be taken into account, such as *information security*.

Contents

Chapter 1

Introduction

In this chapter, we describe the motivation, problem statement, research objective, contribution, list of publications as well as the outline of this dissertation.

1.1 Motivation

Agile software development (ASD) brings a lot of benefits to modern software engineering, such as fast delivered products, fast feedback, fast responding to change and enhanced communication and cooperation [Coc02]. It contains software development methodologies, development processes, as well as other techniques under agile values and principles including requirements specification or business development and management, such as Extreme Programming (XP) [Bec00], Scrum [Sch95], Rational Unified Process (RUP) [Kru04], Dynamic Systems Development Method (DSDM) [Sta97], Feature Driven Development (FDD) [PF01] and Lean Software Development [Pop07].

Safety-critical systems (SCS) perform safety assurance following norms[1] [Bow93], such as IEC 61508 [Com11a] in general Electrical/Electronic/Programmable Electronic (E/E/PE) safety-related systems, ISO 26262 [Sta11] in automotive systems and DO-178 B/C [Aer12] in avionic systems. These norms regulate safety assurance activities covering the whole product development lifecycle together with recommended techniques. The industries are trying to satisfy these requirements to reach a high degree of safety assurance capability for certification.

However, traditional development processes contain drawbacks in developing SCS and further lead to a lot of potential costs, such as to manage requirements volatility, when introducing emerging technologies as well as producing and maintaining documentation [GPM10]. ASD becomes a possible solution.

To implement ASD in SCS, researchers prefer seeking a consistency or a combination between ASD and norms. Safe Scrum [SMH12] is representative in combining scrum with IEC 61508. The model adds traditional safety assurance activities into scrum outside each iteration. They increase documents including agile safety plan and safety product backlog as well as a role like safety expert. However, few current research focuses on safety assurance inside each iteration due to a changing architecture [Kni02]. Safety analysis

[1] In the dissertation, we use "norm" and "standard" synonymously.

seems difficult to work.

The norms recommend techniques and organisational management of safety analysis. The popular techniques include Failure Mode and Effect Analysis (FMEA) [Sta03a] and Fault Tree Analysis (FTA) [Lar74]. FMEA is a table-based technique, while FTA derives a figure called fault tree. The models need a detailed architecture to start their analyses. They are based on reliability theory, which considers that the hazards are raising from single component's or function's failure. However, today's software-intensive sociotechnical systems are becoming sophisticated, hazards are caused more from component interactions, cognitively complex human decision-making errors and social, organisational and management factors [Lev11]. We need to analyse safety based on *systems theory*.

System-Theoretic Process Analysis (STPA) is a new safety analysis technique proposed by Leveson in 2012 [Lev11], which is based on systems theory. STPA considers the accidents are caused by unsafe control actions (UCAs). It has been successfully used in a lot of domains, such as medical devices [Ant13], air traffic management [FL15] and railway systems [Fan+15]. In addition, STPA supports *safety-guided design*. The UCA can be derived from a high-level architecture. Hence, we start this dissertation by integrating STPA in scrum.

1.2 Problem Statement

Current safety assurance in ASD for developing SCS is still immature. The existing research prefers using hybrid models by combining ASD with traditional development processes based on norms. Yet, the norms are not totally suitable for regulating and guiding the ASD, since there is a lack of suitable safety analysis techniques to face the changing architectures.

In addition, safety analysis needs safety verification to ensure that the safety requirements have been implemented in the systems. The traditional safety verification, such as model checking, is not suitable in ASD. First, it needs a model, which does not exist in ASD. Second, the formal specification

increases the difficulties to communicate, which should not be neglected when developing SCS [MG17].

More than that, practitioners need special considerations on the changed safety management from traditional development processes to ASD with integrated systems theory-based safety analysis and verification. It includes: (1) Documentation. The norms regulate a huge amount of documents, whereas ASD advocates lightweight documents. It seems to be a weakness for keeping traceability. (2) Communication. ASD supports communication. Yet, the communication during systems theory-based safety analysis and verification encompasses challenges, such as a misunderstanding between safety expert and development team. An ineffective communication among group members negatively influences their execution. (3) Group work. systems theory-based safety analysis and verification in ASD have more group works than in traditional development processes. These group works happen among multiple functional departments. An inappropriate control might lead to an ineffective group work. During the execution of systems theory-based safety analysis and verification, the occurrence of groupthink might cause a sub-optimal or an unsafe design decision.

1.3 Research Objective

This dissertation aims at proposing a scrum development process, S-Scrum, based on *systems theory* to apply ASD in SCS. From the technique viewpoint, we integrate STPA and BDD in an existing scrum development process for SCS, which is called Safe Scrum. From the management viewpoint, we improve documentation, communication and group work through avoiding groupthink when performing STPA and BDD. To this end, we evaluate our final S-Scrum by reviewing ISO 26262 to make S-Scrum applicable both in academia and industry.

1.4 Contribution

This dissertation provides four contributions:

- **We propose a preliminary S-Scrum by integrating STPA.** To solve the problem concerning a lack of safety analysis in ASD, we propose a preliminary S-Scrum by integrating a new systems theory-based safety analysis technique, STPA, in an existing scrum development process, Safe Scrum. The preliminary S-Scrum covers documents, roles and activities. We illustrate the preliminary S-Scrum with a running example - Airbag System. We explore the preliminary S-Scrum in a one-year student project - "Smart Home" with 14 participants. The results show that the preliminary S-Scrum has a good capability on ensuring safety and agility, while challenges exist.
- **We improve the preliminary S-Scrum by proposing STPA-BDD.** To improve the challenges concerning verification and communication in the preliminary S-Scrum, we propose STPA-BDD. We validate it in two controlled experiments with 55 participants. Through developing a semi-automated tool, STPA-BDD shows a good capability on productivity (7 times greater), test thoroughness (1.5 times greater), fault detection effectiveness (2 timers greater) and communication effectiveness compared with STPA-UAT.
- **We enhance the safety management of the preliminary S-Scrum concerning documentation, communication, group work through avoiding groupthink.** To improve the challenges concerning documentation (planning), communication and group work to avoid groupthink (requirements prioritisation) in the preliminary S-Scrum. We adapt and develop three documents: Safety epic; Safety story; Agile safety plan. We evaluate them by conducting a case study in a one-year student project with 14 participants. In addition, we investigate existing communication channels, their purposes and their challenges during safety analysis and verification in 7 safety-critical companies with 60 experts. We further map our results in S-Scrum. Finally, we notice the

negative influences of groupthink in safety analysis and verification. We conduct a case study in the same 7 safety-critical companies and find the Top 10 phenomena together with their reasons and solutions in our research context as well as in S-Scrum.

- **We propose and evaluate the final S-Scrum by reviewing ISO 26262.** To evaluate the applicability of the final S-Scrum in both academia and industry, we conduct one round of informal review by the author and one round of walkthrough by one certified safety expert and one safety expert with more than 20 years working experiences in the area of functional safety. More than that, 15 interviews are conducted with senior-level safety experts in 6 safety-critical companies. S-Scrum is evaluated to be a possible ASD for developing SCS in both academic projects and partially industrial projects assigned an ASIL A or ASIL B.

1.5 List of Publications

This dissertation is based on the following publications where I am the first author:

- Yang Wang, Stefan Wagner. *Toward Integrating a System Theoretic Safety Analysis in an Agile Development Process*, published in the Proceeding of the 2016 Software Engineering Conference (CSE, SE 2016).
- Yang Wang, Stefan Wagner. *Towards Applying a Safety Analysis and Verification Method Based on STPA to Agile Software Development*, published in the Proceeding of the 2016 International Conference of Software Engineering (CSED, ICSE 2016).
- Yang Wang, Jasmin Ramadani, Stefan Wagner. *An Exploratory Study of Applying a Scrum Development Process for Safety-Critical Systems*, published in the Proceeding of the 2017 International Conference on Product-Focused Software Process Improvement (PROFES 2017).
- Yang Wang, Ivan Bogicevic, Stefan Wagner. *A Study of Safety Documentation in a Scrum Development Process*, published in the Proceeding

of the 2017 International Conference on Agile Software Development (ASCS, XP 2017).

- Yang Wang, Stefan Wagner. *Combining STPA and BDD for Safety Analysis and Verification in Agile Development*, published in the 2018 STAMP Conference, MIT (STAMP 2018).
- Yang Wang, Stefan Wagner. *Combining STPA and BDD for Safety Analysis and Verification in Agile Development: A Controlled Experiment*, published in the Proceeding of the 2018 International Conference on Agile Software Development (XP 2018).
- Yang Wang, Daniel Ryan Degutis, Stefan Wagner. *Speed up BDD for Safety Verification in Agile Development: A Partially Replicated Controlled Experiment*, published in the Proceeding of the 2018 International Conference on Agile Software Development (ASCS, XP 2018).
- Yang Wang, Stefan Wagner. *Combining STPA and BDD for Safety Analysis and Verification in Agile Development*, published in the Proceeding of the 2018 International Conference on Software Engineering Companion (ICSE 2018).
- Yang Wang, Stefan Wagner. *On Groupthink in Safety Analysis: An Industrial Case Study*, published in the Proceeding of the 2018 International Conference on Software Engineering (ICSE-SEIP 2018).
- Yang Wang, Daniel Graziotin, Stefan Wagner. *An Industrial Case Study on Communication Channels in Safety Analysis*, submitted to Journal of Systems and Software (JSS 2018).

1.6 Outline

The remainder of this dissertation is organised as follows: In Chapter 2, we present the background of this dissertation, including: (1) General agile development and especially scrum; (2) General safety analysis techniques and especially STPA; (3) General safety verification techniques and especially BDD; (4) General safety system management concerning documentation,

communication and groupthink. The state of the art is presented in Chapter 3. In Chapter 4, we present our preliminary S-Scrum. The improvement by integrating BDD in S-Scrum is presented in Chapter 5. Chapter 6 presents the improved documentation, while Chapter 7 presents the communication channels. In Chapter 8, we improve the group work in S-Scrum through avoiding groupthink. In Chapter 9, we propose the final S-Scrum and evaluate it by reviewing ISO 26262. In Chapter 10, we conclude this dissertation with a discussion of implications as well as limitations and propose future work.

Chapter 2

Background

In this chapter, we provide an overview of the background to this dissertation. The topics are ASD, safety analysis, safety verification, documentation, communication and groupthink.

2.1 Agile Software Development

2.1.1 General Agile Software Development

ASD was introduced around the year 1990 [CLC04]. Agile Alliance [All] defines ASD as "*the ability to create and respond to change in order to succeed in an uncertain and turbulent environment.*" Many practitioners provide their understandings, such as Ericksson et al. [ELS05] think ASD as "*a way to strip away as much of the heaviness, commonly associated with the traditional software development methodologies, as possible to promote quick response to changing environments, changes in user requirements, accelerated project deadlines and the like;*" Williams and Cockburn [WC03] consider ASD as "*embracing rather than rejecting fast feedback and higher rates of change.*"

In 2001, 17 signatories proclaimed an "Agile Manifesto" [All01], which encompasses 12 principles within 4 core values. These are individuals and interactions over processes and tools, working software over comprehensive documentation, customer collaboration over contract negotiation and responding to change over following a plan.

In Table 2.1, we summarise the main agile methods. Extreme Programming (XP) was developed in 1999 by Kent Beck [Bec00]. It encompasses a set of agile practices derived from four basic activities: Coding, such as pair programming; Testing, such as TDD; Listening, such as planning game; Designing, such as refactoring. XP advocates the agile principles including feedback, assuming simplicity, and embracing change. Scrum [SB02] is a development process for project management including roles like a scrum master, workflows including daily scrum meeting, artifacts including product backlog and tools including Jira. Kanban [Bre15] is a specific visualized scheduling system by using mainly a board. DSDM [Sta97] covers the whole development lifecycle. It uses "MoSCoW" prioritisation with "musts", "shoulds", "coulds" and "won't haves" to adjust the project to meet the time constraint. Feature Driven Development (FDD) blends a number of industry-recognised best practices, which are driven from a client-valued functionality (feature) perspective to develop an overall model, build feature list, plan

by feature, design by feature and build by feature [PF01]. Lean software development adapts seven principles from production system to software development including eliminate waste, amplify learning, decide as late as possible, deliver as fast as possible, empower the team, build integrity, and see the whole [Pop07].

During the last three decades, the research in ASD has come from practical applications. The domains span the spectrum from pure software-intensive systems to sophisticated cyber-physical systems. Dybå and Dingsøyr [DD08] offer a currently representative picture by conducting a systematic literature review to summarise recent research on ASD. The results show that ASD can enhance productivity and satisfaction from both developers and customers. Gregory et al. [Gre+16] propose further challenges of ASD in terms of organisation, sustainability, culture and teams. Another recurring practical challenge in ASD is how to handle human and social factors [GN16].

Table 2.1: Main Agile Methods

Agile Method	Major Area	Example	Reference
XP	Agile practice	Pair programming TDD Planning game Refactoring	[Bec00] [Bec04]
Scrum	Project management	Scrum master Daily scrum meeting Product backlog	[SB02] [Rub12]
DSDM	Development lifecycle	Timeboxing MoSCoW	[Sta97] [Sta03b]
FDD	Requirements specification	Domain object modelling Developing by feature	[PF01]
Lean	Production Software development	Lean principles	[Pop07]
Kanban	Agile practice	Kanban board	[HS14] [Bre15]

2.1.2 Scrum

In this dissertation, we mainly use scrum development process. Scrum was firstly published by Ken Schwaber at OPPSLA 1995 [Sch95]. Since then, several scrum-specific publications have appeared [SB02] [Sch04] [SS11].

Scrum is a development process, which adopts agile values and principles for organising and managing work. It advocates honesty, openness, courage, respect, focus, trust, empowerment and collaboration [Rub12].

Scrum consists of one or more scrum teams including three *scrum roles*: product owner, scrum master and development team. The product owner is responsible for what will be developed and in what order, while the scrum master is responsible for guiding the team in creating and following the scrum framework. The development team is self-organised and cross-functional including coding, testing, design and so on. The *scrum workflow* includes: sprint, sprint planning meeting, daily scrum meeting, sprint review meeting and sprint retrospective meeting. Other extended activities like progress refinement for understanding the product backlog or two-part sprint planning for planning product backlog items in details are also used in practice. The *scrum artifacts* are: product backlog and sprint backlog. The product backlog comprises an ordered list of requirements that a scrum team maintains for a product, while the sprint backlog is the list of work the development team must address during the coming sprint. Other extensions like story map for an overview of the product and burn-down chart for controlling the process are also popular.

2.2 System-Theoretic Safety Analysis

2.2.1 General Safety Analysis

Safety analysis is important in safety assurance [Eri15]. IEC 61508-1: 2011 defines safety analysis as being used "*to determine the hazards, hazardous events and hazardous situations relating to the Equipment Under Control (EUC) and the EUC control system; to determine the event sequences leading to the hazardous events; to determine the EUC risks associated with the hazardous events*". ISO 26262-1: 2011 defines safety analysis as being used "*to identify and categorise hazard events of items and to specify safety goals and Automotive Safety Integrity Level (ASILs) related to the prevention or mitigation of the associated hazards in oder to avoid unreasonable risk.*" Leveson [Lev11] defines

safety analysis as being used "*to identify potential causes of accidents, that is, scenarios that can lead to losses.*" as well as mentioning the importance "*to investigate an accident before damage occurs.*"

The most popular safety analysis techniques are Failure Mode and Effect Analysis (FMEA) and Fault Tree Analysis (FTA) [Eri15].

FMEA was proposed by the U.S. military for weapons systems in 1949. It is used to define, identify, and eliminate known and/or potential failures, problems, errors, and so on from the system, design, process and/or service before they reach the customer. FMEA analysis involves three steps: (1) Identify known and potential failure modes; (2) Identify the causes and effects of each failure mode; (3) Prioritise the identified failure modes according to the RPN - the product of frequency of occurrence, severity, and detection; (4) Provide for problem follow-up and corrective action [Sta03a].

FTA was proposed in 1961 for the Air Force for evaluation of the Minuteman Launch Control System. It is a logic diagram used for analysing, visualising and evaluating failure paths in a system [Lar74]. FTA analysis involves five steps: (1) Define the undesired event to study; (2) Obtain an understanding of the system; (3) Construct the fault tree; (4) Evaluate the fault tree; (5) Control the hazards identified.

2.2.2 STPA

In this dissertation, we propose to use System-Theoretic Process Analysis (STPA) for performing safety analysis in ASD.

STPA [Lev11] is a relative new safety analysis technique based on Systems-Theoretic Accident Model and Process (STAMP) causality model. STAMP is built on three basic concepts: (1) Safety constraints; (2) A hierarchical safety control structure; (3) Process models. In STAMP, systems are viewed as interrelated components kept in a state of dynamic equilibrium by feedback control loops. An accident occurs, because either the safety constraints were not enforced by the controller, or appropriate control actions were provided but not followed [Lev11].

STPA aims to identify potential causes of accidents and eliminate and control in design or operations before damage occurs. The execution of STPA consists of two main steps: (1) Identify the potential for inadequate control of the system that could lead to a hazardous state; (2) Determine how each potentially hazardous control action identified in step 1 could occur. Before safety analysis starts, we identify accidents and hazards at the system-level. To apply STPA, we start from a control structure of the system[1]. Based on the control structure, we evaluate each control action against four general types of hazardous behaviour: (1) A control action required for safety is not provided; (2) An unsafe action is provided; (3) A potentially safe control action is provided too early, too late or out of sequence; (4) A safe control action is stopped too soon or applied too long. Then we derive the initial safety requirements from the UCA. By using STPA step 2, we focus on causal factors for the UCA of step 1. We identify a process model and variables that affect the safety of the control actions and include them in the software controller in the control structure diagram to document how each UCA could occur. The process model contains three types of variables: Internal variables of the software controller; Interaction interface variables, which receive data/command of the environmental components; Environmental variables of other components in the system interacting with the software controller. After that, we identify detailed software safety requirements to constrain the unsafe combinations of process variables.

In comparison with FMEA and FTA, STPA is based on systems theory rather than reliability theory. In reliability theory, accidents are caused by single component's and function's failures in a chain of events. However, due to an increasing complexity of systems, accidents are more often resulted from inadequate control actions among multiple dysfunctional failures, as well as human-machine interactions and organisational management [Lev11]. To ensure the safety of today's sophisticated SCS, safety analysis should be based on systems theory.

In addition, FMEA and FTA need a big design upfront, whereas STPA can

[1] Before performing STPA, we assume that system design exists.

start from a high-level architecture to guide design. The modern market for SCS is fast-changing. A big design upfront seems too heavyweight to face the increasing number of customised requirements. The rework on a detailed architecture after safety analysis requires much more effort, while process-end safety analysis relevant activities are easy to cut when facing a strict time limitation on delivery. Thus, safety analysis needs to be able to start from a high-level architecture, as well as happen before, or in parallel with, development, especially in ASD, to avoid the effort on reworking and an added-on safety assurance.

2.3 Safety Verification

2.3.1 General Safety Verification

Safety verification is used for verifying safety requirements, which are derived from safety analysis. Safety verification is defined in IEC 61508-1: 2011 as a process "*to demonstrate, for each phase of the overall, E/E/PE system and software safety lifecycle (by review, analysis and/or tests), that the outputs meet in all respects the objectives and requirements specified for the phase.*" ISO 26262-1: 2011 defines safety verification as a process "*to determine the completeness and correct specification or implementation of requirements from a phase or sub-phase.*"

Most of the practitioners mix unit test, integration test, system test, field test and user acceptance testing (UAT) to verify safety requirements [CHR17]. These existing methods perform safety verification either through reviewing requirements in a meeting, or mixing with functional testing during development. However, according to systems theory, safety requirements constrain system behaviours. Thus, safety verification should focus on verifying system behaviours.

Formal methods [Bow93] are popular for safety verification, such as model checking [CGP99] by using Cadence SMV in the nuclear power domain [YJC09] or temporal logic LTL/CTL in the automotive domain [AW14]. Model checking consists of three steps: (1) Formulating safety requirements

by using formal language; (2) Modeling source code as an input model; (3) The safety requirements specification and the input model are verified by using a model checker. In this way, the safety requirements of system behaviours can be generated into formal language and verified within a source code model. However, model checking is a kind of black-box mode. When using model checking in ASD, it shows weaknesses to guide design as well as an effective communication among team members.

2.3.2 BDD

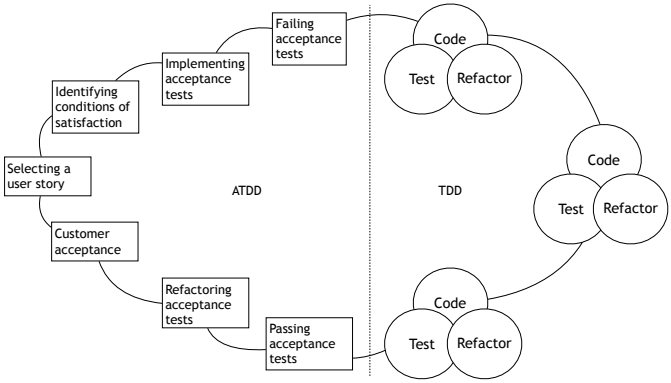


Figure 2.1: TDD and ATDD [Coh10]

In this dissertation, we propose using BDD for performing safety verification in ASD.

BDD is an agile method for automated testing. BDD is in the family of test-driven development (TDD) and acceptance test-driven development (ATDD). As we can see in Figure 2.1, ATDD is on the left side for verifying user stories, while TDD is on the right side for verifying functional units. BDD, however, starts with a failing acceptance test to describe the *behaviour* of a system in scenarios. BDD implements and passes the acceptance test further. Additionally, BDD can work from outside-in to write a failing unit test and pass the failing unit test through coding and refactoring.

BDD rethinks TDD and ATDD by using natural language or domain-specific language to describe system behaviours. BDD specifies that business analysts and developers should collaborate in this area and should specify behaviour in terms of user stories, which supports an effective communication.

BDD is executed by implementing a template to describe scenarios and generate test cases with three main steps: Given[Context], When[Event], Then[Outcome]. The context describes pre-conditions or system states, the event describes a trigger event, and the outcome is an expected or unexpected system behaviour. The descriptions in square brackets are in ubiquitous language. The business analyst, QA and developer merge the scenarios in a "3 Amigos Meeting" [WH12]. The problems, possible scenarios, solutions and test cases are generated from business-level to development-level.

In summary, (1) BDD aims to test system behaviours; (2) BDD performs testing before development; (3) BDD uses natural language or domain-specific language to support an effective communication between developers and business analysts.

2.4 Safety Management

2.4.1 General Safety Management

An effective execution of safety analysis and verification is correlated with its relevant management.

Safety management was originally popular used in aviation systems. The Federal Aviation Administration (FAA) defines safety management as "*a formal, top-down, organisation-wide approach to managing safety risk and assuring the effectiveness of safety risk controls. It includes systematic procedures, practices, and policies for the management of safety risk.*" The International Civil Aviation Organisation (ICAO) defines safety management as "*safety management identifies hazards with the potential to adversely affect safety. It provides effective and objective mechanisms to assess the risk presented by hazards and implement ways to eliminate these hazards or mitigate the risks*

associated with them." Recently, E/E/PE safety-related industry [Com11a], automotive industry [Sta11] and airborne industry [Aer12] has established safety management in their own organisations.

Safety management consists of commitment and responsibility, accountabilities, appointment of key safety personnel, coordination of emergency response planning, documentation, risk identification, assessment and mitigation, safety monitoring and measurement, change management, continuous improvement, safety training and education, and communication [Org13].

ASD promotes self-organised and adaptive management. The core values of ASD have conflicts with the traditional safety management, such as a huge amount of requirements on documentation [BT05]. Documents in traditional safety management are important for keeping traceability as well as certification. Yet, ASD generates documents for supporting communication and collaboration in a lightweight way. In addition, safety analysis and verification are performed across multiple functional departments due to the complex SCS. Communication occurs frequently, such as through formal meetings or emails. When using ASD, communication happens even throughout daily work. The management of communication influences the quality of safety analysis and verification. Moreover, group works are becoming more numerous than before when using ASD. The team members have diverse knowledge backgrounds. An effective group work among them influences decision making. We notice that groupthink is the most notorious.

Therefore, we investigate safety management in ASD through improving documentation, enhancing communication and avoiding groupthink in group work in this dissertation. We illustrate them in detail in the following sections.

2.4.2 Documentation

Documentation is important for keeping traceability and providing evidence for certification of safety management [Com11a]. The norms concerning functional safety, such as ISO 26262, regulate documentation throughout the overall project lifecycle as work products. These work products provide

not only the proof that the safety management is in place, but also the review and understanding of the documentation, which provide a continuous improvement of the system.

The documentation in safety management includes: (1) The documentation required by the policy, which is generated before establishing a project. (2) The documents concerning legal applicability of regulations, which is generated during the setting of a project. (3) The documentation for organisational structure and responsibilities, which is generated during the planning of a project. (4) The documentation of the objective and goal of the system, such as functional safety concept, which is generated when performing system analysis. (5) The documentation concerning process activities, such as a safety plan, which is generated when performing high-level safety analysis. (6) The documentation of data and measures, such as functional safety assessment/audit plan, which is generated after performing the high-level safety analysis. (7) The documentation of change management, such as a change report, which is generated whenever there are changing requirements during development. (8) The documentation of review, such as a verification report, which is generated after the execution of safety verification. (9) The documentation supporting maintenance, such as a documentation management report, which can be generated before delivering a product or at the beginning of a project. Modern software development including ASD prefers to use lightweight documents, such as "user stories" for recording requirements and "story map" for planning. The practitioners use them also for QA documents including safety.

2.4.3 Communication

Communication has implications from a philosophical perspective, such as a system theoretical approach, which depicts communication messages as being created, delivered and received in a complex web of relationships (safety analysis happens in a complex environment in industry), as well as a critical culture perspective, which points out that critical culture encourages the exploration of alternative communication channels (safety analysis, as a

critical factor, is greatly influenced by safety culture[1]) [Key17].

In practice, communication channels are used with diverse taxonomies, such as verbal or nonverbal, synchronous or asynchronous and internal or external. Johnson et al. [Joh+94] illustrated the use of communication channels in different taxonomies through 380 respondents of surveys in a large midwestern state governmental agency. The authors mentioned that choosing between formal or informal communication channels, perceived applicability, output's effect and cultural norms are three possible criteria. Traditional safety-critical systems have a preference for formal communication to ensure preciseness [BS93]. Yet, many informal communication channels arise in modern software development.

Modern software development advocates face-to-face conversation as the best form of communication [All01]. The practical activities encompass pair programming [WK02] to enhance communication between developers or stand up meetings [SB02] to enhance communication among team members.

Developing SCS encompasses specific communication challenges. For example, Dobson, Moors and Norris [DMN14] illustrate the tragic loss in an Esso gas plant explosion in 2001, which includes the hazard from a combination of ineffective communication and inadequate hazard assessment. They pointed out that: (1) There is a link between communication and safety; (2) There are a range of mechanisms by which communication can fail; (3) There are a range of factors that shape the safety of communications; (4) Formal communication is most effective but needs to be used appropriately.

Possible communication *problems* in SCS include missing, unnecessary, inaccurate, poor quality and ambiguous information. Improvements might be a careful specification, the utilisation of aids such as logs, the development of communication skills and the setting of standards for effective and safe communication [Pri10]. Some instinct *causalities* of communication problems in SCS can be classified into internal and external causalities. Internal causalities can be attributed to characteristics of individuals, while external

[1] Safety culture is the attitude, beliefs, perceptions and values that employees share in relation to safety in the workplace. Safety culture is a part of organisational culture, and has been described by the phrase "the way we do things around here".

causalities can be attributed to environments. The *recommendations* for improving communication in SCS are mentioned as explicitness, timing and assertiveness [FOC08].

2.4.4 Groupthink

Groupthink [Jan08] is a psychological phenomenon. It was introduced by Janis in 1972. People under groupthink try to minimise conflicts and reach a consensus decision without critical evaluation of alternative viewpoints. As we can see in Figure 2.2, Janis summarised a linear model of how seven antecedents, which are cohesion, group insulation, an impartial leader, lack of norms, homogeneous, high stress from external threats and temporarily low-self esteem, increase the likelihood of groupthink, which leads to eight psychological symptoms, which are illusion of invulnerability, collective rationalisations, belief in the inherent morality of the group, stereotypes of out-groups, direct pressure on dissenters, self-censorship, illusion of unanimity and self-appointed mindguards. These result in nine recommendations, which are assign critical evaluator, make key member impartial, establish multiple groups, discuss the group's ideas with trusted people outside of the group, invite external experts, assign devil's advocate, devote a block time to discuss conflict and provide alternative scenarios, split the group and provide a second-chance meeting [Mul+94].

Groupthink was initially investigated in the political area [TH90] and military contexts [Kra98] as a causal factor for a defective decisions during a military engagement, such as the "Bay of Pigs" [Kra98], or a political event, such as the "Watergate Event" [Rav98]. The recent research covers case studies, experiments, literature reviews, applications and modifications of groupthink. The application areas are extended to jury decision-making or ice hockey team performance [Ros11].

In modern software engineering, practitioners have noticed groupthink [Bro14], such as the phenomenon about "dream team" (illusion of invulnerability) and "managers have not taken employees' preferences into account" (illusion of unanimity) on groupthink when performing project planning and

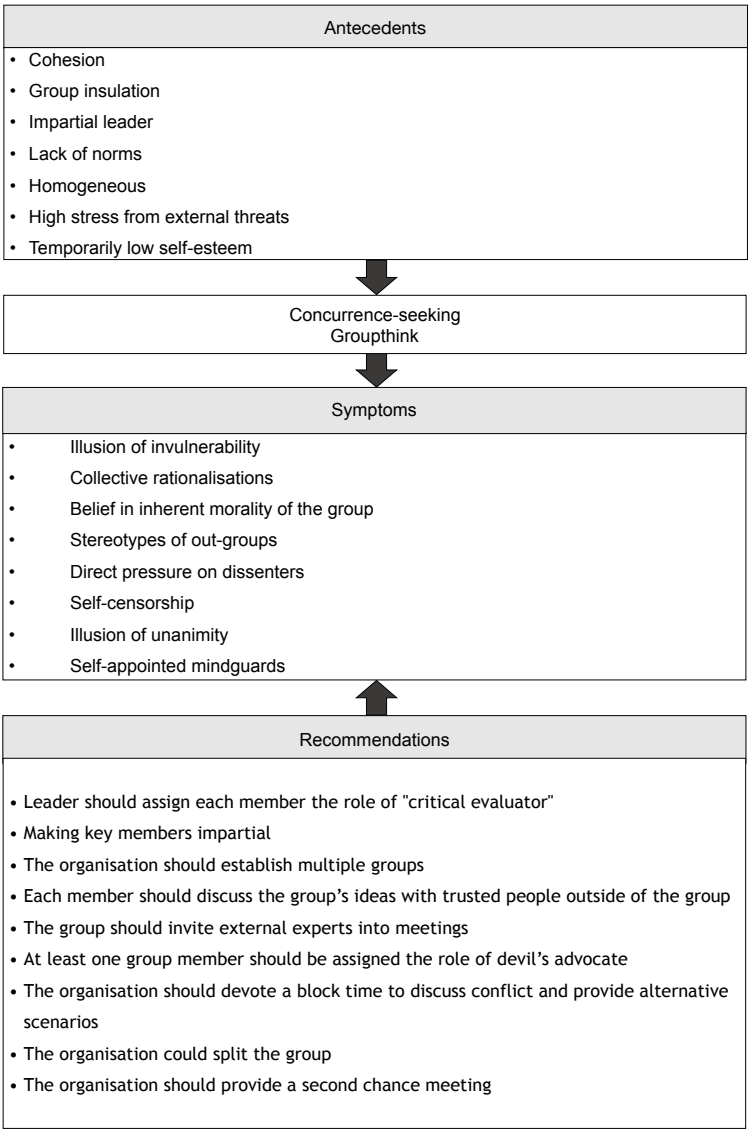


Figure 2.2: Groupthink Model [Jan08]

scheduling [DSVWJ11], as well as a negative impact on market success from "cohesion". The possible causalities of groupthink in software engineering include concepts such as "clan culture" or "political dominance" [Bro+10].

Chapter 3

State of the Art

In this chapter, we provide an overview of the closely related work to this dissertation, such as Safe Scrum, R-Scrum, IF-FMEA, model checking in ASD and existing communication challenges, groupthink problems in ASD for SCS.

3.1 Existing Scrum in SCS

There are several proposed ASD for developing SCS [Fit+13] [GPM10] [Vuo11]. Safe Scrum [SMH12] and R-Scrum are the two most popular scrum development processes for developing SCS.

Safe Scrum was proposed to make scrum a certifiable process in developing SCS. Safe Scrum is based on the original IEC 61508-3:2010 and expanded to IEC 60880 [SKM13], EN50128 [MSL15] and DO-178 B/C [HWS17]. It considers change impact analysis [Stå+14], quality assurance [Han+16] and documentation [Myk+14]. Safe Scrum puts all the safety analyses at the system-level outside the iterations including specifying the SIL. Before an iteration, the system description and concept, overall scope definitions, hazard and risk analysis and overall safety requirements (SSRS) phases 1-4, which are required by IEC 61508, are performed to derive safety requirements at the system-level. After an iteration, the final validation is performed as Reliability, Availability, Maintainability, Safety (RAMS) validation. RAMS validation is also performed at the end of the whole project. During an iteration, a safety product backlog is proposed for keeping traceability and maintainability. It solves the interdependencies between functional requirements and safety requirements. Test-driven development (TDD) is recommended to motivate developers considering design before implementation. In addition, TDD provides low-level documents to satisfy the requirements from norms. Safe Scrum uses a role named safety expert to take responsibilities for safety issues. The aforementioned introduction pertains to the basic Safe Scrum. The Safe Scrum is still under development [MS18].

R-Scrum is a scaled scrum development process for developing SCS in regulated environments, such as the automotive and the aviation [Fit+13]. R-Scrum summarises the core characteristics of complying ASD with regulated development processes, which are quality assurance (QA), safety and security, effectiveness, traceability, verification and validation. The authors conducted an industry case study in a real project called Quality and Compliance Management Software Solutions for Life Sciences (QUMAS) and compared the core characteristics with their own experiences. For instance,

"safety and security are system-level characteristics, and as such must be built-in from the start and not considered after the fact." In QUMAS, they used continuous compliance, and considered regulations at the beginning as well as performing an audit at the end to avoid risks. However, R-Scrum is an individual case based on a practical experience. The challenges are worth noticing, yet the solutions have not been evaluated empirically, nor have they been generalised.

To compare this dissertation with Safe Scrum, we propose the use of *systems theory*-based safety assurance methods in ASD. We fill the gap of Safe Scrum about a lack of integrated safety analysis and verification to face the problem concerning the changing architecture. To compare this dissertation with R-Scrum, we propose a general scrum development process with individual methods rather than a specific case, together with empirical evaluations, to face the challenges in R-Scrum.

3.2 Safety Analysis in ASD

To the best of our knowledge, the only research in performing safety analysis in ASD is the input-focused FMEA (IF-FMEA) [SM16a].

In 2016, Stålhane and Myklebust proposed IF-FMEA to handle the frequently changing requirements in ASD when performing safety analysis. Compared to a standard FMEA, IF-FMEA includes the inputs to the component under analysis as a list of inputs field in the table. By using these inputs, the changing parts of systems from user stories become clear. The other informations in the IF-FMEA table include component, user story ID, result, failure mode, input deviation, component failure, suggested barriers and new requirement.

The safety analyst performs IF-FMEA when there is a user story to be inserted or reinserted into the product backlog. IF-FMEA is started by checking the generic hazards list to see whether a new hazard is introduced or the existing hazards are affected. The new requirements from IF-FMEA might be new user stories.

IF-FMEA, as an extension of FMEA, is considered easy to learn in practice and to be able to handle changing requirements. However, there is still a need for a detailed architecture to start the safety analysis. In addition, IF-FMEA is based on reliability theory. Therefore, from our viewpoint, STPA, which is based on systems theory, is more suitable to perform safety analysis in ASD in today's sophisticated SCS.

3.3 Safety Verification in ASD

In most industries, safety verification in ASD is performed by testing [AD17], either mixed with functional testing or as UAT, which causes the safety verification to always be delayed, together with other challenges, such as methods' appropriateness, establishment and effectiveness.

Recent research proposes using formal methods in ASD. Shafig and Minhans [SM14] proposed integrating formal verification from requirements specification. Ghezzi et al. [Ghe+13] suggested an agile verification environment "AGAVE" that enables developers to use model checking. However, formal methods are not suitable in ASD due to the need for a model from source code, which does not exist. In addition, formal verification is executed in a kind of blackbox mode, which causes difficulties in communication among stakeholders [MG17].

TDD and ATDD are two agile testing methods proposed for safety verification [RR08]. TDD is used to test technical-level requirements, while ATDD happens at the end of each iteration by inviting experts to test high-level requirements. Stålhane and Myklebust combined TDD with IF-FMEA [SM16a]. They start the IF-FMEA as soon as they have selected a user story and decided which components to develop. The inputs and outputs are identified based on the relevant stubs, fakes or mocks. The IF-FMEA table can then be used both for safety analysis, as well as identifying new test cases. TDD fits well with IF-FMEA, since they consider safety more on single component's or function's failures. However, from our viewpoint, either safety analysis or safety verification should consider systems theory. Safety requirements

are derived for constraining system behaviours. Safety verification should include verifying system behaviours between TDD and ATDD.

3.4 Safety Documentation in ASD

The safety documentation in ASD shows challenges in a lot of articles [SH11] [Sel09] [Voi+16]. Paige et al. [Pai+11] found that a lack of explicit documentation limits agile practices' applicability to High-Integrity Systems (HIS) development. Heeager and Nielsen [NH17] explored their ASD project at a Danish pharmaceutical company. Their experience showed that most of the activities need to be recorded but the team members pay little attention, which causes a big mess in the project. Vuori [Vuo11] pointed out in his technical report that safety documentation in ASD should consider communication and relevant analysis.

In 2012, a safety product backlog was proposed in Safe Scrum [SMH12]. In 2016, Stålhane and Myklebust proposed other three safety-related documents in a scrum development process, namely an agile safety plan [MSL16], safety epic pattern [MS16], and safety story pattern [MS16], which have been suggested with reference to the norm EN 50126 [Sta06], respectively.

Safety epics illustrate high-level safety requirements: *To satisfy <the overall safety needs> the system must <always be able to reach a safe state>*. Safety stories record safety requirements, which are either from norm requirements or the safety analysis: *To keep <function> safe, the system must <achieve or avoid something>*. They suggest using an agile safety plan in achieving the certification. The agile safety plan follows requirements from EN 50126 clause 6.2.3.4 and E.1 in EN 50129.

However, the safety documentation should consider the relevant safety analysis techniques. These three documents are primarily used for FMEA. In this dissertation, we use STPA. An adaption is necessary.

In 2017, Jane Cleland-Huang [CH17] proposed five safety story formats depending on various behaviours:

1. Ubiquitous: *The <component name> shall <response>*;

2. Event driven: *When <trigger>, the <system name> shall <system response>*;
3. State driven: *While <in a specific state>, the <system name> shall <system response>*;
4. Optional: *Where <feature is included>, the <system name> shall <system response>*;
5. Unwanted: *If <optional preconditions> <trigger>, the <system name> shall <system response>*.

Since these five safety story formats were proposed after our evaluation, we leave this as future work to extend our safety documents in this dissertation with various behaviours.

3.5 Safety Communication in ASD

As far as we know, there has been little research concerning communication in ASD for developing SCS. Yet, some problems have been mentioned in recent articles.

Communication shows its importance in the second version (draft) of the norm ISO 26262 to be published in 2018, and in particular, when the safety issues encompass more information security issues, an effective communication between safety departments and information security departments in organisation has become increasingly important.

Leveson [Lev11] mentioned that the establishment of an appropriate communication and feedback channel is important when performing safety analysis. It should cover the knowledge sharing from organisation to operation or maintenance, as well as the implementation of hazards and safety requirements at the system-level or at the development-level.

Vilela et al. [Vil+17] conducted a systematic literature review with 57 papers to investigate the communication between requirements engineering and safety engineering (including safety analysis). The research questions concern the activities, techniques, information artifacts, tools and benefits of

performing safety analysis in connection with requirements engineering. 36 activities were listed by the authors. They mentioned that during these activities, no unified vocabulary among stakeholders hampers the communication. The techniques are illustrated by taxonomies, such as inductive/deductive and qualitative/quantitative. These taxonomies are considered to reduce the gap in communication between requirements engineering and safety engineering. To promote an effective communication, practitioners should create an agreed-upon vocabulary and semantic structure containing all the relevant concepts, their relations and axioms within the safety domain for the purpose of exchanging information and facilitating reasoning.

Thus, we agree with Vilela et al.'s opinions that it is worth noticing and establishing effective communication channels in safety analysis to reduce errors in requirements specification, improve system safety, help design, reduce costs and time, improve cooperation, enhance traceability, better present safety information, reduce workload, reduce interface faults, increase confidence and allow user feedback.

Pikkarainen et al. [Pik+08] conducted a case study in F-Secure with two agile software development projects. The authors pointed out that there are challenges in communication among team members and between team members and customer, management, support group, enterprise staff. The existing challenges include a lack of coordination [MPB16] and a non-deterministic decision-making process [DGCA17].

Hummel, Rosenkranz and Holten [HRH13] conducted a systematic literature review with 333 relevant papers on agile software development and communication. The authors noted that the project domain poses specific issues on communication channels, such as safety or security-critical systems which rely on documentation, yet the changing requirements are not well documented. Moreover, when developing complex systems, there are lapses of memory which negatively influence communication.

Communication occurs more frequently when using ASD for developing SCS. Especially during safety analysis and verification, the challenges in communication have a directly negative impact on the effectiveness of execution and quality of results.

3.6 Groupthink in ASD for SCS

Similar to the research on groupthink in ASD for SCS is scarce. It appears in accident reports, such as "Columbia Space Shuttle Disaster" [FC03] as well as in norms, such as in ISO 26262, "*groupthink*" is an indicator of a poor safety culture. Yet, the influence of groupthink has not been investigated in safety engineering, especially during safety analysis and verification.

Ferraris and Carveth [FC03] explained how groupthink led to a faulty decision-making, and further caused the "Columbia Space Shuttle Disaster". Due to a common organisational change, more complex communication issues arise from trying to keep to an on-time delivery. Some groupthink symptoms appeared in a real Columbia investigation, such as "some people are reluctant to raise certain issues of importance". The problems are clear, yet the solutions have not been raised.

Schiano and Weiss [SW06] focused on groupthink in security-critical systems. The solutions include "the norm security must be integrated into meetings". As security-critical systems show similarities with SCS in terms of organisation management, the problems can be considered, such as "not following norms" and "just doing what the business asked for".

Gren, Torkar and Feldt [GTF17] conducted a qualitative and quantitative investigation of eight large companies through surveys and interviews concerning group development and maturity when building agile teams. They pointed out the importance of considering group psychology in ASD, such as the increase in job satisfaction and personality. Groupthink has been found as a negative aspect for group maturity.

Coyle, Conboy and Acton [CCA13] mentioned that groupthink is prominently a significant counter-arguments. Groupthink causes group process losses in agile software development. Other findings, such as "group member domination" and "members afraid to speak up" are also related to groupthink.

McAvy and Butler [MB09] collected a number of groupthink symptoms during a longitudinal case study over two agile teams. Due to the cohesive agile teams, groupthink has a negative impact on decision-making. The authors suggested that the project manager should take a devil role to

reduce this phenomenon.

In conclusion, researchers in SCS as well as in ASD have both noticed groupthink. In SCS, groupthink causes disasters, while in ASD, groupthink is a normal phenomenon that causes faulty decision-making. When using ASD to develop SCS, especially to execute safety analysis and verification in this dissertation, groupthink occurs more frequently and becomes dangerous.

Chapter 4

A Preliminary S-Scrum

*In this chapter, we begin this dissertation by integrating a system-theoretic process analysis (STPA) in a scrum development process for facing the problem of lacking an integrated safety analysis during each iteration due to a changing architecture. We name it a preliminary "**S-Scrum**".*

The main contributions of this chapter are:

- **We propose a preliminary S-Scrum.** *We integrate STPA into an existing scrum development process, Safe Scrum.*
- **We illustrate the preliminary S-Scrum by using an example - Airbag System.** *The illustration shows a detailed execution of STPA in the preliminary S-Scrum with simple examples of system safety goals, accidents, hazards and safety requirements.*
- **We validate and explore the preliminary S-Scrum in a one-year student project.** *We ran the preliminary S-Scrum in a one-year student project with 14 participants. The results demonstrate that, to some extent, the preliminary S-Scrum can ensure safety, while agility is slightly reduced. Challenges exist. We propose further initial solutions. After that, both safety and agility have been enhanced.*

4.1 Concept

4.1.1 Safety-Guided Design

The major part of the preliminary S-Scrum is to integrate STPA in iterations to perform safety-guided design.

As we can see in Figure 4.1, Leveson describes: "*Most of the time, hazard analysis is done after the major design decisions have been made. But STPA can be used in a proactive way to help guide the design and system development, rather than as simply a hazard analysis technique on an existing design.*"

At the beginning of a project, hazards and system-level safety requirements are identified either from safety norms or from customers. A high-level architecture exists between the development team and the customers. Then, the safety-guided design can start. During an iteration, a safety expert together with the development team eliminate hazards or potential hazards from the high-level architecture. They create a system control structure and assign responsibilities for enforcing safety requirements. The safety expert refines the requirements by: (1) Identifying potentially hazardous control actions that would violate system requirements; (2) Determining the factors which could lead to a violation of safety requirements; (3) Augmenting and providing information and basic design decisions to the development team to eliminate or control potentially UCAs. During the iteration, the safety expert communicates with the development team and decides if the existing architecture needs to perform STPA again.

4.1.2 The Preliminary S-Scrum Process Model

As we can see in Figure 4.2, we extend Safe Scrum and propose the preliminary S-Scrum in three aspects: (1) During each sprint we integrate STPA as safety-guided design; (2) At the end of each sprint, we use STPA on the product instead of an RAMS validation; (3) We replace the final RAMS validation with STPA. The other safety analysis, such as in SSRS phase 1-4, is also possible to use STPA. The other parts which are kept consistent to Safe Scrum are: (1) The environment description and the SSRS phases 1-4;

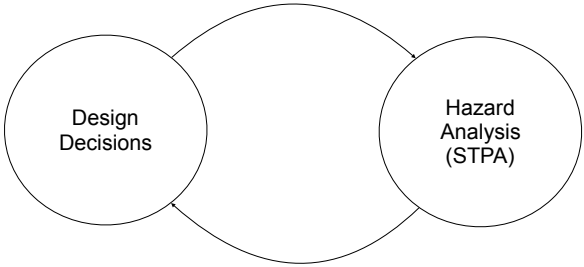


Figure 4.1: Safety-Guided Design [Lev11]

(2) TDD; (3) Safety product backlog; (4) A safety expert.

In STPA, the accidents are regarded as resulting from inadequate control. Thus, a control structure is generated from the architecture, which is considered as the crosspoint between STPA and Safe Scrum. This control structure can be reused in the following STPA safety analysis.

Pre-steps We start STPA from a general description of the environment and initial systems through SSRS phases 1-4. Approximate safety goals and requirements are determined in the sprint planning meeting. The development team and the safety expert generate a high-level system architecture based on the system requirements.

STPA step 1 After the sprint planning meeting, the safety expert begins to perform STPA step 1 including investigating the control actions with four unsafe modes and setting constraints on the unsafe modes. These constraints are translated as safety requirements for the following architecture design.

STPA step 2 The safety expert begins to perform STPA step 2. The causal factors that could lead to violate the safety constraints are determined through analysing the process variables and algorithms in the process model of the control system. More detailed safety requirements are to be elicited depending on the stepwise system design. These safety requirements from

STPA step 1 and step 2 are provided to the development team and implemented into the system under design and development by using TDD. When the new architecture of code is formulated by the development team, they can inquire the safety expert to perform STPA step 1 and step 2 again. The new architecture spans from an easy improvement of a single function module to a whole restructured system architecture.

Post-steps After each sprint, a deliverable product is prepared. We finally perform STPA step 1 and step 2 on it for the reasons: (1) Getting a final safety assessment; (2) Combining with safety verification at the system-level; (3) Driving the next sprint development. All the safety analysis activities are performed by a safety expert and the results are documented in the safety product backlog.

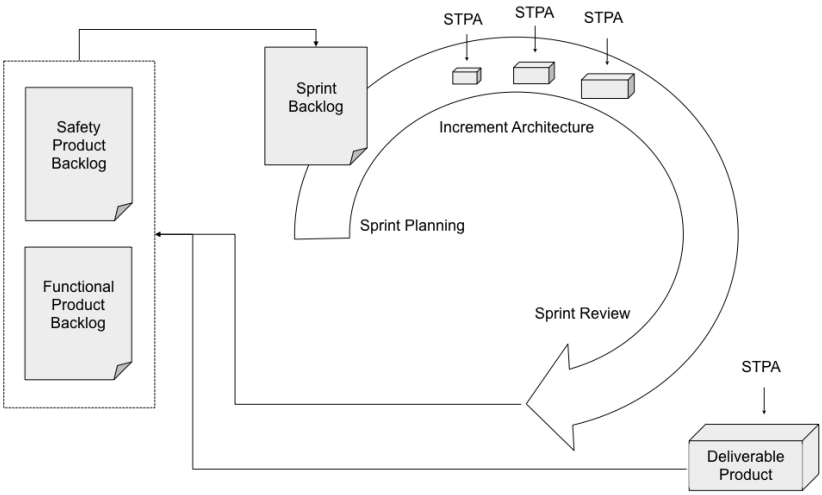


Figure 4.2: The Preliminary S-Scrum

4.2 Example

We demonstrate the preliminary S-Scrum through an illustrative example - Airbag System.

4.2.1 System Overview

Airbag system is equipped as one of the SCS in modern cars to protect the occupants from fatal injuries [Alj+09]. According to the ISO 26262, a new airbag system has to comply with ASIL D for unintended deployment of the airbag. An airbag system can be divided into three major parts: Sensors; Crash evaluation; Actuators. Once an impact happens, it would be detected by acceleration sensors and pressure sensors. Rollover accidents are typically detected by roll rate sensors. Through the sensor information, micro-controllers decide whether the sensed acceleration corresponds to a crash situation or not. The deployment of the airbags is activated if there was indeed a critical crash. Using airbags can protect the passengers from critical injury.

4.2.2 A Preliminary S-Scrum in Airbag System

Pre-steps We analyse the airbag system at the system-level.
System safety goal: During a critical crash, the airbag system should protect the passengers from being injured.
Accident: The occupants in the target vehicle are injured when a traffic accident occurred.
Hazard 1: The airbag is not ignited even though a critical crash occurred.
Hazard 2: The airbag is deployed unintentionally, which means that it is ignited even though no crash at all or only a non-critical crash has occurred.
Hazard 3: The airbag is ignited after T = 45ms.
The highest-level safety requirements transform directly from the identified hazard for the system.
System safety requirements 1: If a critical crash occurred, the airbag must be ignited.

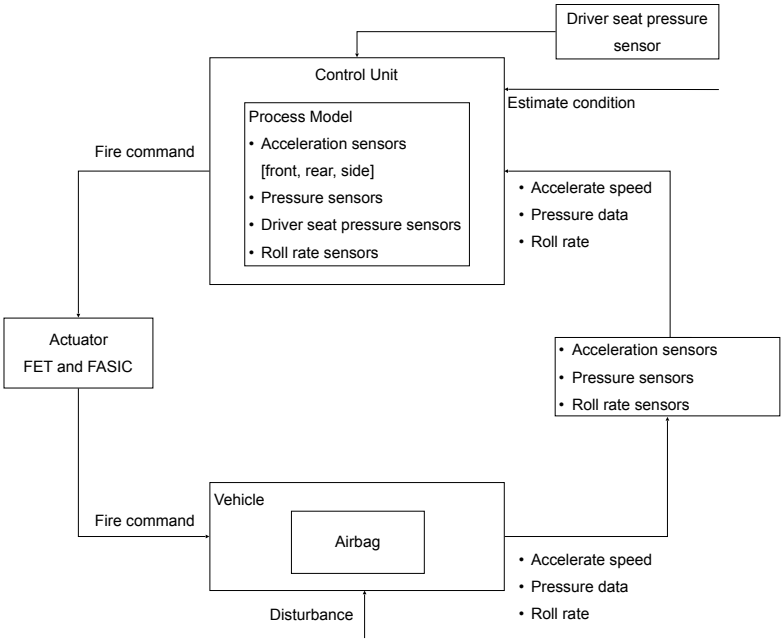


Figure 4.3: Airbag System Control Structure

System safety requirements 2: If there was no crash or only a non-critical crash, the airbag must not be ignited.
System safety requirements 3: If a critical crash occurred, the airbag must be ignited before T = 45ms.

The hazard and related safety requirements must be recorded in the safety product backlog in the first sprint planning meeting and taken as reference throughout the whole development process.

STPA step 1 We draw a control structure in Figure 4.3. The control structure depicts not only the components at a high-level of airbag system, but also the main interconnections. There should be two micro-controllers for decreasing the hazard of unintended airbag deployment. Due to the

same functions (one of them is for redundancy), we integrate them into one micro-controller in the control structure diagram. The deployment (actor) of the airbag is secured by two protection mechanisms, the Field Effect Transistor (FET) controls the power supply. The Firing Application Specific Integrated Circuit (FASIC) controls the airbag squib. Only if the control unit receive the signals from the sensors and FET has enough electrical power, the FASIC will ignite the airbag squib. The sensors we considered in the architecture are: (1) Two acceleration sensors and two pressure sensors to detect front or rear crashes (x direction and -x direction acceleration); (2) Driver seat pressure sensor to detect the present of occupants; (3) Angular rate or roll rate sensors to detect rollover accidents. The input of this step is a certain amount software code with architecture during sprint. The output is a control structure diagram with a process model.

As shown in Table 4.1, we identify each control action into four general hazardous types, which is shown in Section 2.2.2. In Table 4.2, we formulate the safety requirements from UCAs. The input of this step is the control structure diagram with control actions. The output is the initial safety requirements based on UCAs.

Table 4.1: Airbag System - UCAs

Not provided	Provided	Too soon or late	Too long or short
UCA.1: Not sending fire command is hazardous if there was a critical crash.	UCA.2: A fire command is needlessly sent to FET and FASIC, thus causing an unintended deployment of the airbag.	UCA.3: The fire command for the airbag in case of a crash is delayed, thus causing the airbag to be ignited too late.	/

STPA step 2 As shown in Table 4.3, we identity each UCA with the causal factors, which could violate the safety requirements of each UCA from STPA step 1.

In the micro-controller process model of the airbag system, we formulate four types of interface variables that can affect the safety of these control

Table 4.2: Airbag System - Safety Requirements

1. The airbag system control unit shoud provide fire command when there was a critical crash. 2. The airbag system control unit must not send fire command when there was no crash or non-critical crash. 3. The airbag system control unit must send fire command T <= 45ms, when there was a critical crash.

actions: (1) Acceleration speed; (2) Pressure intensity; (3) Driver seat pressure intensity; (4) Roll rate. We determine the status of each variable, which could lead to the UCA. From the control unit viewpoint, the causal factors might be: (1) Airbag system control unit does not detect the driver is present, thus the airbag system control unit does not been enable. (2) Acceleration threshold is incorrect and allows the vehicle get an abnormal acceleration without sending a signal. From the sensor viewpoint, the causal factors might be: (1) Pressure sensor is unable to detect pressure due to road and weather conditions. (2) Noise is not adequately filtered and a rapidly acceleration is not real-time detected. The **post-step** is to be performed after several iterations of STPA in a sprint and to be happened at the end of the project among all the stakeholders.

Table 4.3: STPA Step 2 - Causal Factors for UCA.1

Process Model ABS Enabled: [Y, N] Driver Present: [Y, N] Acceleration Speed >= T: [Y, N] Roll Rate >= T: [Y, N]	1. Airbag system control unit does not detect the driver is present, thus the airbag system control unit does not been enable. 2. Acceleration threshold is incorrect and allows the vehicle get an abnormal acceleration without sending a signal.
Sensor Acceleration Sensors Pressure Sensors Driver Seat Pressure sensor Roll Rate Sensors	1. Pressure sensor is unable to detect pressure due to road and weather conditions 2. Noise is not adequately filtered and a rapidly acceleration is not real time detected.

4.3 Evaluation

In this section, we aim to evaluate the preliminary S-Scrum. We validate the agility and safety of the preliminary S-Scrum as well as explore the challenges and their relevant optimisations in a one-year student project called "Smart Home". We conduct this study following the guideline by Runeson [RH09] and Yin [Yin13]. We design this case study with a multi-staged procedure. Each stage has different objectives and research questions. We explore challenges and optimisations in the *preliminary S-Scrum* in stage 1, while we validate the *optimised preliminary S-Scrum* in stage 2.

4.3.1 Context

The case study, including stage 1 and stage 2, was performed in a project developing the SCS, "Smart Home", between March, 2016 and March, 2017 at the Institute of Software Technology, University of Stuttgart. The project had 400 planned working hours per head with a headcount of 14 students. The students have taken part in a training program for ASD and STPA before joining the project and a course on automation systems during the project. The scrum master was one research assistant with experienced project management background, while the product owner and safety expert was another research assistant majoring in using ASD for SCS. All the students were supervised by three research assistants.

As we can see in Figure 4.4, the project was to work on an IoT based smart home with a smart coffee machine, smart light alarm system, autonomous parking system, door open system, and smoke detector alarm system through the IoT server - KAA[1]. The smart home contains individual SCS, such as autonomous parking system and smoke detector alarm system, while an non-intentional interaction among these individual systems will also cause hazards. For example, when the door is forced open during the night, but the light alarm system does not work. That causes dangerous for residents.

[1] https://www.kaaproject.org/overview/

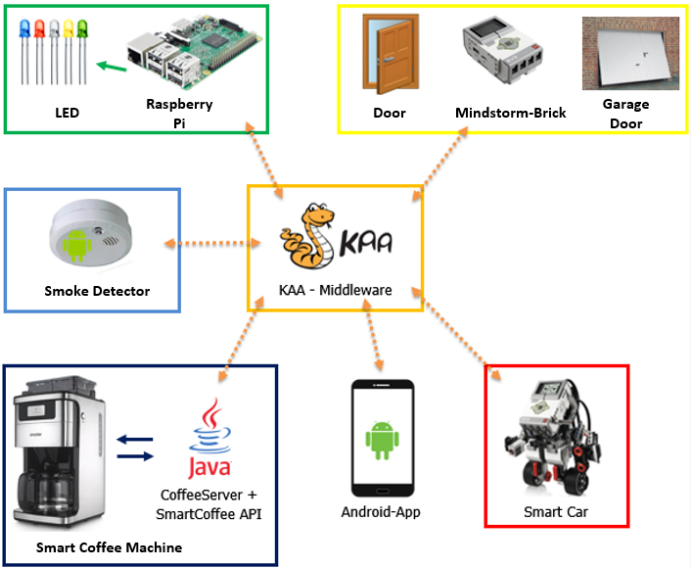


Figure 4.4: "Smart Home" Project

We open the project "Smart Home" in GitHub[1].

4.3.2 Research Question

We formulate four research questions to steer the design of our study, as shown in Table 4.4.

Table 4.4: Research Questions

RQ 1	How does the preliminary S-Scrum handle agility and safety in SCS?
RQ 2	What are the challenges of the preliminary S-Scrum in such a context?
RQ 3	How could the preliminary S-Scrum be optimised to overcome the challenges?
RQ 4	What are the effects of the optimised preliminary S-Scrum on safety and agility?

[1] https://github.com/ISTE/student-project—Smart-Home

4.3.3 Case Study 1

The objective of stage 1 is to validate the safety and agility of the preliminary S-Scrum and optimise it. In stage 1, we focus on answering RQ 1, RQ 2, and RQ 3. The general research strategy in stage 1 is shown in Table 4.5.

Table 4.5: Research Strategy in Stage 1

Time	Sprint 1 to sprint 5	Sprint 6 to sprint 7	Sprint 8	Sprint 9
Process	Scrum	Preliminary S-Scrum	Preliminary S-Scrum	Preliminary S-Scrum
Data collection	Participant observation; Scrum artifacts; Documentation review.	Participant observation; Scrum artifacts; Documentation review.	Questionnaires	Semi-structured interviews
Participants	Developers Stakeholders	Developers Stakeholders	13 Developers	5 Developers 1 Scrum master
Data types	Quantitative	Quantitative	Quantitative	Qualitative
Analysis	Sum	Sum	Median MAD	Coding
Output	No safety data	Safety data: M16.1-M16.3 M17.1-M17.3	Agility data: M1-M15	Challenges and optimisations

4.3.3.1 Data Collection 1

Stage 1 spans from sprint 1 to sprint 9. Each sprint lasts three weeks. The agility related quantitative data, M1 to M15 (as illustrated in 4.3.3.2), were collected through 13 questionnaires[1]. We conducted participant observations as product owner, scrum master, and customer. The safety related data, M16.1 to M16.3 and M17.1 to M17.3 (as illustrated in 4.3.3.2), were quantitatively collected during sprint 6 and sprint 7. From sprint 1 to sprint 5, we executed normal scrum without safety analysis for the adaptation and preparation for the project. The STPA was performed by the safety expert

[1]The questionnaire is available: https://zenodo.org/record/439696#.WODCovl96Uk

and recorded privately by using the STPA tool, XSTAMPP[1], while the hazards and safety requirements were recorded in the safety product backlog in Jira.

Based on the quantitative data for agility and safety, we then designed semi-structured interviews with 6 voluntary participants from the development team, including the scrum master and five developers. The interviews lasted 270 minutes overall. The questions began with a specific set of questions regarding the observations. Further, we asked about the causalities. Finally, the optimisations were collected in an open-ended mode. The interview guideline[2] was provided before each interview. We recorded interview data in field notes and we used audio recordings for text transcription.

4.3.3.2 Data Analysis 1

We analysed the data using the combination of two classic approaches to generate software metrics, which are Goal Structuring Notation (GSN) [KW04] and Goal Question Metric (GQM) [Bas92] referring partially to an SCS evaluation framework - VMF [CMS09], as shown in Figure 4.5. The data are from two aspects: agility (S1) and safety (S2). To evaluate and optimise agility (S1), we set 15 goals (G1 to G15) considering Comparative Agility Survey [WRC10]. They are: G1 (Team work composition); G2 (Team work management); G3 (Communication); G4 (Requirement emergency); G5 (Technical design); G6 (Planning levels); G7 (Critical variables); G8 (Progress tracking); G9 (Sources of dates and estimates); G10 (When do we plan); G11 (Customer acceptance test); G12 (Timing); G13 (Quality focus); G14 (Reflection); G15 (Outcome measure). To reach G1 to G15, we analysed M1 to M15 indirectly by setting sub-metrics. For example, M1 (Team work composition) was analysed by M1.1 (Team members are kept as long as possible), M1.2 (Specialists are willing to work outside their specialty to achieve team goals), M1.3 (Everyone required to go from requirements to finished system is on the team), and M1.4 (People are no more than two teams). Each sub-metric was analysed on an ordinal scale of 5 (e.g., from 1

[1] http://www.xstampp.de/

[2] The interview guideline is available: https://zenodo.org/record/439696#.WODCovl96Uk

to 5 means "Negative", "More negative than positive", "Neither negative nor positive", "More positive than negative", and "Positive"). To investigate the in-depth challenges, we selected either the negative values of the results or the significant differences between the normal scrum and the preliminary S-Scrum to formulate further interview questions. To analyse the interview results, we used NVivo11 for text encoding [SC97]. We start with open coding to record the answers line by line. Then, we use selective coding to derive the relevant answers. Finally, we use axial coding to relate the challenges with their possible solutions. Concerning safety, G16 is extended with 3 questions together with 3 metrics including: Number of software hazards (M16.1); Number of software safety requirements (M16.2); Number of safety requirements traceable to hazards (M16.3). G17 is extended to be evaluated by: Number of mitigated hazards (M17.1); Number of accepted safety requirements (M17.2) in the present sprint; Number of rejected safety requirements (M17.3) in the project.

4.3.4 Results 1

4.3.4.1 RQ 1: How does the preliminary S-Scrum handle agility and safety in safety-critical systems?

We investigate the effect on agility by comparing the normal scrum and the preliminary S-Scrum according to the 15 metrics in Figure 4.6. From the general overview, we can conclude that most of the values regarding agility in preliminary S-Scrum are slightly worse than those in the normal scrum, while one metric shows strongly negative values ("when do we plan"). We discussed the results with the technical support from the Comparative Agility Survey and got the feedback: "*When most of the values are more positive than negative (more than 3), we could say that the process is agile enough.*" Moreover, most values show relatively small differences between normal scrum and preliminary S-Scrum. Thus, we consider the agility of the preliminary S-Scrum to be acceptable. Yet, optimisations are needed. Regarding the safety of the preliminary S-Scrum, we performed STPA two

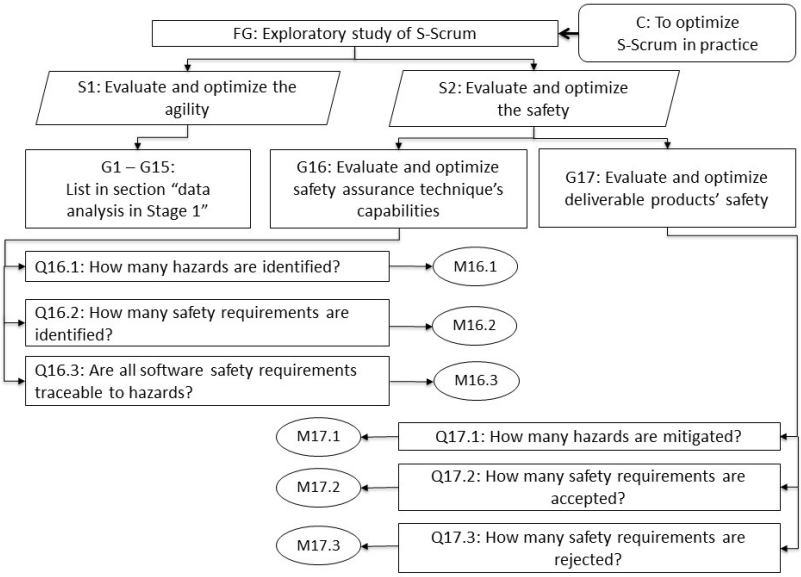


Figure 4.5: General Data Analysis Strategy

rounds in sprint 6. We found 6 software hazards (M16.1) and 15 safety requirements (M16.2), which can all be traced back to software hazards (M16.3). Three hazards were mitigated (M17.1), while 14 safety requirements were accepted (M17.2). In sprint 7, we performed two rounds of STPA analysis. We found 10 software hazards (M16.1) and 24 safety requirements (M16.2), which can also all be traced back to software hazards (M16.3). Six hazards were mitigated (M17.1), while 23 safety requirements were accepted (M17.2). Each sprint has 1 rejected safety requirement due to hardware limitation (M17.3).

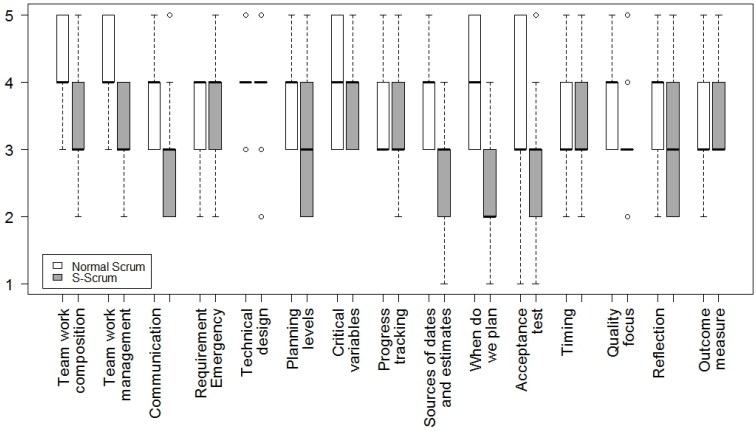


Figure 4.6: Boxplots for General Agility Comparison between Normal Scrum and Preliminary S-Scrum ("1" to "5" mean "Less Agile" to "More Agile")

4.3.4.2 RQ 2 & RQ 3: What are the challenges of the preliminary S-Scrum in such context? & How could the preliminary S-Scrum be optimised to overcome the challenges?

To optimise preliminary S-Scrum, we derived six challenges[1] from the six abnormal values (as shown in Section 4.3.3.2) from the sub-metrics inside these 15 metrics.

Challenge 1: The priority management of safety requirements and functional requirements has conflict. In the normal scrum, the management and development team determine the sprint backlog with functional requirements in the sprint planning meeting. All the team members have a clear overview of and commitment to the sprint plan with relatively high-level features.

[1] In the following chapters, we summarise these six challenges into four major challenges: Requirements prioritisation; Communication; Planning; Verification.

The developers accomplish each item with their own detailed tasks. The requirements from the management and the concrete realisations from the developer reach a consensus during each sprint. In preliminary S-Scrum, the integrated STPA and the safety requirements break the balance. The functional requirements are correlated with the safety requirements. However, some developers thought: "*Functional requirements are more important than safety requirements.*" It was found that the need for long-term quality was given a lower priority than the need for short-term progress [MAD12]. Moreover, the safety expert spent a relatively short time working with the team members which influences also the decision making. As one developer mentioned: "*The safety expert is not working in the same room with the development team and has an inconsistent working time.*" Thus, a lack of an in-time decision maker on the safety requirements together with the ignorance of safety requirements in the development team cause the conflict.
To face this challenge, a **safety culture** should be integrated into a lightweight development process. We suggest to include an **internal safety expert** in the development team to: (1) Spread the safety culture; (2) Increase the safety expert's working time with the team members; (3) Clarify the bewilded safety requirements. An **external safety expert** is necessary to communicate with other stakeholders. To fill the gap between the external safety expert and the development team, the development team suggests that the external safety expert should join at least once the weekly scrum meeting. The discussion between the management, the external safety expert and the internal safety expert could strive a fresh balance on the priorities.

Challenge 2: The communication between team members and safety expert is disturbed. To start with, the unclear safety-related documentation influences an effective communication. The team members mentioned: "*It is difficult to comprehend the purpose of the safety expert and integrate into our daily work from the existing documents.*" Moreover, a lack of safety-related knowledge of the development team influences the discussion concerning safety issues. Finally, the insufficient time spent between safety expert and development

team causes also a poor communication. Without a non-obstacle work place to communicate within the team about the work progress, the safety assurance could either be a superficial decoration or even worse, a roadblock during fast product delivery.
To face this challenge, in addition to the separated **internal safety expert** and **external safety expert**, a **weekly safety meeting** is suggested by an interviewee: "*The internal safety expert and external safety expert should meet each other at least once a week to exchange the status of the development team. Because the discussion should be deep in the safety area, it is not supposed to be established during the normal weekly scrum meeting.*" Last but not least, we improve our **safety epics** and **safety stories** to support an effective communication, as shown in Section 4.3.4.1.

Challenge 3: The safety requirements are not determined early enough to appropriately influence design and testing. In sprint 6 and sprint 7, the safety requirements were determined by the development team and the safety expert together in the sprint planning meeting. However, as one interviewee mentioned: "*The determination of safety requirements from the safety product backlog is too late to avoid a conflict between the functional requirements and their suitability for the coming sprint.*" Thus, sometimes the functional design and testing have to start without the in-time safety requirements.
To face this challenge, we propose **a pre-planning meeting** for solving the time pressure problem. First, the internal, external safety experts and product owner discuss the safety product backlog and the functional product backlog in the pre-planning meeting. Then they brainstorm the results with the whole development team in the sprint planning meeting to gather more ideas and make each safety requirement clear.

Challenge 4: The planning at the start of each iteration is insufficient. In the normal scrum, the development team and the product owner plan the upcoming sprint in the sprint planning meeting by formulating the sprint backlog with estimated items, which makes the development team sufficiently confident about their plan. However, the estimation and planning for

the safety product backlog seem not ideal, as well as the interconnection with the functional product backlog, which make an in-time identification of the sprint backlog difficult. An interviewee said: "*It is difficult to determine the safety requirements when the development team has not planned the functional requirements for the coming sprint.*"
To face this challenge, we suggest and adapt an **agile safety plan** [MSL16] to perform planning of safety activities iteratively, in connection with the **pre-planning meeting** to increase the understanding of safety issues and enhance confidence. In our project, the results of STPA are part of the agile safety plan.

Challenge 5: The time to perform upfront planning is late. A team member said: "*The pre-planning meeting for safety issues should start before the sprint planning meeting. But the concrete time should be decided between the external safety expert, the internal safety expert and the product owner.*" Based on the experience of the previous sprints, it is better to start upfront planning one week before the sprint planning meeting (3 weeks/sprint). The time could be changed depending on the sprint length. More explanations are in challenge 4.

Challenge 6: The safety requirements lack well-defined completion criteria. In the normal scrum, we have various testing methods to determine the completion of each feature such as unit testing, system testing, regression testing, and acceptance testing, which are promoted to be automated in an agile context. However, few agile testing methods are suitable for validating safety requirements, as the safety requirements are either from norm requirements or the safety analysis, which differentiates safety testing and functional testing. In preliminary S-Scrum, we use UAT (User Acceptance Testing) for validating safety requirements. Thus, a suitable safety criterion becomes important.
To face this challenge, we use a "Given-When-Then" format [WH12] as **safety requirements' criteria**. The development team suggest that the external safety expert could decide the **safety stories' criteria** and the internal safety

expert could decide the **safety tasks' criteria**. The whole development team could **brainstorm** both criteria. To this end, the product owner and safety expert perform the acceptance testing.

4.3.5 Case Study 2

After the optimisations described above, the objective of stage 2 is to validate the safety and agility of the optimised preliminary S-Scrum and discuss it in industry. We focus on answering the RQ 4 together with some discussion from industry. The general research strategy in stage 2 is shown in Table 4.6.

Table 4.6: Research Strategy in Stage 2

Time	Sprint 10 - Sprint 11	Sprint 12	Sprint 13
Process	Optimised preliminary S-Scrum	Optimised preliminary S-Scrum	Optimised preliminary S-Scrum
Data collection	Participant observation; Scrum artifacts; Documentation review.	Questionnaires	Semi-structured interviews
Participants	Developers Stakeholders	8 Voluntary developers	1 PO (from EPLAN) 1 SM (from EPLAN)
Data types	Quantitative	Quantitative	Qualitative
Analysis	Sum	Median and MAD	Coding
Output	Safety data: M16.1-M16.3 M17.1-M17.3	Agility data: M1-M15	Preliminary discussion in industry

4.3.5.1 Optimised Preliminary S-Scrum

To have a clear overview, we compare the optimised preliminary S-Scrum to the normal scrum and the preliminary S-Scrum in our project respectively in Table 4.7. In the optimised preliminary S-Scrum, we differentiate between an internal safety expert and an external safety expert. A pre-planning meeting and weekly safety meetings are established between safety experts. We include the safety epics, to satisfy <the overall safety needs>, the system must <always be able to reach a safe state> [MS16], in the story map. The

safety product backlog is improved with optimised safety story: To keep <control action> safe, the system must <achieve or avoid something>. An agile safety plan based on STPA technology is suggested for a clear overview. The safety culture is expected to be enhanced by the additional activities.

Table 4.7: Normal Scrum, Preliminary S-Scrum and Optimised Preliminary S-Scrum in "Smart Home"

Normal Scrum	14 DLs 1 SM 1 PO	Sprint planning meeting Weekly scrum meeting (2 times/week) Sprint review meeting Sprint retrospective meeting	Story map PB
Preliminary S-Scrum	14 DLs 1 SM 1 PO 1 SE	Sprint planning meeting (with safety planning) Weekly scrum meeting (2 times/week) (with safety discussion) Sprint review meeting (with safety review) Sprint retrospective meeting	Story map Functional PB Safety PB
Optimised Preliminary S-Scrum	13 DLs 1 SM 1 PO 1 external SE 1 internal SE	Pre-planning meeting Sprint planning meeting (brainstorming requirements/criteria) Weekly scrum meeting (2 times/week) Weekly safety meeting (1 time/week) Sprint review meeting (with safety review) Sprint retrospective meeting	Story map (with safety epics) Functional PB Safety PB (with safety stories) Safety plan

4.3.5.2 Data Collection 2

Stage 2 is from sprint 10 to sprint 13. The safety related data, M16.1 to M16.3 and M17.1 to M17.3, were collected in the same way as in stage 1. The safety results were collected by both internal and external safety experts. The agility related data, M1 to M15, were collected by the second round

questionnaires[1]. We further discussed the optimised preliminary S-Scrum by conducting 2 semi-structured interviews with one scrum master and one product owner from EPLAN GmbH, Germany. The interview lasted 2 hours. We formulated questions about the status of the scrum development process in the company projects, the feasibility of the optimised preliminary S-Scrum in industry, and further suggestions from the industrial perspective. A project background illustration was provided before the interviews, together with the interview guidelines[2]. The field notes, interview transcripts, and voice recordings were all preserved for backup.

4.3.5.3 Data Analysis 2

The quantitative data were compared with the numbers in stage 1. The interview results from the industry were text encoded with: Status; Challenges; Possible solutions; Feasibility of preliminary S-Scrum.

4.3.6 Results 2

4.3.6.1 RQ 4: What are the effects of the optimised preliminary S-Scrum on safety and agility?

As shown in Figure 4.7, most of the evaluated agility aspects sustained a good level of satisfaction with little variance. However, the "technical design" is slightly reduced. Due to the new role, the collaborative part of design between safety work and development work fell on the internal safety expert. The personal capability is becoming important. To improve the technical design, cooperation should increase between the external safety expert and the development team. In addition, the team members are still confused about how to conduct design in ASD.

Regarding the safety of optimised preliminary S-Scrum, as we can see in Figure 4.8, safety aspects improved (M16.1, M16.2, M16.3, M17.1, M17.2). We also rejected few safety requirements (M17.3): 1 (sprint 6), 1 (sprint 7),

[1]The questionnaire is available: https://zenodo.org/record/439696#.WODCovl96Uk
[2]The interview guideline is available: https://zenodo.org/record/439696#.WODCovl96Uk

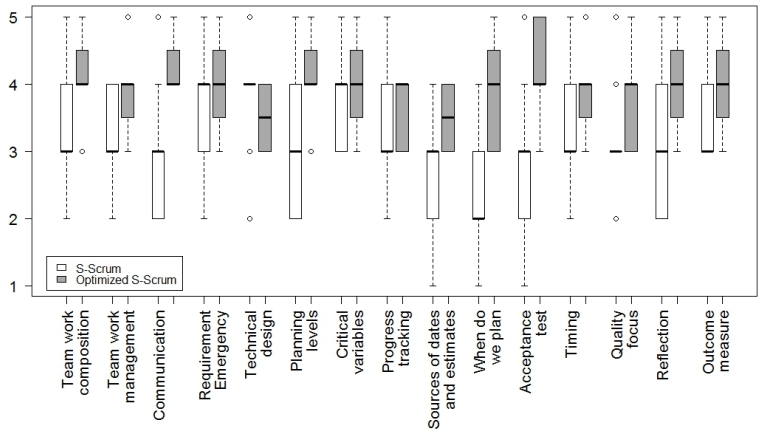


Figure 4.7: Boxplots for Agility Comparison between Preliminary S-Scrum and Optimised Preliminary S-Scrum ("1" to "5" means "Less Agile" to "More Agile")

0 (sprint 10), 2 (sprint 11). We can conclude that, in general, the optimised preliminary S-Scrum has better safety assurance capabilities. However, there are still some abnormal values in sprint 7. The number of safety requirements, the number of safety requirements traceable to hazards and the number of accepted safety requirements in sprint 7 are more than in sprint 10. This may be traced back to the fitting-in phase of the optimised preliminary S-Scrum. Since the training of STPA for the internal safety expert, we finished STPA in sprint 10 only once. In sprint 6, sprint 7, and sprint 11, we finished STPA twice. After the adaption of the new role, the safety data rose in sprint 11. Another reason of the rising data in sprint 11 might be the increasing complex system, which contains more possible hazards as well as safety requirements.

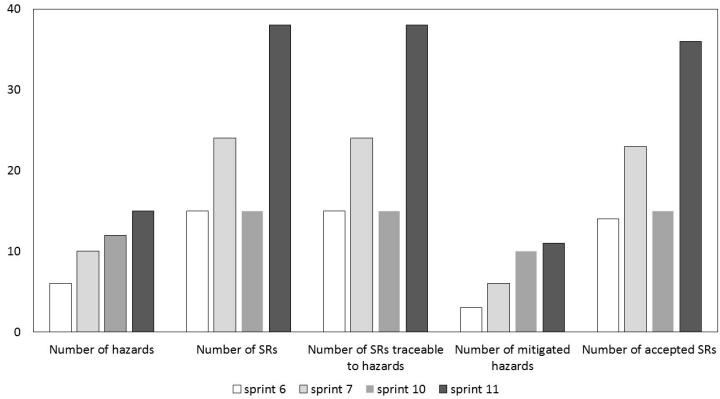


Figure 4.8: Safety Data Comparison between Preliminary S-Scrum and Optimised Preliminary S-Scrum

4.3.7 Discussion

To strength the study further, we discussed our results preliminarily in industry. For *Challenge 1*, the conflict between functional requirements and non-functional requirements seems not obvious. As one interviewee mentioned: "*Since we have a relative small amount of non-functional requirements, the priorities are always determined by the product owner together with the discussion with some external experts.*" For *Challenge 2*, one interviewee mentioned: "*To enhance the communication between the team members and the experts, we have a technical meeting before each sprint planning meeting. The product owner sends the emails to the relevant experts depending on the goals of each sprint. The experts are welcomed to join the daily stand-up meetings.*" Thus, the experts have sufficient time to keep up with the development team, while the technical knowledge is deeply discussed in the technical meeting before the sprint planning meeting. The project has also a good knowledge sharing mechanism to support the communication during each sprint. One interviewee mentioned: "*We use pair programming, formal guidelines to teach new colleagues, chat clients, and screen sharing. When the team includes experts, the product owner will contact 2-3 colleagues to discuss technical stuff,*

who will inform other colleagues." A hierarchical communication mode is preferred for a multi-expert team. For *Challenge 3*, the industrial projects have also mentioned this problem: "*Internal user stories are used to record the non-functional requirements. The execution of internal user stories is up to the team.*"

For *Challenge 4*, the two teams execute a sufficient planning. An interviewee mentioned: "*We have a refinement time slot to get all product backlog items approved (each team member has understood) and not so much discussion in the sprint planning meeting.*" The team members are beginning the refinement in the present sprint for the user stories in the next sprint. In scrum, not all requirements have to be at the same level of detail at the same time [Rub12]. The progressive refinement could be further extended for the safety planning and assessment to: (1) Avoid a premature development decision from the high-level safety requirements; (2) Reserve sufficient time for managing priorities between safety requirements and functional requirements; (3) Increase the rework possibilities; (4) Enhance the likelihood of using conversation to clarify safety requirements. That could also illustrate the *Challenge 5*. For *Challenge 6*, the refinement phase helps building a pre-understanding of each requirement and reaching a common criterion in the sprint planning meeting.

The *external expert* is a regular member in industry. An interviewee mentioned: "*We prefer some experts with deep knowledge in the team, but the arrangement of an internal expert has to take more issues into account, such as training, responsibility, and even personal development.*" An external safety consultant to test the products and delivered trainings and an internal safety initiative [Pol+17] to promote safety practices across groups in industry could be align with our internal and external safety expert. *Safety culture* in industry is enhanced either by setting the regulations or by the established organisational structure and activities. An *agile safety plan* is also required from some norms. They draw the safety plan either in the technical meeting or in parallel with the refinement. The technical meeting suggested in industry could also be considered as an *extra (weekly) safety meeting*. The *pre-planning meeting* seems to be a suitable form for realising progres-

sive refinement in industry. This alignment motivates more combinations between our optimisations and existing industrial practices. All the requirements and *acceptance criteria* are retrieved by *brainstorming*. An effective communication plays a vital role in executing acceptance testing.

4.3.8 Threats to Validity

4.3.8.1 Construct validity

The first threat to construct validity is the general data analysis framework. To apply scrum for SCS, we focus primarily on safety aspect and agility aspect in our exploratory study. In terms of agility, we referred to an official agility comparative survey [WRC10] for ensuring the coverage of measurement. In terms of safety, preliminary S-Scrum was extended from Safe Scrum, which was originally developed in accordance with the general functional safety norm IEC 61508. Thus, the validation regarding to the consistency with IEC 61508 has not been included in the framework. Furthermore, in preliminary S-Scrum we mainly integrate STPA. We aim to validate the enhanced safety concerning the integrated safety analysis technique. Thus, the safety assurance technique's capability and the deliverable products' safety are set as two relevant goals. Yet, the goals and metrics seem not enough and the validation framework is possible to be extended. The second threat to construct validity is the validation periods for preliminary S-Scrum and optimised preliminary S-Scrum are shorter than our expectations. We executed the normal scrum in the first five sprints to strengthen students' background knowledge of agile methods and prepare the detailed organisational structure , which took us a lot of time.

4.3.8.2 Internal validity

The first threat to internal validity is the arrangement of team roles. One of the authors acted as the product owner and the safety expert concurrently in sprint 6 and sprint 7. To avoid this threat in alignment with the optimisations in sprint 10 and sprint 11, the product owner acted further as an external

safety expert. An internal safety expert has been arranged in the development team. The second threat to internal validity exists in the qualitative data from the semi-structured interviews. The interviews have been performed by one of the authors together with the audio record. The language we used has also partial German. To avoid subjective and language bias, the audio recording has been transcribed independently by two researchers (one is a native German speaker) and compared to formulate a final result.

4.3.8.3 External validity

A student project is different from an industrial project. However, Höst et al. [HRW00], Tichy, Kitchenham et al. [Tic00] proposed that students could be acceptable. To consider this debatable issue, we mainly referred to an empirical study conducted by Falessi in 2017 [Fal+18a]. 16 statements are provided by 65 empirical researchers. They mentioned: "*Conducting experiments with professionals as a first step should not be encouraged unless high sample sizes are guaranteed or performing replicas is cheap.*" In our research, there exists few industrial projects for developing SCS fully adopted a scrum development process according to the preliminary research [The+15]. Preliminary S-Scrum was also proposed in 2016 as a high-level process model. In addition, the long learning cycles and a new technology are two hesitations for using professionals. STPA was developed in 2012. In industry, there is still a lack of experts. Thus, we believe that in our research area, a student project is a relative suitable way to aggregate contributions. Even though, the generalisability is considered critical when evaluation processes for SCS due to the various SIL.

4.3.8.4 Reliability

The student project is a suitable way for a first validation. Yet, the results from the students are limited by their personal experience. Besides, the "grading power" of the researchers may influence the results. We separated our research work from the final examination of the product to mitigate this

threat.

4.4 Conclusion

In this chapter, (1) we propose a preliminary S-Scrum by integrating STPA into Safe Scrum. To face a changing architecture in ASD, STPA seems to be a possible technique to be integrated into each iteration to perform safety-guided design. (2) The illustrative example, airbag system, provides us a clear picture and shows the feasibility of performing STPA iteratively. (3) We evaluate the preliminary S-Scrum concerning safety and agility in a student project "Smart Home". The measures are in a moderate-level with six challenges, which are safety requirements' acceptance criteria, priority management, communication, time pressure on determining safety requirements, safety planning. We propose further initial solutions include the split of the safety expert, pre-planning meeting, regular safety meeting, improved safety epics, STPA-based safety stories and an agile safety plan. Safety and agility of the preliminary S-Scrum are greatly improved. For further research, we summarise these six challenges into four major aspects in the following chapters: *Verification; Planning; Communication; Requirements prioritisation.*

Chapter 5

Safety Verification in S-Scrum

*In this chapter, to face two of the four challenges (verification, planning, communication and requirements prioritisation) in preliminary S-Scrum concerning "**verification**" and "**communication**", we propose STPA-BDD in the preliminary S-Scrum.*

The main contributions of this chapter are:

- **We propose STPA-BDD in S-Scrum.** *We propose to use BDD combining with STPA for safety analysis and verification based on systems theory in ASD.*
- **We develop a semi-automated tool to speed up BDD.** *The tool supports generating BDD test scenarios and test cases partially by clicking one bottom rather than writing full test suites.*
- **We evaluate STPA-BDD as well as a semi-automated tool by conducting controlled experiments with overall 55 participants.** *The results show that STPA-BDD is an effective method concerning communication effectiveness, while the semi-automated tool promotes a high productivity, test thoroughness and fault detection effectiveness when performing STPA-BDD.*

5.1 Concept

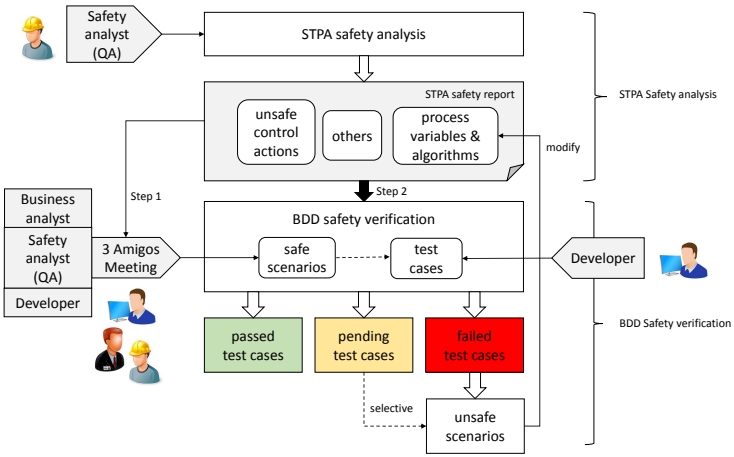


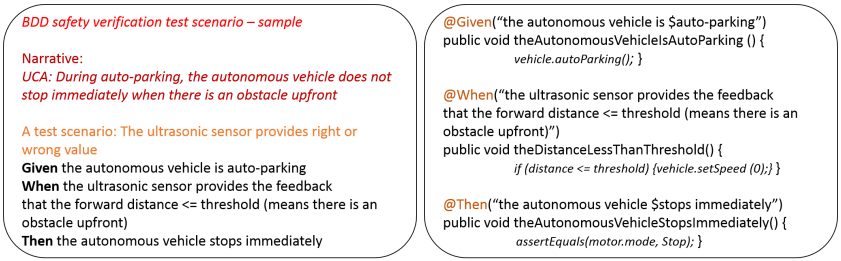
Figure 5.1: STPA-BDD Concept

STPA is evaluated to be suitable for performing safety analysis during each iteration in a scrum development process. However, the derived safety requirements from STPA need verification, which should be in accordance with systems theory. We propose to use BDD in combination with STPA for safety verification based on the following reasons: (1) BDD is able to guide design at an early stage; (2) BDD strengthens communication; (3) BDD focuses on verifying system behaviour.

As we can see in Figure 5.1, STPA-BDD has two main parts: STPA safety analysis and BDD safety verification. A safety analyst or a QA expert starts performing STPA safety analysis when there is a sufficient amount of code after the discussion with development team. STPA is executed by firstly identifying potentially hazardous control actions, and secondly determining how UCA could occur. STPA derives the safety requirements, which constraint the UCAs, as well as system behaviours. Additionally, it explores the causal factors in scenarios for each UCA. The output from the safety analyst (the QA expert) is an STPA safety report with system description, control

structure, accidents, hazards, UCAs, corresponding safety requirements, process variables and algorithms.

In BDD safety verification, to generate test scenarios, the UCAs (in STPA step 1), process variables and algorithms (in STPA step 2) from the STPA safety report are needed. We write other data into "others" in Figure 5.1. BDD safety verification has two steps: In step 1, the business analyst, the safety analyst (the QA expert) and the developer establish a "3 Amigos Meeting" to generate test scenarios. In a BDD test scenario, we write the possible trigger event for the UCA in **When [Event]**. The other process variables and algorithms are arranged in **Given [Context]**. **Then [Outcome]** presents the expected behaviour - A safe control action. In Figure 5.2 (a), we present an example. The safety analyst (the QA expert) has provided a UCA as: *During auto-parking, the autonomous vehicle does not stop immediately when there is an obstacle upfront.* One of the process variables with relevant algorithms detects the forward distance by using an ultrasonic sensor. The developer considers a possible trigger as the ultrasonic sensor provides the wrong feedback. Thus, a BDD test scenario should test: *If the ultrasonic sensor provides the feedback that the forward distance <= threshold (means there is an obstacle upfront)* and whether the vehicle stops. They write this after **When**. The context could be: *The autonomous vehicle is auto-parking.* We write them after **Given**. **Then** constraints the safe control action as *the autonomous vehicle stops immediately*. More possible triggers are expected to be generated after **When** to test them. We illustrate a BDD test scenario using only three basic steps "Given" "When" "Then". More annotations, such as "And", can also be added. In step 2, after the three amigos discuss and determine the test scenarios, the developer starts generating them into test cases, as shown in Figure 5.2 (b). BDD test cases use annotations such as **@Given**, **@When**, and **@Then** to connect the aforementioned test scenarios with real code. The developer produces code to fulfill each annotation. We can identify unsafe scenarios when the test cases fail. We correct the trigger event to pass the test cases to satisfy the safety requirement.



(a) Test scenario example (b) Test case example

Figure 5.2: BDD Safety Verification Example

5.2 Evaluation: STPA-BDD

We conduct a controlled experiment following the guideline by Wohlin et al. [Woh+12]. The goal is:
Analyse BDD and UAT[1] for safety verification.
For the purpose of comparing their effect.
With respect to *productivity* by measuring the number of implemented (tested) user stories per minute; *test thoroughness* by measuring line coverage; *fault detection effectiveness* by measuring a mutation score indicator; *communication effectiveness* by conducting a post-questionnaire.
From the point of view of the developers and business analysts.
In the context of B.Sc students majoring in software engineering or other related majors executing acceptance testing.

5.2.1 Context

Participants: The experiment ran off-line in a laboratory setting in an "Introduction to Software Engineering" course at the University of Stuttgart. Since the course includes teaching BDD and UAT technology, the students are suitable subjects for our experiment. We arrange them based on Java pro-

[1] To execute UAT, we mainly refer to [CG09] with fictional business analysts.

gramming experiences (not randomly). According to a pre-questionnaire[1], 88.6% of the students are majoring in software engineering. We conclude from Table 5.1 that they have attended relevant lectures and handled practical tasks relating to Java programming, acceptance testing, SCS (with a median value >= 3 on a scale from 1 to 5). The agile techniques show less competency (with a median value of 2 on a scale from 1 to 5). We provide a 1-to-1 training, which lasts 44 hours overall, to reduce the weaknesses.
Development environment: We use a simplified Java code with mutants from a Lego Mindstorms based Autonomous Parking System (APS) and Crossroad Stop and Go System (CSGS). These two systems are comparable by lines of code and number of functional modules[1]. To ease writing test cases, we use a lejo TDD wrapper, Testable Lejos[2] to remove deep dependencies to the embedded environment. The BDD groups (Group A1 and Group A2) operate in an Eclipse IDE together with a JBehave plug-in (based on JUnit)[3]. We use Eclipse log files and JUnit test reports for calculating the number of implemented (tested) user stories. Finally, we use PIT Mutation Testing[4] to assess line coverage and a mutation score indicator. The UAT groups (Group B1 and Group B2) write the test cases in Microsoft Word.

Table 5.1: Medians of the Student's Background

Area	Group A1	Group A2	Group B1	Group B2
Java programming	3	3	3	3
Acceptance testing	4	5	3	3
Safety-critical systems	3	4	4	4
Agile techniques	3	3	3	2

Note: "1" to "5" mean "Non-experienced" to "Experienced".

[1] https://doi.org/10.5281/zenodo.846976
[2] http://testablelejos.sourceforge.net/
[3] http://jbehave.org/eclipse-integration.html
[4] http://pitest.org/

5.2.2 Hypotheses

We formulate the null hypotheses as:
$H_{0\ PROD}$: There is no difference in productivity between BDD and UAT.
$H_{0\ THOR}$: There is no difference in test thoroughness between BDD and UAT.
$H_{0\ FAUL}$: There is no difference in fault detection effectiveness between BDD and UAT.
$H_{0\ COME}$: There is no difference in communication effectiveness between BDD and UAT.
The alternative hypotheses are:
$H_{1\ PROD}$: BDD is more productive than UAT when producing safety test cases.
$H_{1\ THOR}$: BDD yields better test thoroughness than UAT.
$H_{1\ FAUL}$: BDD is more effective regarding fault detection than UAT.
$H_{1\ COME}$: BDD is more effective regarding communication than UAT.

5.2.3 Variables

The independent variables are the acceptance testing techniques. The dependent variables are: (1) Productivity (PROD). It is defined as output per unit effort [EMT05]. In our experiment, the participants test the user stories in the STPA safety report and produce safety test cases. We assess it via the number of implemented (tested) user stories per minute (NIUS) [EMT05]; (2) Test thoroughness (THOR). Code coverage is an important measure for the thoroughness of test suites including safety test suites [Mar99]. Considering a low complexity of our provided systems, line coverage (LC) [Mad10] is more suitable than branch coverage (BC); (3) Fault detection effectiveness (FAUL). Mutation testing [Ham77] is powerful and effective to indicate the capability at finding faults [Mad10]. In our experiment, we measure how well a safety test suite is able to find faults at the code-level. We assess this via a Mutation Score Indicator (MSI) [Mad10]; (4) Communication effectiveness (COME). We assess this via a post-questionnaire with 11 questions for developers covering topics like understandability and 13 questions for busi-

ness analysts covering topics like confidentiality according to Adzic [Adz09]. The results are in a 5-point scale from -2 (negative) to +2 (positive).

5.2.4 Pilot Study

Six master students majoring in software engineering took part in a pilot study. We arranged a four-hour training program. The first author observed the operation and concluded as follows: (1) The STPA safety report was too complicated to be used by inexperienced students. We used a comprehensive STPA report by using XSTAMPP[1] in the pilot study. However, a lot of unnecessary data, such as accidents, hazards and safety requirements at the system-level, influenced the understanding. It costs too much time to capture the information. Thus, we simplified the STPA report with the process variables, algorithms, and UCAs. (2) We used the original Java code from a previous student project. The complex code affected the quick understanding. After the pilot study, we simplified it. (3) Training is important. In the pilot study, one participant had not taken part in the training program, which led to his experiment being unfinished. We provide a textual tutorial and system description for each participant as a backup. (4) We have only used an experiment report to record the measures. However, the pure numbers sometimes cannot show clear causalities. Thus, we use a screen video recording in parallel with the experiment report.

5.2.5 Experiment Operation

As we can see in Figure 5.3, we divide the 44 participants into 4 groups. We provide 2 systems and evaluate 2 acceptance testing methods. Group A1 uses BDD for system 1. Group A2 uses BDD for system 2. Group B1 uses UAT for system 1. Group B2 uses UAT for system 2. We use two systems to evaluate the communication between developers and business analysts. The developers are the participants in each group, while the fictional business

[1] http://www.xstampp.de/

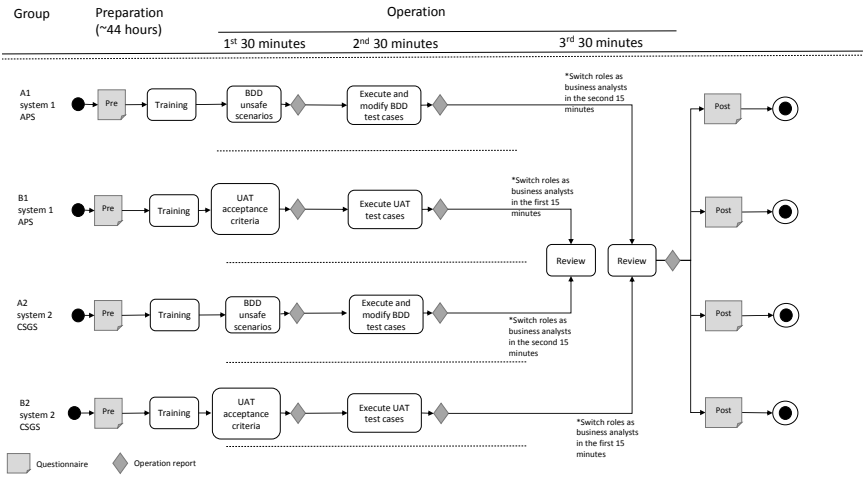


Figure 5.3: Experiment Operation

analysts are portrayed by the participants in the other group using various testing methods and systems.

The experiment consists of 2 phases: *preparation* and *operation*. The *preparation* was run 2 weeks before the experiment to perform the pre-questionnaire and training. The *operation* consists of three sessions (30 minutes/session). In the 1^{st} session, four groups write acceptance test cases. Group A1 (BDD) and Group A2 (BDD) write test scenarios in Eclipse with the Jbehave plug-in as a story file. Group B1 (UAT) and Group B2 (UAT) write acceptance criteria in plaintext. We provide 30 UCA (UCAs) in an STPA safety report. When the students finish all the 30 UCAs in 30 minutes, they record the time in minutes. After the 1^{st} session, the participants record the NIUS and the time in the operation report. In the 2^{nd} session, Group A1 (BDD) and Group A2 (BDD) write each test scenario into a test case and run the test case. If it fails, they should modify the trigger (code) and pass the test case. Group B1 (UAT) and Group B2 (UAT) review Java code, execute the test cases manually and complete their acceptance test report. At the end of the 2^{nd} session, they run PIT mutation testing. The LC and

MSI are generated automatically in the PIT test report. They write down the results in the operation report. In the 3^{rd} session, the participant portrays as a developer for 15 minutes and a business analyst for 15 minutes. The developer is expected to explain his/her testing strategy as clearly as possible, while the fictional business analyst should try to question the developer. To this end, they answer a post-questionnaire.

5.2.6 Results

5.2.6.1 Descriptive Statistic

In Table 5.2, we summarize the descriptive statistics of the gathered measures[1]. To sum up, the results from the two systems in one treatment are almost identical. BDD and UAT have only small differences regarding NIUS and MSI. However, COME in BDD (Mean = 1.27, 1.18; Std.Dev = 0.81, 0.70) and UAT (Mean = -0.05, 0.01; Std.Dev = 1.20, 1.13) differ more strongly. LC has a small difference. In Figure 5.4, we show a clear comparison and can see some outliers concerning LC. In Figure 5.5, we use an alluvial diagram to show COME. We can conclude that BDD has a better communication effectiveness than UAT from the perspective of developers and business analysts respectively (depending on the length of black vertical bar on the right side of Figure 5.5 (a) and Figure 5.5 (b). On the left side, we list 24 sub-aspects of assessing the communication effectiveness. The boldness of the colorful lines indicates the degree of impact. A thicker line has a larger impact on each aspect. We can see six noteworthy values from Fig. 5 (a) that BDD is better than UAT: (4) Test cases have a clear documentation. (5) They could flush out the functional gaps before development. (6) They have a good understanding of business requirements. (7) Test cases have a good organisation and structure. (8) Realistic examples make them think harder. (11) There is an obvious glue between test cases and code. From Fig. 5 (b), five noteworthy values show that BDD is better than UAT: (6) The developers consider safety requirements deeply and initially. (8) It is easy to identify

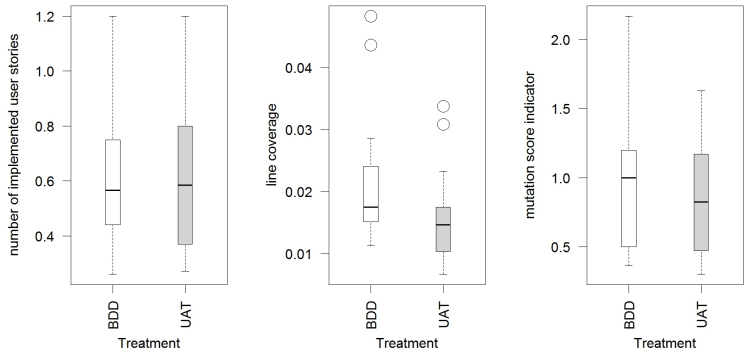
[1] Raw data is available online: https://doi.org/10.5281/zenodo.1154350

conflicts in business rules and test cases. (9) They are confident about the test cases. (12) They are clear about the status of acceptance testing. (13) They could spend less time on sprint-end acceptance testing but more in parallel with development. In addition, the other aspects show also slightly better results when using BDD than UAT.

Table 5.2: Descriptive Statistic

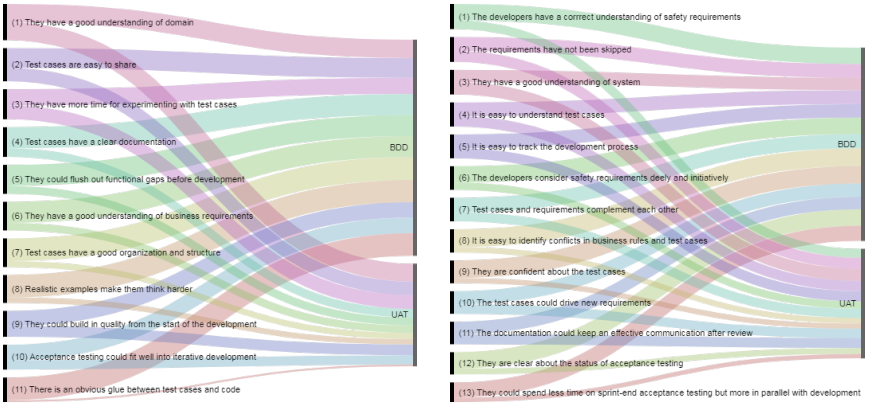
Measure	Treatment	Experiment	Mean	St.Dev	Min	Median	Max	95% CI lower	95% CI upper
NIUS	BDD	Group A1	0.52	0.24	0.26	0.45	1.20	0.37	0.66
		Group A2	0.69	0.19	0.42	0.65	1.00	0.58	0.80
	UAT	Group B1	0.58	0.22	0.33	0.57	1.00	0.45	0.71
		Group B2	0.67	0.29	0.27	0.60	1.20	0.50	0.84
LC	BDD	Group A1	0.02	0.01	0.01	0.02	0.05	0.02	0.03
		Group A2	0.02	0.01	0.01	0.02	0.04	0.02	0.03
	UAT	Group B1	0.02	0.01	0.01	0.01	0.03	0.01	0.02
		Group B2	0.02	0.01	0.01	0.01	0.03	0.01	0.02
MSI	BDD	Group A1	0.90	0.38	0.36	1.00	1.33	0.67	1.13
		Group A2	0.93	0.49	0.44	0.83	2.17	0.63	1.22
	UAT	Group B1	0.89	0.36	0.42	0.88	1.56	0.67	1.10
		Group B2	0.85	0.46	0.30	0.65	1.63	0.58	1.12
COME	BDD	Group A1	1.27	0.81	-2.00	**1.50**	2.00	0.79	1.75
		Group A2	1.18	0.70	-1.00	**1.00**	2.00	0.76	1.58
	UAT	Group B1	-0.05	1.20	-2.00	**0.00**	2.00	-0.75	0.66
		Group B2	0.01	1.13	-2.00	**0.50**	2.00	-0.67	0.67

Note: St. Dev means norm deviation; CI means confidence interval. NIUS means number of implemented (tested) user stories per minute. LC means line coverage. MSI means mutation score indicator. COME was assessed via questionnaire with the results in a 5-point scale from -2 (negative) to +2 (positive).



(a) NIUS (b) LC (c) MSI

Figure 5.4: Boxplot for PROD, THOR and FAUL



(a) Developer's perspective (b) Business analyst's perspective

Figure 5.5: Alluvial diagram for communication effectiveness

5.2.6.2 Hypothesis Testing

As we can see in Table 5.3, to start with, we evaluate the pre-questionnaire. No statistically significant differences between BDD and UAT groups are found concerning Java programming, acceptance testing, knowledge on SCS and agile techniques (t-test, $\alpha = 0.05$, $p > 0.05$ for all test parameters). Furthermore, we test the normality of the data distribution with Kolmogorov-Smirnov and Shapiro-Wilk tests at $\alpha = 0.05$. The results show that the data for NIUS in Group A1, for LC in Group A1, A2, B2 and for MSI in Group A1, A2 are not normally distributed. Thus, we use non-parametric tests in the analysis. In addition to the use of p-values for hypotheses testing ($\alpha = 0.05$, one-tailed) from the Mann-Whitney test, Wilcoxon test and ANOVA test, we include the effect size Cohen's d. Since we expect BDD to be better than UAT, we use one-tailed tests. NIUS is not significantly affected by using the BDD or the UAT approach (system 1: $p=0.206$; system 2: $p=0.359$, non-significant). LC is not significantly affected by using BDD or UAT (system 1: $p=0.057$; system 2: $p=0.051$, non-significant). MSI shows no statistically significant difference between using BDD or UAT (system 1: $p=0.472$; system

2: p=0.359, non-significant). However, COME is significantly different (system 1: p<0.00001; system 2: p<0.00001, significant). We accept the alternative hypothesis that BDD shows better communication effectiveness than UAT. Cohen's d shows the values around 0.2, which signifies small effects, around 0.5 stands for medium effects and around 0.8 for large effects. Thus, for COME, system 1 shows a large effect (d = 2.908). For LC we have both medium effects (system 1: d = 0.684; system 2: d = 0.662). The rest of the effects are small, as shown in Table 5.3.

Table 5.3: Hypothesis Testing

System	Hypothesis Testing	U	p-value	Z	W	F	Effect size d	Reject H_0 or not
System 1	NIUS	47.5	0.206	-0.821	25	0.394	0.257	Not reject
	PPAT	11	**0.001**	3.218	0	19.322	2.003	Reject
	LC	36	0.057	1.576	18	2.341	0.684	Not reject
	MSI	59	0.472	0.066	32.5	0.007	0.027	Not reject
	COME	2.5	< **0.00001**	5.877	0	97.276	2.908	Reject
System 2	NIUS	54.5	0.359	0.361	29.5	0.043	0.082	Not reject
	PPAT	28	**0.018**	2.101	1	11.693	1.551	Reject
	LC	35	0.051	1.642	12	2.192	0.662	Not reject
	MSI	54.5	0.359	0.361	24	0.138	0.168	Not reject
	COME	28	< **0.00001**	5.351	3	71.208	0.238	Reject

Note: We compare Group A1 and B1 in system 1 and Group A2 and B2 in system 2. U is from Mann-Whitney testing, W is Wilcoxon W, F is ANOVA F.

5.2.7 Discussion

The main benefit of STPA-BDD is the support for communication effectiveness. The other measures show also some remarkable results. The *productivity* has no statistically significant difference between BDD and UAT. That contradicts our original expectation. We would expect BDD, as an automated testing method, to be more productive than manual UAT. Yet, as the students are not experts in our experiment, they need considerable time to get familiar with the BDD tool. The students use Jbehave to write BDD test cases in our experiment, which has strict constraints on hierarchy and naming conventions to connect test scenarios with test cases. UAT should be easier to learn. We therefore analysed our video recordings and found that BDD developers use nearly 25% to 50% of their time to construct the hierarchy and naming. Scanniello et al. [Sca+16] also mentioned this difficulty when students apply TDD. In the future, we plan to use skilled professionals in

test automation to replicate this study. This could lead to different results. The *test thoroughness* and *fault detection effectiveness* show a non-significant difference between BDD and UAT. We could imagine that our provided Java code is too simplified to show a significant difference. The mutants are easily found with a review. These aspects need further research.

The *communication effectiveness* shows better results by using BDD than UAT on 24 aspects. We highlight 11 significant aspects. The *developers* found that: **BDD has a clear documentation.** A clear documentation of acceptance test cases is important for communication [HS12]. The scenarios are written in plain English with no hidden test instrumentation. The given-when-then format is clear for describing test scenarios for safety verification based on system theory. **The developers using BDD could flush out functional gaps before development.** The communication concerning safety could happen at the beginning of the development. They discuss safety requirements with the business analysts and spot the detailed challenges or edge cases before functional development. UAT happens mostly at the end of the development. It makes the rework expensive and is easy to be cut in SCS. **The developers using BDD have a good understanding of the business requirements.** A good understanding of safety requirements helps an effective communication. They could build a shared understanding in the “3 Amigos Meeting" to ensure that their ideas about the safety requirements are consistent with the business analysts. The developers using UAT might understand safety requirements with a possible bias. **BDD test cases have a good organisation and structure.** This makes the test cases easy to understand, especially during maintenance. They include strict naming conventions and a clear hierarchy to manage test scenarios and test cases. **Realistic examples in BDD make the developers think harder.** The safety requirements are abstract with possibly cognitive diversity, which leave a lot of space for ambiguity and misunderstanding. That negatively influences effective communication. Realistic examples give us a much better way to explain how safe scenarios really work than pure safety requirements do. **There is an obvious glue between BDD test cases and code.** There is glue code in BDD safety verification, which allows an effective separation between

safety requirements and implementation details. This glue code supports the understanding and even communication between business analysts and developers. In addition, it ensures the bidirectional traceability between safety requirements and test cases. The *business analysts* thought that: **The developers using BDD consider the safety requirements deeply and initiatively.** The collaboration promotes a sense of ownership of the deliverable products. That increases an initiative communication. Instead of passively reading the documents, the developers participate in the discussion about writing test scenarios and are more committed to them. **The business analysts are more confident about the BDD test cases.** Confidence promotes effective communication [Adl77]. The business analysts could give a big picture with safety goals to the developers. Feedback from developers and their realistic unsafe scenarios give the business analysts confidence that the developers understand the safety goals correctly. **It is easy to identify conflicts in business rules and test cases when using BDD.** BDD has a set of readable test scenarios focusing on business rules (safety requirements). Each test scenario and test case are directly connected to the code. The business analysts can pull out test cases related to a particular business rule. This helps communication, especially when there is a changing request. **The business analysts are clear about the status of acceptance testing when using BDD.** It promotes a state-of-art communication. That can be attributed to the automated test suites, which might be connected with a continuous integration server and a project management tool to receive a verification report automatically. **The business analysts could spend less time on sprint-end acceptance tests but more in parallel with development.** They can verify the safety requirements periodically and therefore enhance communication throughout the project.

5.2.8 Threats to Validity

5.2.8.1 Internal validity

First, note that we have four groups in our experiment. To avoid a multiple group threat, we prepare a pre-questionnaire to investigate the students' background knowledge. The results of the t-tests show no statistically significant differences among the groups concerning each measure. *Second*, concerning the instrument, UAT is faster to learn than BDD regarding the use of tools. Even though we provide a training to narrow the gap, the *productivity* might have been influenced, since the students have to get familiar with the hierarchy of writing test suites in a BDD tool. The artifacts, such as tutorials and operation report, are designed respectively with the same structure to avoid threats. In addition to the observation, we save the participants' workspaces after the experiment and video recordings for deep analysis. *Third*, the students majoring in software engineering might identify more with the developer role than the business analyst role. Thus, we design two comparable systems. The students in each pair use different systems and test approaches to reduce the influence of prior knowledge. Moreover, we provide a reference [Gre12] on how to perform as a business analyst in an agile project. We also mention their responsibilities in the training.

5.2.8.2 Construct validity

First, the execution of BDD is a variant. BDD should begin with writing tests before coding. However, in our experiment, we use BDD for test-last acceptance testing rather than test-driven design. Thus, we provide source code with mutants. The measures we used could be influenced. In BDD test-first, we write failing test cases first and work on passing all of them to drive coding. According to [HH09] [RM13], BDD test-first might be as effective as or even more effective than BDD test-last. *Second*, the evaluation concerning productivity, test thoroughness, fault detection effectiveness and communication effectiveness does not seem to be enough. As far as we know, our study is the first controlled experiment on BDD. We can base our

measurement (PROD, THOR, FAUL) mainly on TDD controlled experiments and some limited experiments on safety verification. There might be better ways to capture how well safety is handled in testing.

5.2.8.3 Conclusion validity

First, concerning violated assumptions of statistical tests, the Mann-Whitney U-test is robust when the sample size is approximately 20. For each treatment, we have 22 students. Moreover, we use Wilcoxon W test as well as Z to increase the robustness. Nevertheless, under certain conditions, non-parametric rank-based tests can themselves lack robustness [Kit+17]. *Second*, concerning random heterogeneity of subjects, we arranged them based on the Java programming experience. According to the pre-questionnaire, the students are from the same course and 88.6% of them are in the same major.

5.2.8.4 External validity

First, the subjects are students. Although there are some skilled students who could perform as well as experts, most of them lack professional experience. This consideration may limit the generalisation of the results. To consider this debatable issue in terms of using students as subjects, we refer to [Fal+18a]. STPA was developed in 2012, so there is still a lack of experts on the industrial level. BDD has not been used for verifying safety requirements. Thus, we believe that using students as subjects is a suitable way to aggregate contributions in our research area. We also refer to a study by Cleland-Huang and Rahimi, which successfully ran an SCS project with graduate students [CHR17]. *Second*, the simplicity of the tasks poses a threat. We expect to keep the difficulty of the tasks in accordance with the capability of students. Nevertheless, the settings are not fully representative of a real-world project.

5.3 A Semi-Automated Tool

Based on the results of the previous controlled experiment on STPA-BDD, we develop a semi-automated tool to speed up the execution of the second part of STPA-BDD, BDD for safety verification.

Compared with manual BDD, semi-automated BDD is sped up in two steps, as shown in the grey boxes in Figure 5.6. The first step is to generate test scenarios, while the second step is to generate test cases. The automatically generated test scenarios need manual revision to remove the unrealistic or overlapping scenarios. Nevertheless, a lot of time has been saved by these two steps. The selection of UCA is done by the three amigos depending on the progress of project and the requirements of each iteration, which needs human to be the decision makers. The writing of code in test cases is also done manually by developers, since there is a need to understand the test scenarios in natural language and call the relevant source code.

The semi-automated tool called ***TESTS4JBEHAVE***. The input for using this tool is an STPA safety report (from STPA for safety analysis) including UCA, process variables and algorithms. The STPA safety report needs to be recorded in .csv files. The selection of generation tools is not restricted. We use XSTAMPP[1] with some modifications[2]. We trigger *TESTS4JBEHAVE* by running Java command line code[3]. After that, all the test scenarios for the selected UCA are shown in a specific folder, while the test cases are shown in another specific folder.

[1] http://www.xstampp.de

[2] We add a new class for calling process variables and their values in the open source tool: XSTAMPP.

[3] A possible Java command line code is: *java -jar tests4jbehave/tests4jbehave-1.0.0-SNAPSHOT-jar-with-dependencies.jar -j res/imports.txt -k res/declarations.txt -r workspace/autonomous-parking-system/src/test/ -s java/com/bddexperiment/jbehave/ -p com.bddexperiment.jbehave -v res/variable-types.csv -x res/xstampp-all.csv -z res/variables-values.csv -u res/uca-id-list.csv -e pairwise.*

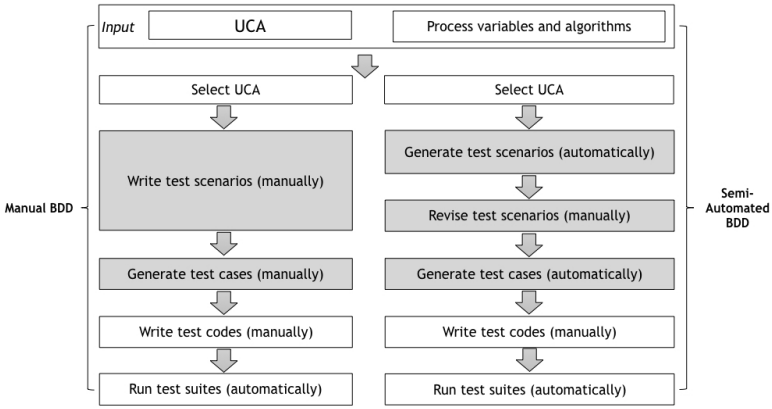


Figure 5.6: Comparison between Manual BDD and Semi-Automated BDD

5.4 Evaluation: Semi-Automated Tool

We evaluate ***TESTS4JBEHAVE*** by conducting a replicated experiment. We refer to [Kit08] and [Shu+08]. Besides a new treatment that we use BDD with the semi-automated tool (TESTS4JBEHAVE), other materials as well as experiment operation are kept as same as possible to our original experiment. In this evaluation, we aim to:

Analyse semi-automated BDD (BDD-R), manual BDD and manual UAT for safety verification.

For the purpose of comparing their impacts.

With respect to *productivity* by measuring the number of implemented (tested) user stories per minute; *test thoroughness* by measuring line coverage; *fault detection effectiveness* by measuring a mutation score indicator.

From the point of view of developers.

In the context of B.Sc students majoring in software engineering or other related majors executing BDD and UAT.

5.4.1 Replicated Experiment

The *context* is as same as the original experiment with 11 additional participants from a "Program Development" course at the University of Stuttgart. We use the same questionnaire to test the participants' background.

As we can see in Table 5.4, the participants in BDD-R handle the relevant knowledge with a median value of 3. In terms of acceptance testing, Group A1 and Group B1 are selected from the course "Introduction to Software Engineering", which encompasses teaching BDD, while Group A1-R is selected from "Program Development", which does not contain the introduction of BDD. Thus, we provide the 1-to-1 training to Group A1-R, as same as what we facilitated to Group A1 and Group B1 in the original controlled experiment. The development environment is almost the same settings. Only Group A1-R has a semi-automated tool, which is developed as a plug-in in Eclipse. They need to import the requirements files and trigger the semi-automated tool. The test suites are generated automatically. Then, they run the test suites as same as Group A1.

Table 5.4: Medians of the Students' Background

Area	UAT Group B1	BDD Group A1	BDD-R Group A1-R
Java programming	3	3	3
Acceptance testing	3	4	3
Safety-critical systems	4	3	3
Agile techniques	3	3	3

Note: From "1" to "5" mean "Non-experienced" to "Experienced".

The *variables* are the same, while the *operation* period is reduced to 1 hour with 2 sessions. One week before the replicated experiment operation, we prepare a training and pre-questionnaires. In the first session, Group A1-R selects the UCA (safety requirements) and generates test scenarios automatically, which happens only several seconds. After that, they revise the generated test scenarios and remove the unrealistic and overlapped scenarios. When the students finish the UCA (safety requirements) in 30

minutes, they record the number of implemented (tested) user stories and time in minutes. In the second session, Group A1-R generates test scenarios into test cases automatically by using the semi automated tool. The other steps are in the same as Group A1. The *null hypotheses* H_0 and the *alternative hypotheses* H_1 are as follows:

$H_{0\ PROD-BDD}$: There is no difference in productivity between BDD-R and BDD.

$H_{0\ THOR-BDD}$: There is no difference in test thoroughness between BDD-R and BDD.

$H_{0\ FAUL-BDD}$: There is no difference in fault detection effectiveness between BDD-R and BDD.

$H_{0\ PROD-UAT}$: There is no difference in productivity between BDD-R and UAT.

$H_{0\ THOR-UAT}$: There is no difference in test thoroughness between BDD-R and UAT.

$H_{0\ FAUL-UAT}$: There is no difference in fault detection effectiveness between BDD-R and UAT.

$H_{1\ PROD-BDD}$: BDD-R is more productive than BDD when producing safety test cases.

$H_{1\ THOR-BDD}$: BDD-R yields better test thoroughness than BDD.

$H_{1\ FAUL-BDD}$: BDD-R is more effective regarding fault detection than BDD.

$H_{1\ PROD-UAT}$: BDD-R is more productive than UAT when producing safety test cases.

$H_{1\ THOR-UAT}$: BDD-R yields better test thoroughness than UAT.

$H_{1\ FAUL-UAT}$: BDD-R is more effective regarding fault detection than UAT.

5.4.2 Results

5.4.2.1 Descriptive Statistic

As we can see in Table 5.5, we summarise the descriptive statistics of the gathered measures, namely mean, norm deviation, minimum, median, maximum, 95% lower confidential intervals and upper confidential intervals.

To illustrate our replicated experiment clearly, we compare the results of BDD-R, BDD and UAT. Three measures between BDD-R and BDD, as well as BDD-R and UAT, show statistically significant differences.

We describe the mean values. In terms of NIUS, BDD-R tests 3.72 user stories per minute, while BDD tests 0.52 user stories per minute and UAT tests 0.58 user stories. In terms of LC, BDD-R covers 7.85 lines of code, while BDD covers 4.83 lines of code and UAT covers 3.44 lines of code. In terms of MSI, BDD-R can find 1.95 mutants, while BDD can find 0.9 mutants and UAT can find 0.89 mutants. In Figure 5.7, we can have a clear comparison on their intervals as well as some outliers. NIUS has two outliers, LC has two outliers and MSI has three outliers. The quality of test cases highly depend on the individual capability.

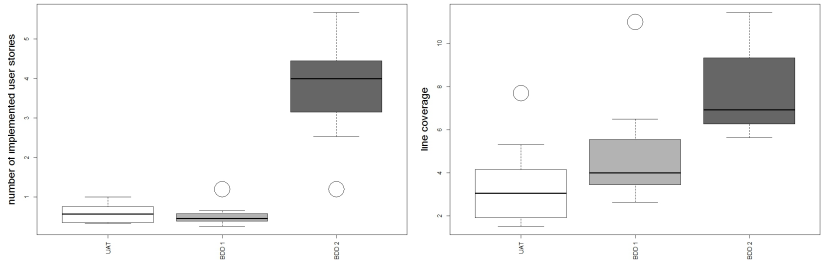
Table 5.5: Descriptive Statistic

Measure	Treatment	Mean	St. Dev	Min	Median	Max	95% lower CI	95% upper CI
NIUS	UAT	0.58	0.22	0.33	0.57	1.00	0.45	0.71
	BDD	0.52	0.24	0.26	0.45	1.20	0.37	0.66
	BDD-R	3.72	1.16	1.20	4.00	5.67	3.03	4.40
LC	UAT	3.44	1.78	1.53	3.06	7.70	2.39	4.49
	BDD	4.83	2.26	2.64	4.00	11.00	3.50	6.17
	BDD-R	7.85	1.86	5.64	6.92	11.43	6.74	8.95
MSI	UAT	0.89	0.36	0.42	0.88	1.56	0.67	1.10
	BDD	0.90	0.38	0.36	1.00	1.33	0.67	1.13
	BDD-R	1.95	0.67	0.99	1.98	3.63	1.56	2.34

Note: St. Dev means norm deviation; CI means confidence interval. NIUS means number of implemented (tested) user stories per minute. LC means line coverage. MSI means mutation score indicator.

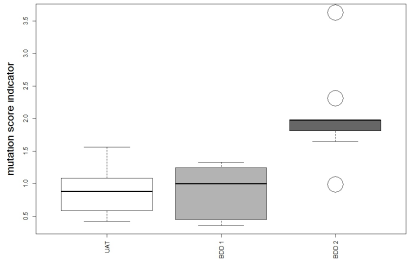
5.4.2.2 Hypothesis Testing

As we can see in Table 5.6, we start by evaluating the pre-questionnaires. The 11 new participants and the 22 original participants show small differences with respect to Java programming, acceptance testing, knowledge on SCS and agile techniques (t-test, $\alpha = 0.05$, $p > 0.05$ for all test parameters). Then, since we used Kolmogorov-Smirnov and Shapiro-Wilk tests ($\alpha = 0.05$) in our original experiment and the data for NIUS in Group A1 are not normally distributed, we use non-parametric tests in our replicated experiment. Finally, we use p-values for hypotheses testing ($\alpha = 0.05$, one-tailed). In addition, the Mann-Whitney test, Wilcoxon test, ANOVA test and the effect size Cohen's



(a) NIUS

(b) LC



(c) MSI

Figure 5.7: Boxplot for PROD (NIUS), THOR (LC) and FAUL (MSI)

d are included. Since we expect BDD-R to be better than BDD and UAT, we use one-tailed tests. NIUS ($p < 0.05$), LC ($p < 0.05$) and MSI ($p < 0.05$) show statistically significant differences.

We accept the alternative hypotheses that BDD-R is more productive than BDD and UAT when producing safety test cases; BDD-R yields better test thoroughness than BDD and UAT. BDD-R is more effective regarding fault detection than BDD and UAT. Cohen's d shows the values around 0.2, which signifies small effects, around 0.5 stands for medium effects and around 0.8 for large effects. Thus, all the results show large effects (d = 3.817, d = 1.459, d= 1.924, d = 3.761, d = 2.422 and d = 1.971).

Table 5.6: Hypothesis Testing

Treatment	Hypothesis Testing	Mann-Whitney U	p-value	Z	Wilcoxon W	ANOVA F	Effect size d	Reject H_0 or not
BDD and BDD-R	NIUS	0.5	< 0.05	-3.907	0	72.918	3.817	Reject H_0
	LC	14	< 0.05	-3.021	3	10.595	1.459	Reject H_0
	MSI	14	< 0.05	-3.021	1	18.619	1.924	Reject H_0
UAT and BDD-R	NIUS	0	< 0.05	-3.940	0	70.512	3.761	Reject H_0
	LC	6	< 0.05	-3.546	0	29.270	2.422	Reject H_0
	MSI	8	< 0.05	-3.415	1.5	19.672	1.971	Reject H_0

Note: NIUS means number of implemented (tested) user stories per minute. LC means line coverage. MSI means mutation score indicator.

5.4.3 Discussion

STPA-BDD is sped up concerning productivity, test thoroughness and fault detection effectiveness by using this semi-automated tool. We evaluate it by replicating our original experiment with 11 new participants (33 subjects overall).

In terms of *productivity*, first, STPA safety analysis results can be automatically generated into test scenarios. Second, developers can construct the hierarchy and name test cases by clicking only one button to trigger the semi-automated tool. Even though the generated test scenarios need human decision makers (3 amigos) to revise and the test code needs human beings (developers) to write. It saves much more time than generating test suites totally manually. More specifically, this semi-automated tool poses a small step of realising continuous test automation [BH06] when using STPA safety analysis and BDD safety verification in agile development. In terms of *test thoroughness and fault detection effectiveness*, the automatically generated test scenarios provide more comprehensive test sets based on the provided UCA (safety requirements), process variables and algorithms. Additionally, generating test suites automatically can avoid human errors to a great extent, which enhances the quality of safety test suites.

5.4.4 Threats to Validity

The replicated experiment is conducted around four months after the original experiment. To ensure the completeness of this partially replication, we recorded the details of our original experiment in an article [WW18a]. The other threats to validity remain the same as our original experiment.

5.5 Conclusion

In conclusion, STPA-BDD is a possible solution for facing the challenge in the preliminary S-Scrum in terms of "*verification*" and "*communication*". It supports an effective communication. By developing the semi-automated tool, we speed up BDD. The productivity, test thoroughness and fault detection effectiveness of STPA-BDD with the semi-automated tool shows positive results (7 times greater concerning productivity, 1.5 times greater concerning test thoroughness, 2 times greater concerning fault detection effectiveness) by comparing with UAT. In the future, our research context is expected to be extended to real-world industrial project settings.

Chapter 6

Documentation in S-Scrum

*In this chapter, to face two of the four challenges (verification, planning, communication and requirements prioritisation) in preliminary S-Scrum concerning "**planning**" and "**communication**", we improve documentation in the preliminary S-Scrum.*

The main contributions of this chapter are:

- **We adapt and develop three documents in S-Scrum.** *We adapt three safety documents' patterns and develop them for our S-Scrum, namely safety epic, safety story and agile safety plan.*
- **We evaluate these three documents in a one-year student project with 14 participants.** *The project uses these three documents in two months. The results from interviews and questionnaires show that safety epics and safety stories have a good capability to support communication, while the agile safety plan supports planning and certification*.

6.1 Concept

6.1.1 Safety Epic

Safety epics illustrate the high-level safety requirements, which are produced mostly by the discussion between users and technical developers rather than analysing the system structures and functions. Few difference between various safety analysis techniques was found in the format of safety epics. Thus, we implement directly the safety epic pattern from [MS16] without modifications:
To satisfy <the overall safety needs> **the system must** <always be able to reach a safe state>.

6.1.2 Safety Story

Safety stories record the safety requirements, which are either from norm requirements or safety analysis [SM16b]. Since different projects use various safety analysis techniques, the expressions of safety stories seem to be different. Safe Scrum uses IF-FMEA for safety analysis, while preliminary S-Scrum uses STPA inside each sprint. FMEA is based on reliability theory, which believes that accidents are caused by single component's or function's failures, whereas STPA is developed on the systems theory, which believes that accidents are caused by the unsafe control actions (UCA). In the systems theory, accidents occur when component failures, external disturbances, and/or dysfunctional interactions among system components are not adequately controlled, rather than single causal variables or factors [Lev11]. Thus, the safety requirements need to be formulated from the UCA. For this reason, we modify the safety story pattern: **To keep** <function> safe, **the system must** <achieve or avoid something> to: **To keep** <the control action> safe, **the system must** <achieve or avoid something>.

6.1.3 Agile Safety Plan

Referring to the norms, Safe Scrum suggests using an agile safety plan to have an overview of the whole safety assurance outside the iterations and further to support certification. In project "Smart Home", due to the disturbed communication between the safety expert and the development team, the insufficient planning and the conflict of priorities between safety and functional development, we implemented an agile safety plan. Preliminary S-Scrum advocates integrating STPA safety analysis into agile development. Hence, the STPA related information has to be integrated into the agile safety plan. Furthermore, due to the iterative safety analysis, except the part of a comprehending overview for certification, some parts of the agile safety plan pattern need to be updated iteratively. We aim to balance the upfront and just-in-time planning, and stepwise realise a continuous planning in preliminary S-Scrum. For this reason, we complement step *f.2: hazard identification and analysis* with the steps of STPA. We take an STPA-based safety test plan format [Mon16] as reference. The parts purely related to test in [Mon16] are still in research. As a result, we added: (1) Summary of Changes; (2) Overview of Findings; (3) Safety Requirements; (4) Types of Tests on the agile safety plan pattern. This part is expected to be updated during each sprint by the safety expert. Step f.1, f.3, f.4 in the agile safety plan pattern will not be included, as the risk analysis and management parts have been covered by STAMP [Lev11] [Mon16] as well as STPA. Other steps in the agile safety plan pattern are kept the same.

6.2 Evaluation

6.2.1 Case Study

The validation process is following Runeson et al.'s case study guidelines [RH09]. The data collection period concentrating on agile safety documentation was between 2017-01 and 2017-02 in the student project "Smart Home" at the University of Stuttgart, Germany. The project has been organised

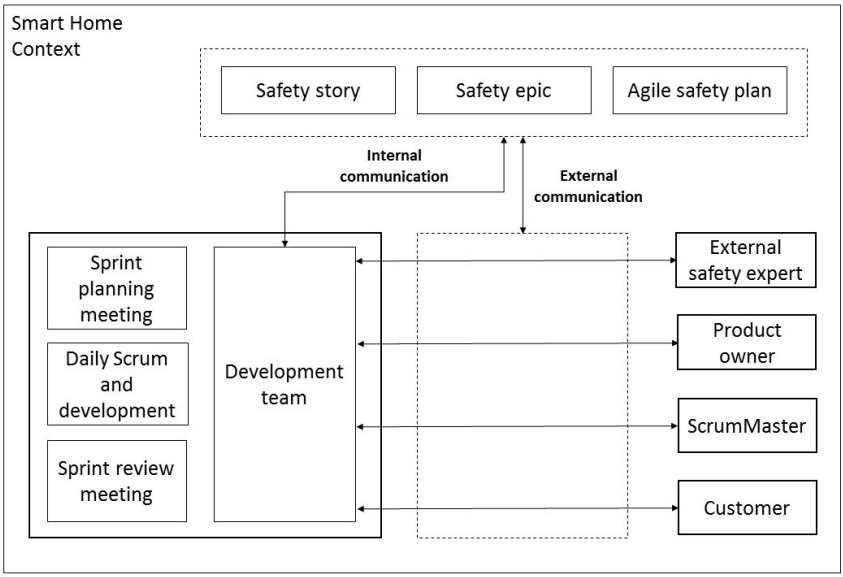


Figure 6.1: Result Analysis Framework

as a regular part of the software engineering curriculum for bachelor students. The students have joined a training for agile development and safety background knowledge before joining the project. During the project, they have also joined the courses such as automation system and programming language. We collected the data through participant observation, scrum artifacts, documentation review and questionnaires combined with interviews. We use a scale of 5 (from "-2" to "+2" means from "negative" to "positive" effect) and calculate Median and Median Absolute Deviation (MAD) to analyse the quantitative results. Qualitative results have been coded by two researchers.

6.2.2 Results

As shown in Figure 6.1, we discuss the results of each document mainly in line with the enhancement of internal communication and external communi-

cation. We use the definition in [Pik+08]. Internal communication highlights the communication within the software development team, whereas external communication in our context means the communication between development team and product owner, scrum master, safety expert and the customer in our context. The internal communication and external communication are further investigated in sprint planning meetings, daily scrums, development, and sprint review meetings. We use quantitative data to show the general effect on communication. The safety culture has also been analysed by quantitative data. Afterwards, detailed observations are derived by qualitative data.

6.2.2.1 Safety Story

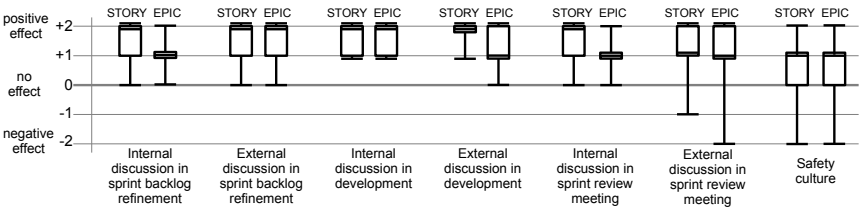


Figure 6.2: Effect on Communication of Safety Story and Safety Epic

As shown in Figure 6.2, the safety story pattern in our context shows positive values, together with little fluctuation between each respondent. Only the external communication in the sprint review meetings and safety culture reveal some doubts.

Positive effect on communication: In the sprint planning meetings, some developers said: "*The safety stories specify a good structure with a clear purpose, which is a basement for the internal discussion.*" Furthermore, they said: "*The safety stories reflect safety concerns and wishes in a clear manner, which allows a constructive safety discussion with safety expert.*" In daily scrum: "*It helps the developers driving an initial idea.*" As the safety story pattern focuses on "what" rather than "how", it does not interfere the concrete developing

tasks and motivates various ideas and discussions in the development team. Moreover, the developers prefer more communication with the safety expert to clarify their execution tactics. To compare with the functional user story pattern, a safety story is mostly produced by an objective safety analysis or norm requirements. A purpose-go-first expression can better reflect safety requirements' properties.

The safety culture shows a positive value by using the safety story pattern. The developers thought that: "*Safety stories are the most detailed unit of safety features.*" Without a clear format of the most detailed unit, it is difficult to integrate safety culture into concrete development.

Negative effect on communication: In our project, we used Jira as the issue tracking tool. Some developers mentioned that: "*Several safety stories seem still too long to be caught in the backlog page.*" The safety story pattern is explicit. However, safety background knowledge in the development team, which helps writing safety stories, could help to hinder some excessive detailed explanation. The comment block in Jira could be fully utilised. The product owner mentioned also that: "*There is no specific place in Jira for a safety product backlog*". A Jira plug-in could be helpful in the future. Moreover, based on the safety story pattern, a constructive discussion in the development team is still the main purpose. People should not neglect an efficient communication and only rely on the existing safety story pattern. Finally, some developers reflected that: "*The commitment of requirements between the safety expert and the development team has been improved, yet the commitment of accepting each requirement has not been focused on.*" The need for an acceptance criterion has been mentioned, yet an explicit pattern is still lacking [Rub12]. Thus, an acceptance criteria pattern would help to improve the external communication in the sprint review meeting.

General impact: Except the effect on communication, the safety expert gave also the feedback that: "*The safety story pattern did help to have a commitment between the safety expert and the product owner.*" Before the implementation of the safety story pattern, there were a lot of conflicts and

interrelations between functional user stories and safety stories. With an explicit "what" focused safety story, the functional user stories not only have less conflicts to safety stories, but also can be linked with each safety story in Jira. Last but not least, the safety story pattern shows an easy adaptation into other safety analysis techniques, such as STPA in our context.

6.2.2.2 Safety Epic

As shown in Figure 6.2, all the values about safety epics are on a relatively positive level, whereas "external discussion in sprint review meeting" and "safety culture" have more variance.

Positive effect on communication: In a sprint planning meeting, we got the feedback that: "*We have more material support for a safety plan and for starting a discussion.*" It also helps to clarify confusing safety stories between safety expert and development team. Since some safety stories seem similar, a lack of a clear description about the system background will cause misunderstandings. Furthermore, because of the increasing amount of requirements, safety stories are necessary to be grouped, especially in complex system, such as Internet of Things (IoT). It helps the discussion in each sub-group and enables a clear identification of the safety scope in different sub-systems. The pre-planned safety epics ensure that the safety consideration has been put into the agenda, which motivates the development team to think about safety from the beginning of the project.

Negative effect on communication: However, during the daily scrum, several people have not reviewed the safety epics again. Some developers said: "*The safety epics do not affect the daily development in a critical manner, because they show only a rough idea and are not really helpful to the internal discussion.*" The safety stories have been tailored into detailed tasks during development. The team members focus more on the development of detailed tasks. That causes some doubts on communication in development. In the sprint review, the safety epics seem also not frequently used. "*Everything*

in the review is mostly focused on the safety stories." However, for a better understanding of the status and planning for the next sprint, we need indeed some discussions about the status of safety epics rather than a temporary discussion in the review meetings. To add a time slot on sprint review meetings for safety epics' discussion could keep the information of safety epics spread throughout the development team, the same as in the sprint backlog refinement.

General impact: From the product owner and safety expert viewpoint, a structured safety epic helps the priority management for the backlogs. The separation of safety requirements depending on the various sub-systems and different sources could help managers decide which safety stories should be put into the current sprint. Furthermore, a clear, structured, concrete safety goal has been integrated into system development. A safety epic helps the spread of safety culture in the middle-level, which fills the gap between detailed safety tasks and the generally overall system safety goals.

Table 6.1: Effect of Safety Documents

Documentation	Positive effect on communication	Negative effect on communication	General Impact
Safety story	1. A clear purpose 2. An explicit structure 3. A quick overview 4. Reflect safety concerns and wishes clearly 5. Help building initial idea for further discussion 6. A "what" focused mode motivates more external communication 7. A "what" focused mode does not interfere the development	1. In Jira, some safety stories are still too long to be caught in the backlog page 2. For a better external communication, the description block in Jira could be used 3. No place in Jira for safety product backlog 4. A structured communication based on the initial idea is more important 5. An acceptance criteria format is needed	1. More clear for safety expert and development team to have a commitment when writing them 2. No difficulty to adapt to an other safety analysis method (STPA) 3. Safety culture has been enhanced
Safety epic	1. More material support in sprint planning meeting 2. Group safety stories 3. Great help for complex system (IoT) development 4. Help the discussion about safety stories	1. During development, the use of safety epics is low 2. In sprint review, no planned time slot for safety epics' discussion 3. Difficult to inform the development team when a safety epic is updated	1. Help managing the priorities 2. Give a clear structure of the integrated safety 3. Spread safety culture in the middle-level
Agile safety plan	Need further research	1. A huge gap between the high-level plan and concrete development 2. The developers prefer a direct communication with the safety expert rather than looking through the plan 3. Possible to be replaced by other documents	1. A clear overview and evidence of the process 2. Delivered safety report 3. Help solving conflicts between safety goals and functional goals 4. Backup knowledge for development team

6.2.2.3 Agile Safety Plan

Negative effect on communication: Regarding communication, most of the students cannot give a value for evaluation. We summarise the causalities as follows: First, during the sprint, few developer looked at the agile safety plan. They mentioned that: "*At the beginning, we got a basic idea and description of the safety plan, but later it did not have a major influence on our development.*" Second, there is still a huge gap between the safety plan and detailed development tasks. As one of the interviewee said that: "*We could have a clear overview of the safety in the project, but have no idea about how to integrate this into our daily work.*" Third, the developers prefer more communication with the safety expert rather than review the agile safety plan when needed. The interviewee said: "*The plan is well structured and with not so many pages, but when a problem occurred, we would like to talk to the safety expert directly.*" One of the original ideas in preliminary S-Scrum was using an agile safety plan to help developers solving unclear safety issues. We added STPA technique's information to it. However, the feedback shows communication is a more effective way. This seems not to be contradictory to our main purpose.

The agile safety plan supports a planned safety process and the certification body and to some extent strengthen the communication with the safety expert and/or the certification [MSL16]. Our intention of adding an agile safety plan and integrating parts of STPA information is because of a weak planning level in preliminary S-Scrum and the conflicts in priority management. Thus, although we failed the investigation of communication effectiveness from the agile safety plan in our case, the general impacts could also help the organisation deciding if an agile safety plan is necessary for its project. For example, using an agile safety plan as a deliverable part of the safety-critical product; capturing the important discussion, decision, or agreement between safety expert and development team to have a clear recollection; helping new team members come up to speed quickly with safety knowledge; and when there is a regulatory requirement for some certain safety-related documents [Rub12].

General impact: We derived the values of an agile safety plan from the interviews: For the scrum master, an agile safety plan provides an overview and evidence of the safety process. For the product owner, it is helpful for a continuous planning. For the customers, they are more clear and confident about the safety assurance of their products, and could have additionally a delivered safety report for the invisible safety artefact. For the safety expert, it is easier to have a commitment to the priorities of safety stories and functional user stories for each sprint. For development team, they mentioned that: "*Because of a plan, the developers have a complete safety knowledge as a backup, when safety expert is unavailable.*". A summarised result is depicted in Table 6.1.

6.3 Conclusion

In this chapter, we solve the problems concerning "planning" and "communication" in the preliminary S-Scrum. We adapt and develop three safety documents for S-Scrum. We evaluate them in a one-year student project with 14 participants. The using period of these documents is two months. The data collection lasts one sprint. The results show that safety epic and safety story have a positive effect on internal and external communication, while agile safety plan supports more on planning and certification. In terms of the support for communication of agile safety plan, we need to pursue it in different contexts, especially in industry. In addition, the evaluation period is too short, which may cause limitations to our results.

Chapter 7

Communication in S-Scrum

*In this chapter, to face one of the four challenges (verification, planning, communication and requirements prioritisation) in preliminary S-Scrum concerning "**communication**", we improve communication in the preliminary S-Scrum.*

The main contributions of this chapter are:

- **We investigate communication during safety analysis.** *We investigate specifically communication channels during safety analysis in the preliminary S-Scrum. We point out the importance of clear communication purposes, as well as the possible challenges.*
- **We find 9 communication channels during safety analysis.** *We design a theoretical lens, survey 39 experts and interview 21 experts in 7 safety-critical companies. We find 9 communication channels.*
- **We summarise 28 communication purposes and highlight the importance of "what" to communication.** *The communication purposes are the major contribution of our study. We explore 28 communication purposes within the 9 communication channels to provide practitioners a possible way to select the suitable communication channels.*
- **We explore the Top 10 challenges during safety analysis.** *We derive Top 10 challenges from 21 interviews. We map them further with the 28 communication purposes. We believe that the challenges are highly correlated with the communication purposes rather than the using channels.*
- **We investigate the communication channel, "*meeting*" and "*documentation*", in S-Scrum further.** *Comparing with the other 7 communication channels, such as emails or telephone, S-Scrum differs itself mainly from the increased number of meetings, as well as different types of documentation. We find that the communication channels in S-Scrum can achieve 10 more communication purposes than in our research context.*

7.1 Theoretical Lens

We define a theoretical lens of communication in safety analysis. We refer to the norms ISO 26262, ISO 14971 and IEC 60601 concerning the execution of safety analysis in SCS, as well as several internal safety analysis norms.

Safety analysis happens primarily in four stages in the development of SCS. (1) At the beginning, a safety expert (he or she could be a functional safety manager, external safety expert or internal safety expert) together with other project members, who are responsible for facing customer projects, derive system-level safety requirements. Popular techniques are Hazard and Operability Analysis (HAZOP), Hazard Analysis and Risk Assessment (HARA) and Fault Tree Analysis (FTA). (2) After architectural design, the safety expert, development teams and other stakeholders derive detailed safety requirements and implement them in development. Popular techniques are Failure Mode and Effect Analysis (FMEA) for software and Failure Modes, Effects and Diagnostic Analysis (FMEDA) for hardware. (3) A verification, such as model checking, is necessary for testing the detailed safety requirements. (4) A validation, such as a review, is done by the deployment department or customers to test the system-level safety requirements. In addition to these four stages, safety analysis might happen during change impact analysis when there is a changing request. The concrete execution of safety analysis depends on whether they aim to develop a new product or reuse a product.

The theoretical lens is shown in Figure 7.1. The communication encompasses internal communication and external communication. We simplify the roles in the **development team** with one **internal safety expert**, who mainly performs safety analysis, and other team members, which include architects, developers, testers and so on. The internal communication happens between the internal safety expert and other team members. Besides the development team, the industries also establish a **functional safety department**, which takes responsibility for the whole functional safety issues at the company level. In the functional safety department, a **functional safety manager** fixes mainly the external affairs and monitoring the execution

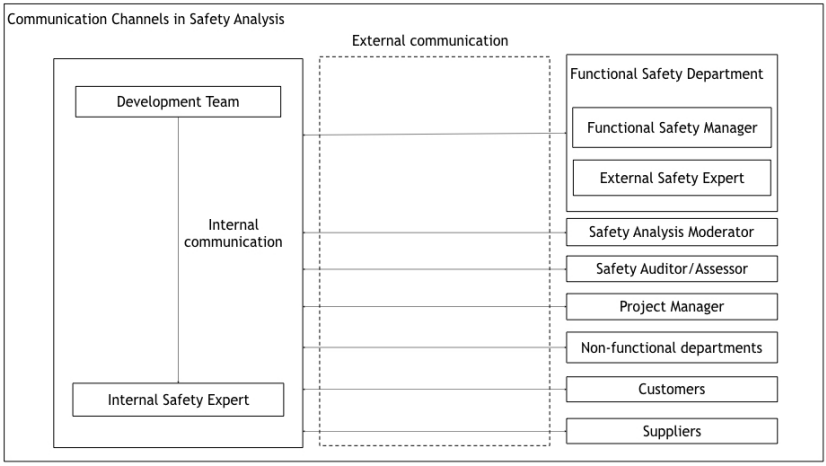


Figure 7.1: Theoretical Lens of Communication Channels

of norms, while an **external safety expert** keeps contact with the internal safety expert to conduct training or knowledge sharing. In industry, the safety analysis related collaborative activities, such as execution and review, are guided by a **safety analysis moderator**, who is from a technical support department. The moderator ensures an official procedure to perform safety analysis. To ensure the safety assurance capability concerning process execution, the industries perform safety audit/assessment periodically by **a safety auditor or assessor**. It happens two to three times per year, especially when there is a deliverable product. The communication happens also between the development team and **project manager**, **customers**, **suppliers** and other **non-functional departments**, when the safety analysis issues concerning project-level, product-level, purchasing, sales and so on.

7.2 Case Study

We chose an exploratory case study design as proposed by Runeson et al. [Run+12]. We conducted this case study in an inductive way by designing a

theoretical lens in Figure 7.1. The reasons are: (1) There is no existing theory on communication channels during safety analysis, such as the possible channels and using regulations. (2) The exploration scope is indeterministic in terms of communication channels in safety analysis due to an omnicooperation. (3) The boundaries of safety analysis activities need to be clear for investigating communications, since some of them are blurred within development activities. This theoretical lens can provide us a clear boundary of our article and lay a foundation for our results.

7.2.1 Context

Table 7.1: Research Context

Company	Size	Location	Domain	Main products	Employees	Participants
A	Large	Germany	Automotive	Automotive parts; Power tools; Electronics; Motorised bicycle motors.	400,500	3
A1	Medium	China	Automotive	Automotive parts; Power tools; Electronics; Motorised bicycle motors.	59,000	26
A2	Medium	China	Automotive	Gasoline engine management systems; Transmission control system; Hybrid and electric drive control system.	9,400	19
A3	Medium	China	Automotive	Diesel systems for passenger cars, light and heavy commercial vehicles.	1,800	3
B	Large	Germany	Medical Equipment	CT/SPECT scanner; Angiography; X-ray products; Molecular diagnostics.	372,000	4
C	Medium	Germany (Italy)	Automotive	Automotive lighting; Powertrain; Suspension systems; Motorsport.	43,000	3
D	Small	Germany	Industrial 4.0 Based Product System	Technical strategy for production; ICT for manufacturing; Communication for factories; Process planning.	Less than 100 (unstable)	2

We conducted a multiple case study in seven safety-critical companies (three of which are the subsidiaries of company A), as we can see in Table 7.1. We selected company A, together with A1, A2 and A3, as our biggest sample (with overall 51 participants). To cover different sizes of companies, we included company C (medium) and company D (small). To expand our research domains (company A is an international automotive industry), we included company B (an international medical equipment industry) and company D (a local industry 4.0 based production system). The duration of

the investigated projects varies from 6 months to 3 years. The safety assurance processes of them follow ISO 26262 (automotive), VDA (automotive), Automotive SPICE, ISO 14971 (medical equipment), ISO 15004 (medical equipment) and IEC 60601 (medical equipment). The safety analysis is following the internal norms of FMEA, FTA and HAZOP. The investigated projects encompass functional development of ECU (Electronic Control Unit), NCU (New energy Control Unit) and body electronic control units, remote monitoring of CT machines, software services for smart factories and so on.

We have in total 60 participants. 39 participants took part in the surveys, while the other 21 participants took part in the interviews. We asked the 21 interviewees before the interviews to ensure that they have not answered the surveys before. Their working experiences in safety-critical companies range from 1 to 23 years. As shown in Figure 7.2, most participants are quality assurance engineers and managers. 17 participants are from the quality assurance (QA) department. 10 participants are from the functional safety department. There are 23 managers and 3 experts. Other roles are 9 developers, 1 analyst and 2 leaders, as well as 1 participant in sales, 1 participant in purchasing and 1 participant in production department.

Figure 7.2: Participants

7.2.2 Research Question

The research goal is to investigate communication channels during safety analysis. We formulate four research questions to steer the design of our study, as shown in Table 7.2.

Table 7.2: Research Questions

RQ 1	Which communication channels are adopted for safety analysis?
RQ 2	How frequently used are safety analysis communication channels?
RQ 3	Which are the purposes of safety analysis communication channels?
RQ 4	Which are the challenges when using safety analysis communication channels?

7.2.3 Data Collection

As we can see in Figure 7.3, we conducted three rounds of data collection incrementally. In the first round, we ran surveys in seven companies between 2017-09-01 and 2017-10-31 to investigate the existing communication channels as well as their usage frequencies. In the second round, we conducted semi-structured interviews and documentation reviews in seven companies separately between 2017-09-01 and 2017-10-31 and between 2017-12-01 and 2018-01-31 to investigate their purposes and challenges. In the third round, from 2017-12-18 to 2018-01-05, the first author participated in several safety analysis meetings and the daily work in a functional safety department (foundation of theoretic software group). The duration of participant observation is three weeks in company A2. Before data collection, we pre-interviewed several experts from four companies, either by telephone or by a face-to-face introductory meeting, to look through the organisational structure , decide on a common objective, establish agreements and help designing the surveys and interview questions. Each interview lasted one hour. These experts were further arranged to be the representative of each company.

In the first round, we used a survey to collect both qualitative and quantitative data, which cover the participant's background (positions, working years, the descriptions and durations of the running projects), the existing communication channels and the frequencies of using each communication channel. We sent the link to each representative via email, as well as the survey in electronic form to ensure that all the participants receive them. Since the investigation including the questions and predictable answers are sensitive in SCS, we decided to hand out the surveys through a company

representative. This leads to a limited number of participants, which poses a challenge. During these two months, the first author checked the progress every two weeks through communicating with the representatives by videophone regarding the distribution of the surveys, as well as problems and feedback.

In the second round, we used semi-structured interviews to investigate the purposes and challenges of the communication channels. We selected firstly the communication channels from the results in the first round. We asked the interviewees to indicate possible additional communication channels as the first question. Moreover, we asked for the possible purposes of each communication channel. We went deeper to gain some real examples. The results may refer to product and customers' information, we keep them confidential. Lastly, we inquired about the challenges. We explained that the challenges they found could be specific to one communication channel, or in a general way across multiple channels. To make each challenge clear, we asked further in-depth sub-questions depending on participants' answers, such as *"Who are the senders and receivers?*", *"What are the possible effects?*", *"How serious are the effects?*", *"What are the causal factors?*", *"How do you treat or fix this challenge?*". We listed some possible in-depth sub-questions in the interview guide. Meanwhile, the first author reviewed the documents, which are related to the communication that happens during safety analysis, such as R&D process instruments including safety analysis activities or interfaces in R&D process, product development reports including the safety part of product development and quality management reports including safety quality management, quality management handbooks, R&D risk management instructions, FMEA, FTA and HAZOP guidelines and working instructions, decision analysis and making reports and technical review reports. We inspected their contents, structures and relevant senders (editors) and receivers (readers or users) to analyse aspects such as *"Are the contents fully written and could they be easy to understand by the receivers (readers and users)?*", *"Are they clearly structured by the senders (editors)?*" and *"How long do the documents need to be looked through and be used?*"

In the third round, the first author conducted a direct observation in

company A2. The first author took part in several team meetings such as a meeting concerning a comparison between the existing safety analysis procedure and a new version of ISO 26262, a meeting to discuss how safety analysis is executed in software and hardware development and how to coordinate with cyber-physical security. The first author observed the daily work in a functional safety department (she sat near to an internal safety expert who performed the safety analysis). Apart from the regular processes such as *"How frequently has the internal safety expert been using for communication by using different channels?*" The first author observed also the internal safety expert's verbal communication with the functional safety manager, developers and stakeholders in non-functional departments, as well as the non-verbal communication, such as the internal safety expert's field notes, which recorded some outputs from communication. In the following section, we present how we analyse our data.

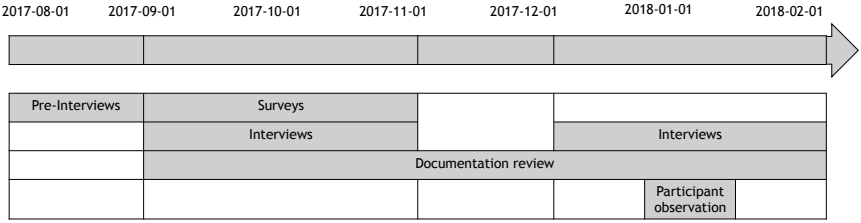


Figure 7.3: Timeline of Data Collection

7.2.4 Data Analysis

To start answering our research questions, we investigate the participants' qualifications, such as positions, working experiences and running projects.

For qualitative data, we consider basic coding steps and follow the logic of grounded theory coding [SC94]. To start with, we conduct **open coding** to keep initial coding open-ended, which without having any preconceived concepts. Since the topic concerning communication channels has a wide

range, even in safety analysis activities, open coding seems the most appropriate way to record transcripts line-by-line. Furthermore, based on an initial coding process, we get some preconceived categories. We list a preliminary category and conduct a **selective coding**. We focus on the codes relating to complementing the existing communication channels for RQ 1 and complementing the frequencies of using each communication channel, their purposes for RQ 3 as well as their challenges for RQ 4. The selective coding process is emergent and iterative. We compare the codes and refine our preconceived categories. For example, in terms of RQ 1, some interviewees mentioned that the communication channels lack a clear clarification, such as *"If a personal meeting with two people is a formal meeting or personal discussion?*" We discuss the bias separately with an expert in company A to complement the results. In terms of RQ 2, after the first round of selective coding, some purposes show similarities, such as *"We use email to inform a temporary meeting*" and *"We call the colleagues to fix an emergent complaint*", we group them as *"A real-time notification*". In addition, there is one sentence of code that encompasses two or more sub-purposes, such as *"We go directly to the sales or deployment (the contact person) to organise a meeting when there comes a temporary but emergent customer's complaint*". It is divided into *"Fix customers' complaints*" and *"Cooperate among multiple functional departments*". We conduct a second round of selective coding to group similar purposes and divide mixed purposes. The same is held for RQ 4 concerning challenges.

Lastly, to connect our results concerning the four RQs coherently, we conduct **axial coding** for RQ 1, RQ 3 and RQ 4. We link the existing communication channels with their purposes. We link the Top 10 challenges with their purposes as well. We demonstrate an example of our coding phase including interview snippet, open coding, selective coding and axial coding in Table 7.3.

For quantitative data concerning RQ 1, we choose pure numbers of participants to represent the utilisation of each communication channel. For quantitative data concerning RQ 2, in the first round, the participants scored the frequencies of the occurrences of each communication channel. The

Table 7.3: Example of Coding Phase

Interview snippet	*Open coding*	*Selective coding (possibly iterative)*	*Axial coding*
Q: "What are the common purposes to facilitate communication concerning safety analysis?" **A**: "...we have to exchange (safety analysis) information with our parent company...but the documents are always missing ... The functions are inherited, detailed architecture design documents are kept by them ... safety analysis cannot be done without considering interfaces with original products ..."	We exchange safety analysis information with our parent company. The documents are missing. The functions are inherited. Detailed architecture design documents are kept by them. Safety analysis cannot be done without considering interfaces with the original products.	**Channels**: Documentation. **Purposes**: Derive safety requirements. Share knowledge.	CHANNEL_Documentation <–> PURPOSES_Derive safety requirements_Share knowledge.
Q: "Have you noticed some challenges in these communication channels?" **A**: "...more importantly, we do care about the confidentiality. We categorised the data related to safety analysis with different confidential levels. High confidential data requires relevant authorities to read or transmit ..."	We care about confidentiality problem. We categorise the safety analysis related information with different confidential levels. High confidential levels' data need authorities to read and transmit.	**Purposes**: Transfer safety requirements. **Challenges**: Transmission of confidential information. Authority problems.	PURPOSES _Transfer safety requirements <–> CHALLENGE_Confidential information_Authority problems.

scale ranges from 1-4 times per day to 1-4 times per year. We consider that only the pure numbers in the surveys might show memory lapses [HRH13], the same as by asking the interviewees, a direct observation is necessary to validate the results.

7.2.5 Results

7.2.5.1 RQ1: Which communication channels are adopted for safety analysis?

As we can see in Figure 7.4, we find 9 communication channels during safety analysis, which are meetings, personal discussion, internal communication software (private), email, telephone, documentation, project coordination tools, training (including tutorial) and boards. Few participants mentioned the use of social communication software. However, considering that it is a

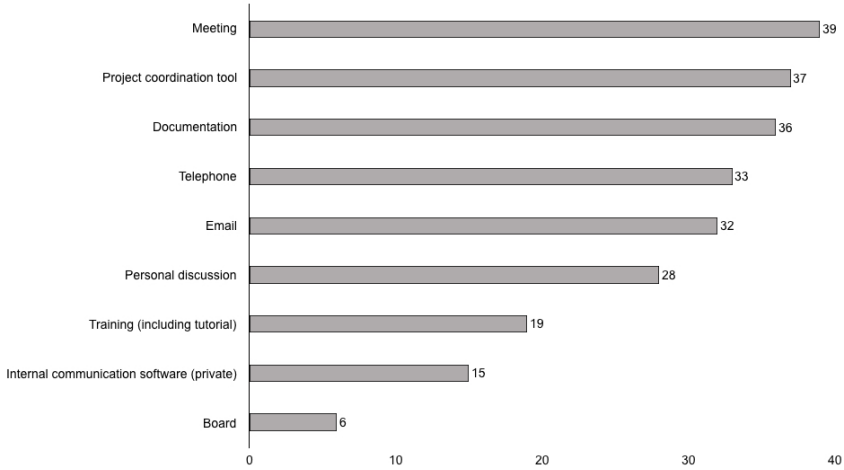


Figure 7.4: Communication Channels in Safety Analysis

rare and non-regulated case[1], we discussed with an expert in company A and decided not to include it in our results. We summarise the results as follows:

Meeting: All the 39 participants (100%) mentioned meeting as a main communication channel during safety analysis. A meeting is defined as a gathering of two or more people that has been convened for the purpose of achieving a common goal through verbal communication, such as sharing information or reaching an agreement. It may take the form of face-to-face or virtually, as mediated by communications technology, such as a telephone conference call, a Skype conference call or a video conference [Ols+92] [Cut+02]. When performing safety analysis, the meetings include planning meetings, review meetings and requirements audit/assessment meetings. The joined members are invited by a meeting organiser, such as a safety analysis moderator. The communication might produce work products as outputs, such as relevant documentation including safety plan or safety

[1]We consider that it is due to a culture difference between Europe and Asia. More descriptions are shown in Section 7.2.7.

requirements implementation decisions.

Project coordination tool: 37 participants (94.9%) mentioned using project coordination tools as a main communication channel during safety analysis. Project coordination tools aim to increase group awareness of current tasks and issues and provide a means for tracking progress and discussing next steps [Sto+17]. Jira is the most popular one in our research context, together with the interfaces to other requirements management tools like Doors to keep the multiple levels and the traceability of safety requirements. Company A, company A1, company A2 and company A3 are using an Application Lifecycle Management (ALM) of IBM Rational Team Concert[1]. In daily work, employees can trace the progresses of safety analysis transparently and provide feedback or comments timely. In other communication channels, such as meetings or personal discussion, project coordination tools provide an up-to-date information, such as the implementation of a safety requirement.

Documentation: 36 participants (92.3%) mentioned documentation as a main communication channel during safety analysis. According to the functional safety norms, safety analysis requires a large amount of documentation, such as safety analysis instrumentations, safety analysis execution plans and safety analysis audit/assessment reports. To record processes and results of safety analysis clearly, the employees have to rely on these documents.

Telephone: 33 participants (84.6%) mentioned the use of the traditional telephone. Even though other modern communication software is springing up, most of the employees believe that telephone is more reliable for local communication and just as easy to use as in daily life [BZL04] [KS92]. We obtained a novel channel calling *a telephone call via Skype* in our research context. Company A links the telephone numbers with the Skype accounts. It enhances the working efficiency. For example, when someone is on the way among different working places, he or she can keep the communication via various mobile terminals, such as mobile phone or tablet personal computer.

[1] https://arcadsoftware.de/produkte/rtc-rational-team-concert/

Email: 32 participants (82.1%) mentioned the use of email. As an asynchronous communication channel, when the issues are not emergent and might involve not only one person, email provides time to structure the information for communication [Dab+05]. Concerning safety analysis, email threads and the contents of emails are traceable. The traceability constitutes an advantage of using email. However, using email is not fully positive during safety analysis. Email can record the process of discussion, yet the practitioners during safety analysis do not prefer recording their discussion, rather only documenting their results. In addition, although using emails seems traceable, there needs a lot of efforts to search for the relevant information.

Personal discussion: 28 participants (71.8%) mentioned personal discussion as a communication channel during safety analysis. Personal discussion is a form of informal face-to-face communication. It happens spontaneously and less supported by communication technology [Kra+90]. When performing safety analysis, an internal communication prefers mostly using personal discussions. It can ensure a correct understanding and a timely feedback. To perform safety analysis in modern software development, personal discussion needs attentions and the effectiveness of it will influence the quality of safety analysis to a great extent.

Training (including tutorial): 19 participants (48.7%) mentioned training including tutorials as a communication channel. Training is a traditional way to enhance personal competence and share knowledge in industries [SB06]. In terms of safety analysis, the education institutes sometimes lack such courses, especially the practical experiences in developing SCS. A training of safety analysis or functional safety norms is a normal way to cultivate employees. However, some of the employees are already experts from other relevant positions or departments in industries, such as product development or quality assurance departments. They have already a deep experience. Thus, not all the participants have to take part in the training.

Internal communication software (private): 15 participants (38.5%) mentioned internal communication software as a communication channel. We consider only the internal communication software that features a private

chat, since not all the software featuring group or public chats in our research context (company A uses Skype that supports public chat, while company A2 uses Microsoft Office Communicator (OCS) that does not support public chat). It is a relative new communication channel popularising in the last decade. It integrates functions like a real-time notification, documents transformation and even meeting organisation [Sto+10]. Even though, the results indicate that internal communication software is not as much used during safety analysis as in other areas, such as software development [Sto+17]. We speculate that concerning each single function, the employees still have a better choice for achieving the purposes of safety analysis, such as for a real-time communication, telephone is more reliable to get in touch.

Boards: 6 participants (15.4%) mentioned the use of boards. Boards, as communication channels, have been popularly used in industries and are becoming a tool during project management, such as whiteboards [Che+07], in modern development processes. They are placed near the working areas. Some of them demonstrate the state-of-the-art or contributions in terms of safety analysis, such as a process flowchart of safety analysis in product development or the developed interfaces in safety analysis and relevant tools. In particular, when external experts or customers come to visit the company, boards are an intuitive way to show competitiveness.

7.2.5.2 RQ2: How frequently used are safety analysis communication channels?

As we can see in Figure 7.5, meetings are mostly held ranging from 1-4 times per week (18 out of 39 respondents, 46.2%) to 1-4 times per month (18 out of 39 respondents, 46.2%). Personal discussion happens 1-4 times per week (18 out of 27 respondents, 66.7%). Internal communication software is used mostly 1-4 times per week (11 out of 15 respondents, 73.3%). Email (18 out of 32 respondents, 56.3%) and telephone (23 out of 33 respondents, 69.7%) are also frequently used 1-4 times per week. Documentation is written, read or managed 1-4 times per week (27 out of 36 respondents, 75%). Project coordination tools are in use 1-4 times per day (36 out of 37

respondents, 97.3%). Training is established 1-4 times per year with 100% respondents' rate. Boards are possibly to be noticed 1-4 times per week (4 out of 6 respondents, 66.7%). In summary, most of the communication channels (7 out of 9) are in use for safety analysis at least every week. Project coordination tools are in use for safety analysis everyday, while training (including tutorial) concerning safety analysis is mostly held every year. In particular, a same number of participants (18 participants) mentions that the meetings are held between 1-4 times per week and 1-4 times per month. We discuss this point with an expert in company A and consider the reason as the distribution of company. For local projects, it is possible to establish necessary meetings. However, the distributed projects occupy also a large percentage in our research context. It is almost impossible for the employees, who are in distributed locations, to find a common time slot, such as in Germany, China, and USA. In these cases, meetings cannot be so frequently held at a 1-4 times per week rate and other communication channels are in use instead.

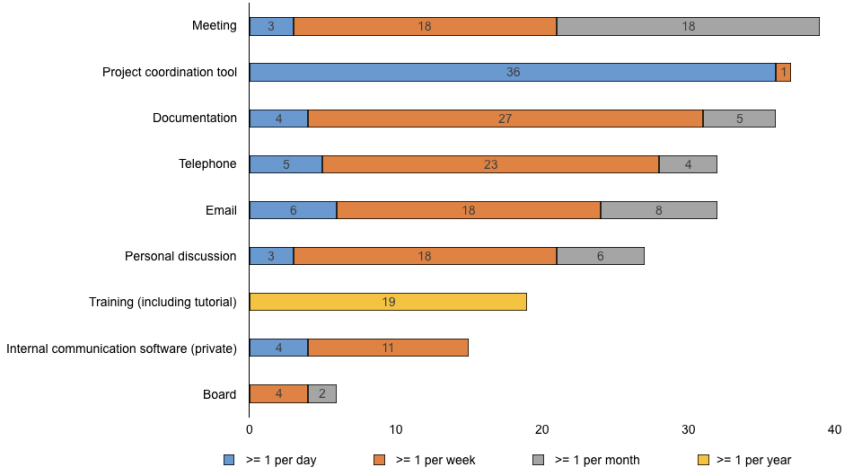


Figure 7.5: Usage Frequencies of the 9 Communication Channel

7.2.5.3 RQ 3: Which are the purposes of safety analysis communication channels?

We address overall 28 purposes of 9 communication channels during safety analysis, as we can see in Table 7.4.

1. Transfer safety requirements (in). To transfer safety requirements internally, the employees use meetings, documentation and project coordination tools. They discuss safety requirements in the form of meetings to keep it formal and possible to be established among multiple functional departments. They record the safety requirements in official documentation and project coordination tools to support daily communication.

2. Transfer safety requirements (ex). To transfer safety requirements externally, meetings and documentation are implemented. However, contrary to the internal transfer of safety requirements, project coordination tools mostly have a limitation on permissions to external members. Some of the customers send the safety requirements per email, which seems to be non-regulated due to a lack of format of constructing safety requirements and an acceptance process of safety requirements.

3. Derive safety requirement. A formal and thorough discussion is necessary to derive correct and complete safety requirements. Thus, meeting is the most effective way with respect to the number of participants and time.

4. Clarify safety requirements internally. Many communication channels are applicable to clarify safety requirements internally, which include meetings, personal discussion, internal communication software (private), telephone, documentation and project coordination tools. It depends on the impact degree and scope of misunderstandings. A meeting is ideal for clarifying safety requirements with a severe impact. Personal discussion, internal communication software (private) and telephone are better for an individual clarification. Documentation is used to record an official clarification, while project coordination tools are specific for a clarification in an open mode.

5. Clarify safety requirement externally. An official and reliable communication channel is extremely important to clarify safety requirements externally. Thus, meetings and documentation are the most appropriate channels. Email

has also been used for an explanation at an early stage.
6. Implement safety requirements. The developers should implement safety requirements to their development. In terms of accountability, they prefer to execute it through meetings and documentation. The generation of ideas may happen through personal discussion.
7. Trace safety requirements (bi). According to the norms, safety requirements should satisfy a bi-directional traceability [Com11b]. Project coordination tools give the employees a direct overview. However, to preserve them long-term, documentation is needed. In addition, some project coordination tools do not support multiple level safety requirements, such as Jira.
8. Planning. To perform safety analysis planning, a planning meeting is a popular way, together with a safety plan as a work product. The execution process is shown in a project coordination tool.
9. Regular discussion. It includes regular meetings in the development team or personal discussion among team members. Sometimes the team members use internal communication software.
10. Demonstrate periodic analysis results. The safety analysis results should be demonstrated periodically to promote development. The results are transmitted automatically in the project coordination tools. The safety analysts demonstrate the results via the meeting and record them in the documentation.
11. Demonstrate periodic V&V results. It is the same way with purposes 10. The acceptance criteria for safety verification and validation are recorded in the documentation and the review process is running in a meeting with a record in the project coordination tools.
12. Review. There is a regular review meeting held at the end of a project. The results are recorded in the review report and in a project coordination tool at the same time.
13. Monitor the status. Some practitioners facilitate communication for monitoring the status of safety analysis. The project coordination tool is the best way for monitoring the status in terms of a convenience and completeness overview.
14. Fix resources/supply problems. Safety analysis sometimes faces the prob-

lems concerning a lack of resources or supply. The resources or suppliers are inside the company, yet among different subsidiaries or multiple functional departments. Meeting and personal discussion are the normal ways to fix such official but less frequent issues.

15. Fix customers' complaints. The employees receive the customers' complaints mostly through emails. Solving the problems is achieved through meetings.

16. Establish commitments and make decisions. The commitments are established among various roles, while the decisions are made in a formally strategic way. Meeting is the most suitable method.

17. Improve processes and techniques. The processes and techniques of safety analysis are continuously developed. No matter how subtle the changes are, these changes have to be considered and discussed systematically and transparently [PGP08]. Meeting can gather omnifarious opinions and perform allocations from an overall perspective.

18. Fix temporary problems, conflicts and obstacles. There are many unforeseeable and even tiny problems, conflicts and obstacles during safety analysis, possible communication channels are meeting, personal discussion, internal communication software, email and telephone. The selection of the channels depends on concrete issues.

19. Cooperate among multiple functional departments. A meeting is possible to be organised across multiple functional departments. The employees might also communicate through personal discussion, when they have a good relationship privately. The employees are also using internal communication software, email and telephone to communication with multiple functional departments.

20. Help to understand the norms. The activities and artifacts of performing safety analysis have to satisfy the norms. Not everyone has a solid background. To understand the norms, safety managers might introduce briefly in a meeting. More than that, the employees could ask relevant experts through personal discussion. To this end, training is popular in safety-critical companies to illustrate the norms systematically.

21. Realise real-time notifications. Some issues related to safety analysis are

emergent, such as a serious customers' complaint, which might influence an ongoing delivery. In this case, the employees usually go directly to find the colleague (personal discussion), using internal communication software or telephone. Some project coordination tools provide a timely notification function in connection with email.

22. Provide feedback and comments. The employees are able to provide a timely feedback or comments through personal discussion, internal communication software, and telephone. Email is better for a clear explanation, while project coordination tools make the feedback and comments open to other employees.

23. Enhance group cohesion. Some communication channels among safety analysis and other stakeholders enhance safety-critical industries' cohesion, such as internal communication software and project coordination tools. Training is especially useful for new employees.

24. Discuss bordered or off topics. The contents of bordered or off topics might influence the safety analysis positively or negatively. The employees mainly use internal communication software or directly personal discussion.

25. Share knowledge. Knowledge sharing is considered important in SCS [NG14], such as general safety knowledge or in-depth safety analysis information. It occurs through meeting, personal discussion, telephone, documentation, training and boards.

26. Transfer documents or links. The safety analysis related documents are mostly saved on a server. Yet, a small change of information management system as well as the complicated hierarchy of folders make some documents more difficult to be found or take more time than asking the responsible colleague. Thus, the employees might use communication channels to transfer them or their link. Internal communication software is a convenient way to share a link or an address, while email is able to send chunky files and keep track of the documents.

27. Enhance safety culture. Safety culture reflects the general attitude and approaches to safety and risk management [Lev11]. Safety culture is difficult to evaluate [Gul00]. Documents are required by the norms. The satisfaction of such requirements of documents determines the organisation's safety

assurance capability. It influences the safety culture. Responsibilities of specific roles, such as safety manager, are important to meeting the norms. Thus, a training of norms is necessary for remaining a good safety culture. Some participants mentioned boards as a good way to enhance safety culture through daily work, such as hanging out recent contributions.

28. Demonstration (external). Boards are also popular to demonstrate the safety analysis capability of the organisations to external experts or customers. An intentional demonstration for a possible cooperation is conducted in meetings.

Table 7.4: Purposes

	Meeting	Personal discussion	Csw.	Email	Tel.	Doc.	Project coordination tool	Training	Boards
1. Transfer safety requirements (in)	✓					✓	✓		
2. Transfer safety requirements (ex)	✓			✓		✓			
3. Derive safety requirements	✓								
4. Clarify safety requirements (in)	✓	✓	✓		✓	✓	✓		
5. Clarify safety requirements (ex)	✓			✓		✓			
6. Implement safety requirements	✓	✓				✓			
7. Trace safety requirements (bi)						✓	✓		
8. Planning	✓					✓	✓		
9. Regular discussion	✓	✓	✓						
10. Demonstrate periodic analysis results	✓					✓	✓		
11. Demonstrate periodic V&V results	✓					✓	✓		
12. Review	✓					✓	✓		
13. Monitor the status							✓		
14. Fix resources/supply problems	✓	✓							
15. Fix customers' complaints	✓			✓					
16. Establish commitments and make decisions	✓								
17. Improve processes or techniques	✓								
18. Fix temp. problems, conflicts and obstacles	✓	✓	✓	✓	✓				
19. Cooperate among multiple functional departments	✓	✓	✓	✓	✓				
20. Help to understand the norms	✓	✓						✓	
21. Realise real-time notifications		✓	✓		✓		✓		
22. Provide feedback and comments		✓	✓	✓	✓		✓		
23. Enhance group cohesion			✓				✓	✓	
24. Discuss bordered or off topics		✓	✓						
25. Share knowledge	✓	✓			✓	✓		✓	✓
26. Transfer documents or links			✓	✓					
27. Enhance safety culture						✓		✓	✓
28. Demonstration (external)	✓								✓

Csw. is internal communication software. Tel. is telephone. Doc. is documentation. V&V is verification and validation. In means internal, while ex means external. Bi means bi-direction.

7.2.5.4 RQ4: Which are the challenges when using safety analysis communication channels?

7.2.5.5 The Top 10 Challenges

We derive the Top 10 challenges across 9 communication channels from interviews and check the results by a safety expert in company A. We list them as follows.

1. The communication of sensitive or confidential information should be monitored.

The results of safety analysis are mostly sensitive and confidential, since they encompass products in details, which influence a safe operation and end-use of products. Communication channels have to keep the information confidential, which includes ensuring that sensitive and confidential information is properly stored, maintained, secured, and accessible to those who need it [Ban+12] [Luc04]. One interviewee mentioned: *"We categorised the data (information) related to safety analysis with different confidentiality levels. Higher confidential data (information) requires relevant authorities to read and transmit. Yet, we do not regulate the verbal communication, including using social communication software to discuss the information. These information is not under monitor and control.*" We note one factor concerning initiative leakage (a lack of regulations) of sensitive or confidential information through communication channels rather than passive leakage (a poor security and privacy assurance mechanism). As one participant mentioned: *"We always avoid some non-regulated channels to transmit confidential and sensitive data (information), such as using e-mail to send safety requirements, even though some customers do this. We actually do not know clearly what is such regulations, just following our experiences*." To face this factor, organisation should establish relevant regulations on the communication channels, especially on the verbal channels, when performing safety analysis. The employees can understand clearly which sensitive and confidential information is able to transmit by which communication channels. Moreover, an alternative to

verbal communication is recommended when performing safety analysis, such as text transmission.

2. Some safety analysis information is fragmented.

The fragmentation will cause misunderstandings on safety analysis information. This happens particularly in text-based communication channels, such as safety analysis related-documentation and internal communication software. As one interviewee mentioned: *"Mostly we can find out the safety-analysis relevant documents on the server. But sometimes the information is difficult to be understood or also possibly lacking some critical information, maybe not updated. The files are uploaded by authors themselves with no more reviews and even regular maintenance.*" There is a specific role responsible for technical documents' maintenance in industries. However, the maintenance for online documents seems weak. Moreover, for the synchronous communication channels like internal communication software. One interviewee said: *"To save time, there are too many personal abbreviations. That makes us confused and influences some emergent issues.*"

3. Some safety analysis information is inconsistent.

The tools such as project coordinate tools and safety analysis tools are updated frequently. However, other recordings do not keep the same space. As one interviewee mentioned: *"In Jira and Doors, which we are using for recording safety requirements, there are all the updated safety requirements. But sometimes we use also doc files to record the results from safety analysis firstly and do not use those directly. The update (of official requirements documents) has a determined time point, which is not in a timely manner.*" A real-time consistency seems important for the set of safety analysis related tools. In this paragraph, we focus on the contents, while the forth challenge focuses on the trigger time.

4. Some communication channels concerning safety analysis are asynchronous, when they should be synchronous.

Some communication channels are considered to be synchronous, such as Skype or telephone. However, they are not in our case. One interviewee mentioned: *"When there is a temporary meeting, I will check if this colleague is online (Skype), then call him (or her). It works mostly. But once, I got an emergent customer complaint regarding our safety analysis, we need to find a colleague immediately, he was not online and could not be reached by telephone. I decided to go directly to his office and found him.*" The asynchronous channels will hinder some emergent issues such as production and delivery and might cause fatalities.

5. The communication channels concerning safety analysis lack tool support.

Tools often put a limit to the strategies that can be adopted [Lev11]. For effective communications, tools are important, especially the interfaces among different tools. When performing safety analysis, non-verbal communication takes an increasingly major part of it. An effective tool chain ensures the information's correctness, clearness, completeness and synchronous. However, the integration between safety analysis and functional development still lacks tool supports. Various organisational structures have various communication channels and tools. The new techniques are changing too fast to arrange an immediately effective use of them. As one interviewee said: *"We start to use ALM (from IBM) for project management, but the interfaces (with the existing management tools) are not finished. The traceability is too poor.*" The company has to consider an immediately developed interface with other safety analysis tools, when they introduce a new project management tool. Otherwise, when buying a new tool, a feasible interface to the existing toolchain should be considered by the compnay. As one interviewee mentioned: *"APIS IQ-FMEA is a powerful tool that we used for performing FMEA. We have used it for many years and believe in it. Currently, we are considering*

to use a safety information and activity management tool. A lot of companies have introduced their tools. But our first criteria is if it has a feasible interface with APIS IQ-FMEA."

6. Developers might misunderstand the information from safety experts.

When the products to be developed have novel functions, there will be misunderstandings between functional developers and safety experts. One interviewee said: *"It is no problem when developing old functions like ECU (electronic control unit). But these years we are starting a new functional module NCU (new energy control unit). The traditional safety experts, who have functional development experiences only on developing ECU, lack the new knowledge to understand detailed development.*" The communication channels which support knowledge sharing might be helpful in this case.

7. There are language, geographic and culture barriers in the communication channels.

In terms of language, most of the companies in our context use English as the official language. Nevertheless, some professional terms concerning safety analysis are not uniform, such as incorrect verbs like *operate safety analysis* instead of *perform safety analysis*. This happens in the text-based as well as verbal communication channels. In terms of geographic, distributed locations interfere experience sharing. As one interviewee mentioned: *"We send our colleagues, who perform safety analysis to other countries once a year to enhance communication and collaboration, but it works better for new employees. They want to learn new techniques and former experiences. For safety experts, we prefer that the execution of safety analysis should be suitable for our own environment or context. Different countries have different requirements and problems.*" The basic knowledge has been well transmitted. However, the practical experiences are less shared and worse discussed. The knowledge sharing of safety analysis should not stop in the technique level, rather with more real-life cases. In terms of culture, distributed companies keep detailed

safety analysis results and avoid to communicate them. There is a lack of trust among different cultures [Hol+06]. The parent company keeps some details, such as architecture design, of development and even safety analysis. When the subsidiaries inherit some function modules, they lack such details to perform a thorough and systematic safety analysis. Regulations among different countries might be a reason.

8. The members from functional departments are unwilling to share safety knowledge with non-functional departments.

Functional (including functional safety) departments believe in their knowledge on development and safety of products. As one expert from the functional department said: *"We don't see the necessity to do this.*" We call this stereotype in groupthink theory[1]. This challenge exists during safety analysis and is specifically important for developing and ensuring a safe product [WW18b]. An interviewee said: *"We do not think that other departments like purchasing or sales could help us a lot on developing a safe product. We know more and detailed on the product. For one project, we were required to train them with some general safety knowledge. We do not know how much it works. They also look unwilling.*" In terms of safety knowledge sharing, the functional departments (including functional safety) should appreciate the needs and expectation of other job roles. In addition, a lack of continuous learning and personal development processes reduce the passions of non-functional departments' employees to learn safety-analysis related knowledge, as one interviewee mentioned: *"Safety analysis is not my main job, if just for this project and it does not help for future, we actually do not want to use too much time on it.*" To this end, we should notice if the information is spread widely enough. As one expert said: *"Sometimes the employees are forced to share the safety knowledge.*" A lack of initiative makes the knowledge sharing becoming

[1] Groupthink is a psychological phenomenon, which was introduced by Janis in 1971 [Jan71]. It is a linear model of how seven antecedents (cohesion, group insulation, impartial leader, lack of norms, homogeneous, high stress from external threats, temporarily low self-esteem) increase the likelihood of premature concurrence seeking, which leads to eight psychological symptoms (illusion of invulnerability, collective rationalisations, belief in inherent morality of the group, stereotypes of out-groups, direct pressure on dissenters, self-censorship, illusion of unanimity, self-appointed mindguards).

superficial.

9. The storage, authority, regulation and monitoring problems on the transmitted safety analysis information.

In terms of the storage of safety analysis information, one interviewee mentioned: *"We have a central information management system on the server, but the safety analysis data (information) or files are not clearly classified. Some (of them) are mixed with other process documents.*" The quality of transmitted information interleaves with the effectiveness of communication channels. The company should consider to establish a safety information management system to be separated with other information. In addition, the tools of information storage are important. An expert said: *"The process of our data (information) storage is following the tools, not using tools to support process.*" Thus, a suitable tool for safety information storage is needed, especially for modern SCS with a huge amount of data. As one interviewee mentioned: *"The simulation data for autonomous driving system are thousands of giga bytes. It is relative difficult to find them when there is a migration happened 10 years before.*" In terms of authority, one interviewee said: *"In the information management system, we classify the files into different levels of authorities. But it is difficult to divide levels of safety requirements in Jira.*" The authority should be noticed not only in the information management system, but also in the project management tool. In terms of regulation, some feedback are: *"We are not clear about the regulations, just big sizes of files cannot be sent privately.*" We notice that there are some hidden regulations on transmitting safety sensitive information. However, these regulations are not clearly and obviously announced. In terms of monitoring problems, the safety analysis information are monitored by an IT department as the same with other company level sensitive information. As one interviewee mentioned: *"We do have a monitor group that monitors and controls the transformation of marked files. The high severity files have different kinds of mark. It could be traced by IT department.*" The monitor of safety analysis information does work inside the company, but still needs enhancement.

10. There is a lack of technical documentation to support communication.

Safety analysis should be performed at the system-level [Lev11]. The detailed system description documents are necessary but sometimes not available. One interviewee mentioned: *"Some products are developed on the basis of the original product. The functions are inherited. The detailed architecture design documents are kept by the original company or department. The execution of safety analysis cannot only be performed on new function modules without considering interfaces with original products. A systematic impact causes risks."* This might be caused by the culture (as we indicate in challenge 7), the authority problem (as we mentioned in challenge 9) of subsidiaries, or an incomplete and fragment record of safety analysis information (as we mention in challenge 2). Nevertheless, safety analysis related documentation should consider communication [WBW17].

7.2.5.6 A Mapping between Purposes and Challenges

We map the purposes with challenges, as shown in Table 7.5. The first line depicts the numbers of the Top 10 challenges, while the first column depicts the purposes.

Considering challenge 1: *The communication of sensitive or confidential information should be monitored.* It happens often when the practitioners transfer and demonstrate safety analysis related information (purpose 1, 2, 24, 26, 28). The practitioners, who use the communication channels for achieving these purposes, should notice challenge 1. Other communication channels are better to avoid transferring sensitive or confidential information.

Considering challenge 2: *Some safety analysis information is fragmented.* The incomplete and fragment information occur through almost all the 9 communication channels, especially for demonstrating information (purpose 10, 11). Documentation, project coordination tools, email and internal communication software should keep safety analysis related information as complete as possible. When facing resource or supply problems, the provided

Table 7.5: Purposes versus Challenges

Purposes vs. Challenges	1	2	3	4	5	6	7	8	9	10
1. Transfer safety requirements (in)	✓								✓	
2. Transfer safety requirements (ex)	✓								✓	
3. Derive safety requirements			✓			✓				✓
4. Clarify safety requirements (in)			✓			✓				
5. Clarify safety requirements (ex)			✓				✓	✓		
6. Implement safety requirements						✓				✓
7. Trace safety requirements (bi)					✓					
8. Planning								✓		
9. Regular discussion						✓				
10. Demonstrate periodic analysis results		✓	✓							
11. Demonstrate periodic V&V results		✓	✓							
12. Review						✓				
13. Monitor the status				✓	✓					
14. Fix resources / supply problems		✓								
15. Fix customers' complaints				✓			✓			
16. Establish commitments and make decisions									✓	✓
17. Improve processes or techniques								✓		
18. Fix temp. problems, conflicts and obstacles					✓					
19. Cooperate among multiple functional departments								✓		✓
20. Help to understand the norms						✓			✓	
21. Realise real-time notification				✓						
22. Provide feedback and comments					✓				✓	
23. Enhance group cohesion							✓			
24. Discuss boards line or off topics	✓									
25. Share knowledge						✓	✓			
26. Transfer documents or links	✓								✓	
27. Enhance safety culture								✓		
28. Demonstration (external)	✓									

In. csw (private) is internal communication software. Tel. is telephone. Doc. is documentation. V&V is verification and validation. In means internal, while ex means external.

information from them should notice completeness (purpose 14).

Considering challenge 3: *Some safety analysis information is inconsistent.* When the practitioners clarify the safety requirements (purpose 3, 4, 5) or demonstrate the results (purpose 10, 11), the communication channels should keep consistent information to avoid misunderstandings. Thus, the consistency among the recordings of meetings, the archived documentation and the project coordination tools seem important.

Considering challenge 4: *Some communication channels concerning safety analysis are asynchronous, when they should be synchronous.* It is extremely important when there is a need for real-time notification and timely monitoring (purpose 13, 15, 22). Thus, telephone, internal communication software should be able to reach a specific person when he/she is on-site.

Considering challenge 5: *The communication channels concerning safety analysis lack tool support.* A tool is necessary for tracing safety requirements (purpose 7), monitoring status (purpose 13) and providing feedback (purpose 22). The temporary problems necessitate tools to support recordings

(purpose 18). We should arrange or design efficient tools and their interfaces as well.

Considering challenge 6: *Developers might misunderstand the information from safety experts.* The misunderstanding might occur between developers and safety experts when they use communication channels internally to derive, clarify, implement, discuss, review and share the safety analysis related information (purpose 3, 4, 6, 9, 12, 25). The understanding with respect to norms show also bias (purpose 20). The practitioners need to consider the understandability and an obstacle-free communication among multiple functional departments.

Considering challenge 7: *There are geographic, language and culture barriers in the communication channels.* When the industries aim to enhance group cohesion and share safety knowledge (purpose 24, 26), the communication channels, such as internal communication software, training and boards, might reduce the interferences from geographic, language and culture bias. To face customers or clarify safety analysis externally, it might happen in a multiple geographical distribution (purpose 5, 15). The relevant communication channels, such as meeting, should consider it.

Considering challenge 8: *The members from functional departments are unwilling to share safety knowledge with non-functional departments.* The communication channels regarding the cooperation among multiple functional departments (purpose 20) should avoid this challenge. For example, during meetings, a friendly environment for discussion is necessary. Personal discussion, such as by using internal communication software, email and telephone, is encouraged among multiple functional departments. When clarifying safety requirements externally (purpose 5), this challenge may happen due to a diverse knowledge background. When planning safety analysis (purpose 8), non-functional departments are keeping silence, as well as when improving processes or techniques (purpose 17). To enhance safety culture (purpose 27), non-functional departments should be included.

Considering challenge 9: *The storage, authority, regulation and monitoring problems on the transmitted safety analysis information.* To transfer safety requirements (purpose 1, 2, 26), storage, authority, regulation and monitor-

ing are necessary. To establish commitment and make decisions (purpose 16), relevant regulations are needed. The fulfillment of norms should follow regulations (purpose 20). The feedback and comments need to be stored (purpose 22).
Considering challenge 10: *There is a lack of technical documentation to support communication.* The derive, implementation and decision making of safety requirements need a clear documentation to be understood by multiple functional departments and support an effective communication among them (purpose 3, 6, 16, 19).

7.2.6 Discussion

The main benefit of our article is that we investigate the general topic concerning communication channels in a concrete context concerning the execution of safety analysis. We aim to arouse the awareness of practitioners concerning safety analysis on their communication. The results are multiple: (1) we find 9 communication channels during safety analysis; (2) most of them happens 1-4 times per week; (3) we investigate 28 purposes of these 9 communication channels and (4) the Top 10 challenges across these 9 communication channels.

To compare with the related works, Storey et al. [Sto+17] found around twenty communication channels during software development. When performing safety analysis, the number of communication channels are smaller. We conjecture that (1) safety analysis is an activity within software development, (2) the confidentiality concerning sensitive safety-related data limit the number of possible using communication channels, especially informal communication channels. Kraut et al. [Kra+90] mentioned that the usage frequency can show the importance of communication. In this article, we calculate the usage frequencies of the 9 communication channels during safety analysis to raise the importance of communication during safety analysis. Vilela et al. [Vil+17] proposed more than 11 pros that an effective communication in safety analysis can bring, such as reducing errors in requirements specification or helping design. However, to achieve an effective

communication, the realisation of communication purposes is first and foremost. The existing research concerning communication purposes seems poor. We conjecture the reason might be that the general communication purposes distribute too wide-ranging. However, during safety analysis, the communication purposes can be accounted. The challenges of communication concerning safety issues were mentioned in a plenty of studies, which are either as accident reports [DMN14] or in terms of general organisational management [HRH13]. The summarised communication challenges in software development are more than twenty [Sto+17]. However, to the best of our knowledge, little research focuses on the communication challenges, which might cause unsafe issues, during the execution of safety analysis in organisation management.

"Being a good communicator is one thing. Knowing what to communicate is much more important." [Con+11] (P. 52, Conboy et al.)

To see the results in this article, given the aforementioned opinion, we highlight the major contribution of our results as communication purposes **(RQ 3)**. During our research, a lot of the popular communication channels are used across various areas, not only for safety analysis. The challenges are summarised with similar manifestations, such as incomplete information and asynchronous implementation, but they are totally different in essence in terms of causalities, effects as well as solutions. Some of the challenges cannot map into safety analysis directly. We believe that a purpose during the communication in safety analysis determines why and what to communicate. Communication makes sense when the people achieve their communication purpose.

Thus, in this article, first, we map the 9 communication channels (RQ 1) with the 28 purposes. Within these 28 purposes, each one has 1 to 6 possible selections of communication channels to achieve it. For instance, to derive safety requirements (purpose 3), to establish commitments and

make decisions (purpose 16) and to improve processes and techniques (purpose 17), the practitioners show only the use of meetings. To clarify safety requirements internally (purpose 4) and to share knowledge (purpose 25), each purpose has 6 communication channels as possible selections in practice. Second, we map the Top 10 challenges (RQ 4) with their 28 purposes. The challenges rely heavily on their purposes and have different manifestations on each channel. For instance, “the communication of sensitive and confidential information should be monitored" (challenge 1) when “transfer safety requirements" (purpose 1, 2), as shown in Table 7.5, rather than whether the practitioners use meetings, email, documentation or project coordination tools, as shown in Table 7.4.

In addition to the major contribution on communication purposes, we demonstrate firstly the state-of-the-art in terms of the types **(RQ 1)** of the existing communication channels. The number is not huge comparing with existing communication channels in other areas such as social media [Sto+10].

Meeting is a dominant one during safety analysis, which is able to achieve 20 out of 28 purposes. Especially, three purposes (3, 16, 17) can only be reached by meetings. The practitioners prefer to use meetings in traditional development processes which advocate formality as well as in modern development processes which aim to increase cooperation and communication. Meetings are possible for them both.

Project coordination tool, *documentation*, *telephone* and *email* are four popular communication channels in organisation management, as well as when performing safety analysis. They are able to reach 11 purposes, 12 purposes, 6 purposes and 7 purposes in safety analysis, respectively.

The importance of *personal discussion* is an outstanding result in our study. 71.8% participants mentioned that they are using it during safety analysis. It can achieve 11 out of 28 purposes. Based on our original conjecture, personal discussion might happen occasionally and especially rarely during safety analysis. Safety analysis is considered to be a technical in-depth activity, which needs an official communication or at least, more time to prepare. Comparing with other 8 communication channels, personal discussion seems to be the most unregulated one, which is extremely difficult to control,

monitor and trace. We missed noticing its necessity. Yet, the results caught our attention.

Internal communication software is a novel channel since last decade. Even though it has been moderately mentioned in our context, the results show still its powerful functionalities that it can reach 9 purposes during safety analysis.

Training is a concentrated way to enhance communication concerning safety analysis. It can reach 4 purposes. Apart from the basic training for new employees, some specific training programs, or even expert-level training, should be arranged.

Boards are familiar for industries. It can achieve 3 purposes. Due to the modern development processes, such as whiteboards [Che+07] from Kanban, the practitioners could consider extending the boards' functionalities, particularly during safety analysis.

To strength the existing communication channels further, the usage frequencies (**RQ 2**) in practice can demonstrate their importances. We set 4 scales including 1-4 times per day, 1-4 times, per week, 1-4 times per month and 1-4 times per year. Generally, 7 out of 9 communication channels are used 1-4 times per week. Project coordination tools are used 1-4 times per day. We discussed it with a safety expert and he said: *"To start an (safety analysis) activity, we check the state as the first step, because we just switched from other tasks. Project coordination tool (Jira) is convenient, complete and intuitive.*" A general overview is necessary before facilitating an issue. The project coordination tool is easy to use and it provides almost all the information or ways to get these information in a project. Training concerning safety analysis happens 1-4 times per year. From our viewpoint, it is reasonable.

As an initial step to investigate communication channels during safety analysis, we concentrate on the Top 10 challenges (**RQ 4**) based on analysing our qualitative data and a final check by an expert in company A. Several of them show similar problems as in general communication channels, such as incomplete and fragmented information. Yet, during safety analysis, we notice that the fragmentation of information are in high severities and may cause fatalities. The completeness is not dominant. As one expert mentioned:

" 90% (incomplete) information sometimes is fully sufficient for solving the problems during safety analysis. But when the information is complete but fragmented, such as safety cases, which are spread over different tools, that seems serious." Geographical problems are also popular in general communication channels. Developers cannot get in touch to exchange states. However, during safety analysis, it influences not only the information transmission, but also safety knowledge sharing. These challenges need separate considerations. Other challenges are specifically existing in the communication channels during safety analysis, such as confidentiality concerning safety analysis information and groupthink including unwillingness to share safety analysis information with non-technical members.

For *theory*, RQ 1, RQ 2 and RQ 3 might provide implications, since as far as we know, there are still no reported results on the existing communication channels as well as their usage frequencies and purposes during safety analysis. The communication in safety-critical industries should arouse attentions. We believe that in developing SCS, the practitioners are not keen on enriching the amount of channels, rather, solving the challenges on the existing communication channels to ensure a safe development process and a safe product. Thus, the 9 communication channels during safety analysis are convincing. Most of the communication channels are frequently used per week. In addition, RQ 3 provides 28 purposes of communication during safety analysis. It seems to be a rich result to draw researchers' attentions. In the future, we expect that the researchers could expand this study to more safety-critical companies, domains and countries to check and enrich our initial results on the number of communication channels, their usage frequencies and purposes. Depending on the sizes of companies, there might be various answers.

For *practitioners*, RQ 4 might arouse more interests. Practitioners in safety-critical industries may have these problems in their running projects. However, we do not provide any solutions to each specific challenge in this article, since we believe that the contexts and tools of each communication channel are changing, the challenges may remain challenges in the future but with different manifestations. The solutions should be derived in a specific

way. Moreover, the mappings between RQ 1 and RQ 3 as well as between RQ 3 and RQ 4 seem practically useful. For achieving the communication purposes during safety analysis, the practitioners can select the possible channels from Table 7.4 and know the possibly relevant challenges from Table 7.5. In the future, we expect that the practitioners could propose more challenges that they have met as well as solutions, either in a general way or specifically in a context.

7.2.7 Limitations

We believe that three major limitations threaten the results of our study.

Our sample might not cover all possible roles during safety analysis; the results could be biased by an over-representation of medium sized companies, and we could not cover all possible domains where safety analysis is critical (e.g., aerospace). Finally, the sample could cover only companies from three countries. We believe, however, that the sample is rich and meets a high norm for qualitative case studies. Our sample of 60 participants covers all possible company sizes (small, medium, and large), offers data from two geographically and culturally diverse zones (Germany/Italy and China), and considers three different domains of interest (automotive, medical, and industry 4.0). For a stronger generalization of our results, we call for survey studies between companies to *go wide* where we could *go deep* within companies.

Communication occurs frequently and often spontaneously, so it is challenging to observe it directly. In our case study, while we could perform direct observation sessions, we mostly collected perceptions and experiences of participants. Memory recalling and other cognitive biases could over or under-represent certain communication channels and their usage frequency. This is a typical issue of observation studies, questionnaires, and interview-based designs [Cre09] that we wish to recall nonetheless.

Furthermore, causality chains (e.g., those in RQ 3 and RQ 4) are harder to empirically demonstrate as there was no controlled experiment testing the claims. There is an open debate on whether causality can be inferred

from research approaches other than controlled experiments [Cre09; DN02; GL13]. We agree with several authors, e.g., Gläser et al. [GL13], take the stance that qualitative data analysis can be used to infer causality from the experience of participants, provided that there is a strong methodology for data gathering and analysis. We claim that our methodology is robust. We have employed data triangulation validation whenever possible, for example by adding a three weeks long direct observation to validate the results and by offering our results to a senior functional safety expert from company A to check. Still, given the lack of prior literature, we deem our study to be exploratory in its nature, not confirmatory. We call for future research to empirically demonstrate the relationship chains that we uncovered.

7.3 Mapping Communication in S-Scrum

Table 7.6: A Comparison of Reached Communication Purposes from *Meeting* and *Documentation*

	Meetings	Meetings in S-Scrum	Documentation	Documentation in S-Scrum
1. Transfer safety requirements (in)	✓	✓	✓	✓
2. Transfer safety requirements (ex)	✓	✓	✓	✓
3. Derive safety requirements	✓	✓		✓
4. Clarify safety requirements (in)	✓	✓	✓	✓
5. Clarify safety requirements (ex)	✓	✓	✓	✓
6. Implement safety requirements	✓		✓	
7. Trace safety requirements (bi)			✓	✓
8. Planning	✓	✓	✓	✓
9. Regular discussion	✓	✓		✓
10. Demonstrate periodic analysis results	✓	✓	✓	✓
11. Demonstrate periodic V&V results	✓	✓	✓	✓
12. Review	✓	✓	✓	✓
13. Monitor the status		✓		✓
14. Fix resources/supply problems	✓	✓		
15. Fix customers' complaints	✓	✓		
16. Establish commitments and make decisions	✓	✓		✓
17. Improve processes or techniques	✓	✓		
18. Fix temp. problems, conflicts and obstacles	✓	✓		
19. Cooperate among multiple functional departments	✓	✓		✓
20. Help to understand the norms	✓	✓		
21. Realise real-time notifications				
22. Provide feedback and comments		✓		✓
23. Enhance group cohesion		✓		✓
24. Discuss bordered or off topics				
25. Share knowledge	✓	✓	✓	✓
26. Transfer documents or links				
27. Enhance safety culture		✓	✓	✓
28. Demonstration (external)	✓	✓		✓

To map the results of communication channels in our research context to S-Scrum, the other seven communication channels, namely personal discus-

sion, communication software, email, telephone, project coordination tool, training and boards, show few differences regarding their usage frequencies, purposes and challenges.

S-Scrum differs itself strongly from meetings and documentation. S-Scrum, as an approach extending from Scrum, promotes more meetings than traditional development processes, while S-Scrum adds the value of communication on documentation to fulfill agile principles. Thus, we consider that the *meeting* and *documentation* in S-Scrum can satisfy different communication purposes. The possible communication channels and challenges can be further derived from Table 7.4 and Table 7.5.

As we can see in Table 7.6, the *meetings in S-Scrum* can satisfy almost all the purposes of the meetings in our research context, aside from the one "implement safety requirements" (purpose 6). The development team and safety analysts discuss the implementation of safety requirements throughout the development rather than establishing a meeting. Therefore, although *meetings in S-Scrum*, as a communication channel, fails to reach purpose 6, which can be realised throughout the daily communication.

However, *meetings in S-Scrum* can achieve more communication purposes than meetings in our research context, which are "provide feedback and comments" (purpose 22), "enhance group cohesion" (purpose 23) and "enhance safety culture" (purpose 27). The short iterations in S-Scrum provide a regular time for feedback and comments. The "daily scrum meeting" gives the developers chances to provide feedback and comments on each working day, while the "regular safety meeting" deems on the feedback and comments concerning safety-related issues. Group cohesion is enhanced through the continuous daily scrum meetings. Safety culture is integrated into each meeting in S-Scrum, such as in pre-planning meeting or sprint planning meeting, we design the safety plan.

The *documentation in S-Scrum* also fails to achieve purpose 6 "implement safety requirements". The reason remains the same that most of the implementation of safety requirements are facilitated during development in S-Scrum through discussion without documentation.

However, the *documentation in S-Scrum* can achieve more communica-

tion purposes than the traditional documentation, which are "derive safety requirements" (purpose 3), "monitor the status" (purpose 13), "establish commitments and make decisions" (purpose 16), "cooperate among multiple functional departments" (purpose 19), "provide feedback and comments" (purpose 22), "enhance group cohesion" (purpose 23) and "demonstration (external)" (purpose 28). STPA safety analysis is a cooperative activity in S-Scrum. We make the execution processes as well as the results as transparent as possible. The derivation of safety requirements (purpose 3) is recorded in an STPA safety report. The documentation, such as a story map or sometimes product backlogs, can be open in the working place to monitor the status (purpose 13). The documentation in S-Scrum advocates openness, transparentness and a support of communication. Thus, no matter the internal or external stakeholders or the members in multiple functional departments are possible to discuss the contents in the documentation of S-Scrum. The discussion can achieve purpose 16, purpose 19, purpose 22, purpose 23 and purpose 28.

7.4 Conclusion

In this chapter, we solve the problems concerning "communication" during safety analysis in the preliminary S-Scrum. We firstly conducted an industrial exploratory case study in 7 safety-critical companies with overall 60 participants to investigate the communication channels during the existing safety analysis. We used surveys, interviews, documentation review and participant observation in a large automotive company (with three medium subsidiaries), a large medical equipment company, a medium automotive company and a small industrial 4.0 based production line company. During three rounds of data collection, we found 9 communication channels during safety analysis. Most of them happen 1-4 times per week. We summarised 28 purposes of these 9 communication channels and the Top 10 challenges across these 9 communication channels. We highlight the importance of communication purposes and map them with the existing channels and

challenges.

S-Scrum differs itself from our research context mainly on two communication channels, which are *meetings* and *documentation*. We investigate them in S-Scrum further and find that they can achieve 3 more purposes and 7 more purposes, respectively. Yet, two channels fail to achieve 1 purpose of "implementing safety requirements", which is realised through daily communication in S-Scrum.

Communication is important during safety analysis. S-Scrum shows an absolute advantage on supporting communication through achieving more communication purposes based on the existing communication channels, in particular, meetings and documentation.

Chapter 8

Groupthink in S-Scrum

*In this chapter, to face one of the four challenges (verification, planning, communication and requirements prioritisation) in preliminary S-Scrum concerning "**requirements prioritisation**", we improve group work through avoiding groupthink[1] in the preliminary S-Scrum.*

[1] The definition is shown in Section 2.4.4

The main contributions of this chapter are:

- **We investigate groupthink during safety analysis.** *To face the challenges concerning "requirements prioritisation", together with general group work in the preliminary S-Scrum, we investigate groupthink and provide possible solutions to avoid groupthink.*
- **We provide the Top 10 phenomena of groupthink in seven GSA.** *The Top 10 phenomena of groupthink do exist throughout the seven GSA and negatively influence the execution and results of GSA. We derive them from 17 interviews as well as 39 surveys.*
- **We investigate possible reasons for the Top 10 phenomena.** *We investigate reasons referring to Janis's seven antecedents from 17 interviews. Six of seven antecedents have been found in our study.*
- **We propose solutions on how to handle the Top 10 phenomena in SCS management.** *We propose solutions referring to Janis's nine recommendations from 17 interviews. Eight of nine recommendations were mentioned in our study, together with two additional recommendations concerning communication channels and regulations of procedures.*
- **We map the Top 10 phenomena in S-Scrum and propose solutions.** *We find the Top 10 phenomena in S-Scrum and propose solutions, such as "second-chance meeting" or "cross-functional meeting", to be generated into our final S-Scrum.*

8.1 Case Study

We conduct a case study following the guideline from Runeson et al. [RH09] and Yin [Yin13].

8.1.1 Context

We conduct a case study in the same companies as chapter 7 but with different number of participants. The context is shown in Figure 7.1. In this study, we have 56 participants (17 participants took part in the interviews), who contribute to this chapter, as we can see in Figure 8.1. The main participants are from the quality assurance area. They are quality managers to guide the safety analysis. More than half of the members are from a functional safety department. However, other roles like developer, analyst and leader, as well as the members in the area of sales, purchasing and production are also included.

Figure 8.1: Participants

8.1.2 Research Question

The research goal is to investigate groupthink in GSA. We formulate four research questions to steer the design of our study, as shown in Table 8.1.

Table 8.1: Research Questions

RQ 1	What are the current practices in GSA?
RQ 2	What are the symptoms of groupthink during GSA?
RQ 3	How could these symptoms occur?
RQ 4	How to handle groupthink during GSA?

8.1.3 Data Collection

We conducted three rounds of data collection, which took a period of three months involving the conduction of surveys, semi-structured interviews, direct observations and documentation review.

Before data collection, we pre-interviewed five experts from four companies by telephone to decide on a common objective, establish agreements and help designing the survey. Each interview lasted one hour. These five experts were further arranged to be the representative of each company.

In the first round, we used a survey[1] to collect both qualitative and quantitative data. The qualitative data cover the participant's background and the description of phenomena. The quantitative data cover the participant's background and the frequency of the occurrence of phenomena in GSA. The survey was running for two months. We sent the link to each representative via email, as well as the survey in electronic version to ensure that all the participants are able to receive them. During these two months, the first author checked the progress every two weeks through communicating with the representatives by videophone. The content included the distribution and the respondent's rate from the surveys, as well as the problems and feedback.

[1] https://zenodo.org/record/885231#.We2tEFuCyUk

In the second round, we used semi-structured interviews to investigate the in-depth reasons and solutions. We selected the subjects based on the results from the first round survey.

Meanwhile, the first author reviewed a wide range of documents including R&D process development instrument, product development and process development quality management procedure, quality management handbook, R&D risk management instruction, FMEA/FTA guideline/working instruction, decision analysis and resolution in quality management and technical review. Other informal documents, such as screen-shot of the project management tool and message items, such as email or internal apps, were also reviewed.

The first author conducted also a direct observation in the safety analysis process and team meetings in company A. Beside the regular processes, the first author observed the verbal communication temporarily, as well as the members' field notes. In the third round, we designed a final interview with the specific experts to perform the validation of the solutions.

8.1.4 Data Analysis

We use qualitative data for RQ 1, RQ 2, RQ 3, and RQ 4. We analyse the data by using the coding approach of Grounded Theory [SC97], since it is especially appropriate for investigating human aspects of software engineering. We use open coding to record the transcript line-by-line. After that, we use selective coding to choose the code related to groupthink. Finally, we use axial coding to relate the symptoms, reasons and solutions.

We use quantitative data for RQ 2 to show the occurrence of each groupthink symptom in each GSA in Figure 8.2. In the survey, the participants could select the symptom that they noticed in the seven GSA and give the number of frequency ("1", "2" and "3" mean "low", "medium" and "high"). We use the Shapiro-Wilk test for a normal distribution. Since the data are all non-normally distributed, we use Pearson's r to show the degree of correlation between each symptom and GSA, as shown in Table 8.2. To this end, we summarize the most frequent symptoms in each GSA, from which we

investigate further to generate the Top 10 phenomena in Table 8.3.

8.1.5 Results

8.1.5.1 RQ 1: What are the current practices in GSA?

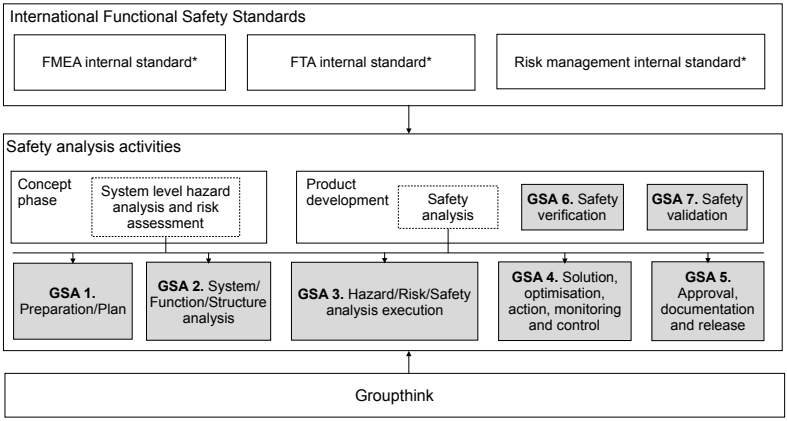


Figure 8.2: Group Safety Analysis

As we can see in Figure 8.2, we reviewed the norms like ISO 26262, ISO 14971 and IEC 60601, as well as FMEA, FTA and risk management internal norms. We divide the safety analysis activities to seven general GSA. We reviewed these seven GSA together with our industrial partners from the primary company A. Below, we introduce these GSA in detail:

1. Hazard/Risk/Safety analysis preparation and plan. This is the first step in safety analysis including making technical decisions and a safety analysis schedule. The processes and results of safety analysis must be proven for all the products and the production processes. Project manager should decide the moderator, contract and plan. The safety analysis plan (including updated plan) should be created with a specific tool (at least an English

version). Cross-functional departments should join this step. The safety analysis might include customers, a general negotiation has to be conducted.

2. System/Function/Structure analysis. This is the second step in safety analysis including determining the scope of the system (system limits, interfaces and boundary conditions), specifying the system (random) failures and type of the analysis for the systems (both qualitative and quantitative). The system determinations and boundary conditions shall be acknowledged by the customer.

3. Hazard/Risk/Safety analysis execution. This is the primary step of safety analysis. Depending on the different techniques, the execution is different. The most popular techniques are FTA, FMEA and HAZOP. FTA is a top-down method, which starts from identifying the top events and goes deeper to find causalities. The safety analyst draws the fault trees. FMEA is a bottom-up method, which finds and evaluates failure concerning the severity (S), probability of occurrence (O) and probability of detection (D). The safety analyst records the failures and effects, as well as measures the risks to provide solutions. HAZOP is a systematic risk analysis method, which identifies, assesses and manages risks at a system-level. It happens mostly at the beginning of the projects and is combined with FMEA or FTA for analysing the systems.

4. Solution, optimisation, action, monitoring and control. This is the forth step in safety analysis. Following the investigation of causal factors including qualitative and quantitative interpretation, a decision can be made, which could be a solution for development, an action for operation or the monitoring and control for a risk. The moderator provides support in the interpretation.

5. Approval, documentation and release. This is the fifth step in safety analysis. The safety analysis execution process and results must be documented. The diagrams, tables and interpretations are necessary to be understood and evaluated. The results are diverse, such as the depth and scope of safety

analysis. The documents should focus on providing recommendations rather than requirements. The structure of a safety analysis report includes system overview, participants, detailed results, descriptions, list of assumptions, sources and attachments. The release has to be determined by customers to ensure that it includes no sensitive data.

6. Safety verification. After the execution of safety analysis. The requirements are to be verified. The purpose of the verification is to demonstrate that the embedded software satisfies its requirements in the target environment [Sta11]. In the design phases, verification is the evaluation of safety requirements from safety analysis.

7. Safety validation. After the safety verification, the requirements need a validation at the system-level. The first purpose of the safety validation is to provide evidence of compliance with the safety goals and that the functional safety concepts are appropriate for the functional safety of the item, while the second purpose is to provide evidence that the safety goals are correct, complete and fully achieved.

8.1.5.2 RQ 2: What are the symptoms of groupthink during GSA?

This section has both quantitative data and qualitative data.
Janis's groupthink symptoms in safety analysis. Groupthink has eight symptoms. Yet, different GSA encompasses different symptoms, as shown in Figure 8.3. We list GSA 1 to GSA 7 on the left side and eight symptoms on the right side. The widths of the lines show the occurrence of each symptom. The thicker lines mean that the symptoms occur more in the GSA. In GSA 1 (hazard/risk/safety analysis plan and preparation), 56.4% of the participants mentioned the "invulnerability", while 56.4% of the participants mentioned "direct pressure". In GSA 2 (system/function/structure analysis), 41% of the participants saw the "stereotypes", while 35.9% of the participants saw the "morality". In GSA 3 (safety analysis execution), 59% of the participants found the "self-censorship", while 56.4% of the participants found the "una-

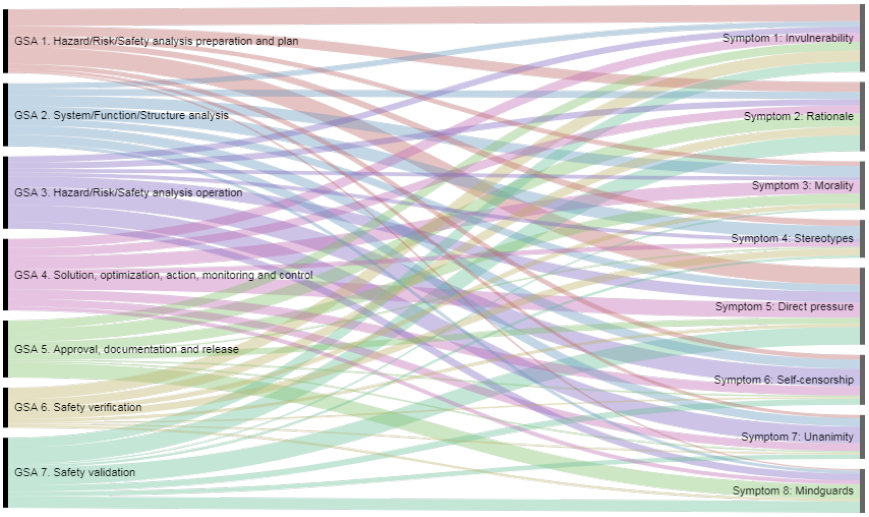


Figure 8.3: Janis's Groupthink Symptoms in GSA

nimity". In GSA 4 (solution, optimisation, action, monitoring and control), 51.3% of the participants mentioned the "direct pressure", while 43.6% of the participants mentioned the "morality". In GSA 5 (approval, documentation and release), 48.7% of the found the symptom "mindsguard", while 43.6% of the mentioned about the "rationale". In GSA 6 (safety verification), 38.5% of the participants saw the "invulnerability", while 28.2% of the participants saw the "rationale". In GSA 7 (safety validation), 59% of the participants found the "direct pressure", while 53.8% of the found the "rationale". The other symptoms show less importance in each GSA.

We statistically calculate the symptoms and show the first two most frequent symptoms in GSA 1 to GSA 7 in Table 8.2. Pearson's r shows the correlations between each symptom and each GSA. The values are between 0 to 1, which indicate a positive linear relationship. The "unanimity" has the most positive relationship with GSA 3, while "rationale" has the worst positive relationship with GSA 7.

The Top 10 groupthink phenomena in safety analysis. As we can see in Table

Table 8.2: Statistic Descriptive on the Occurrence and Influential of Janis's Groupthink Symptoms in GSA

GSA	Symptoms	N	Mean	St. Dev	St. Error	Min	Median	Max	95% CI lower	95% CI upper	r
1	Direct pressure	22	2.091	0.848	0.136	1	2	3	1.737	2.445	0.654
	Invulnerability	22	2.045	0.928	0.149	1	2	3	1.658	2.433	0.712
2	Stereotypes	16	2.750	0.559	0.090	1	3	3	2.476	3.024	0.440
	Morality	14	2.500	0.732	0.117	1	3	3	2.117	2.883	0.622
3	Self-censorship	23	2.696	0.621	0.099	1	3	3	2.442	2.949	0.886
	Unanimity	22	2.727	0.617	0.099	1	3	3	2.470	2.985	0.978
4	Direct pressure	20	1.900	0.831	0.133	1	2	3	1.536	2.264	0.928
	Morality	17	1.471	0.696	0.111	1	1	3	1.140	1.801	0.897
5	Mindguards	19	2.421	0.815	0.131	1	3	3	2.054	2.788	0.833
	Rationale	17	2.824	0.513	0.082	1	3	3	2.580	3.067	0.725
6	Invulnerability	15	2.000	0.894	0.143	1	2	3	1.547	2.453	0.814
	Rationale	11	1.455	0.498	0.080	1	1	2	1.160	1.749	0.400
7	Direct pressure	23	2.565	0.712	0.114	1	3	3	2.274	2.856	0.483
	Rationale	21	2.952	0.213	0.034	2	3	3	2.861	3.043	0.370

Table 8.3: The Top 10 Groupthink Phenomena

#	The Top 10 groupthink phenomena in GSA
1.	The managers do not plan safety analysis activities in detail.
2.	The technical members overestimate their capability in functional development and ignore the importance of safety analysis.
3.	During system/function/structure analysis, the non-functional departments are under stereotype and are always absent.
4.	During safety analysis, beside developers and safety experts, the other members keep silence.
5.	During safety analysis, the team members always keep consistence with the opinions from senior safety experts.
6.	When providing safety analysis solutions, the team members prefer explaining the rationality of the existing solutions.
7.	The safety analysts spontaneously freeze the internal safety-related documents.
8.	During safety verification, the safety analysts set themselves as a "police" to blame the development team.
9.	During safety validation, the internal safety expert takes the role as a "lawyer" to rationalise their safety assurance to a third party.
10.	The team performs safety analysis aiming to provide the required evidence for certification (paperwork culture).

8.3, based on the quantitative data of the fourteen symptoms in Table 8.2, we summarise the Top 10 groupthink phenomena in GSA. One phenomenon might cover two to three symptoms in each GSA.

8.1.5.3 RQ 3 & RQ 4: How could the Top 10 phenomena in groupthink occur and how to handle them?

1. The managers do not plan safety analysis activities in detail.

Description In GSA 1, the project managers plan the safety analysis following norms. All the members believe that they have a process to ensure safety and do not show more concerns on achieving safety goals. Yet, the norm needs a detailed adaption in different projects. That causes the groupthink symptoms "direct pressure" from the project manager on discussing the detailed plan and "invulnerability" on the existing plan from norms or

procedures. One interviewee mentioned: "*Basically, we refer to the norms and internal procedures. The concrete execution shall have been described in the procedure documents by the procedure design department.*"

Reasons and solutions *Reason 1: Lack of norm concerning the implementation and execution procedure.* The industries rely on norms for functional safety assurance. Thus, the project manager do not place emphasis on planning and preparing the implementation and execution activities. When we ask them: "*How do you perform safety analysis*", they answered: "*We have the norms. Yet, the implementation and execution are depending on the projects.*" *Solution 1: Assign critical roles.* A minimum implementation and execution of the requirements in norms shall be established into each critical role or department. Each role or department has its responsibilities, authorities, execution methodology and the corresponding accountability [Lev11]. *Solution 2: Provide a second-chance meeting.* The arrangement of responsibilities or tasks could be established in a second meeting. That will not influence the original structure of the meetings. It aims to implement, update and enforce the execution of norms. *Solution 3: Invite external expert.* The plan and preparation of safety analysis could invite experts from the procedure design department. It helps the design on performing safety analysis in a specific project, whereas the specific project helps the procedure design department to make a general procedure for future work.

2. The technical members overestimate their capability in functional development and ignore the importance of safety analysis.

Description In GSA 1, some technical members are function-oriented and are kind of "super stars". They believe in their function development experiences, which could avoid risks. The voice concerning safety analysis is ignored, which causes groupthink symptom "invulnerability" on the capability of functional development. One interviewee mentioned: "*The functions can bring the largest profits. We have worked in this area many years and are able to avoid some basic risks. We have still safety testing in the end.*"

Reasons and solutions *Reason 1: Cohesion.* In most safety-critical industries,

the employee's mobility is relatively low. Most employees have been working for a long time in the same area and company. That increases cohesion. However, the cohesion decreases the capability to face change and accept outside suggestions, such as the suggestions from safety experts. *Solutions 1: Split groups.* More working groups could provide more ideas. The members in each group should be heterogeneous. In a small group, performing safety analysis is more productive than in a large group [Whe09]. In addition, the competitive mechanism could stimulate passions. *Reason 2: Lack of norms concerning the interface between functional development and safety analysis.* Safety analysis and functional development shall be concurrent. Yet, the execution lacks a clear definition on the interfaces between them. One developer mentioned: "*In the project plan, we follow the functional safety norm. But the establishment is separated from us (development).*" *Solution 2: Invite external expert.* Some safety experts could be invited to the internal meetings of development teams. One solution that has not been mentioned by Janis but specific for SCS is that an effective communication channel between functional development and safety analysis has to be designed in the safety analysis plan and preparation.

3. During system/function/structure analysis, the non-functional departments are under stereotype and are always absent.

Description In GSA 2, the non-functional departments like sales or purchasing are always absent. They do not provide suggestions and follow the technical members. This is a groupthink symptom called "stereotype" on the non-functional departments. As one system developer mentioned: "*We invite them to the meetings to analyse the system, but they always give some reasons like "I have not received the emails" and not take part in them. They said "I have no idea about development" and let us make decisions.*"

Reasons and solutions *Reason 1: Group insulation.* Safety system working groups operate at different levels of organisation [Lev11]. The safety analysis concerning system/function/structure should not stop at the functional level. However, the safety analysis among groups is always isolated, which

includes working places, tasks, responsibilities and organisation/personal development management, which causes too many excuses on not taking part in the GSA 2. It seems a lack of conscientiousness. *Solution 1.1: Invite external expert.* In industry, the employees do not prefer working on the tasks that goes beyond the working scope. Some members are not full-time on safety analysis. The invitations are sometimes personal and non-regulated. As far as we know, these invitations happens mostly occasionally. As one interviewee mentioned: "*Before the meeting, we send the email to a sales manager. But we got no feedback.*" The time, the channel and the complementary methodology are all indeterminate. The cross-functional departments should facilitate interactions and build connections, such as communication channels and interaction patterns. *Solution 1.2: Establish multiple groups.* All the departments including non-functional departments are able to provide suggestions on the system/function/structure from their perspectives. *Reason 2: Lack of norms concerning the requirements on joined departments.* The cooperation among groups could be improved by personally inviting external experts. However, to be more efficient, the practitioners should rely on some regulations. The existing procedures include few requirements on joined departments. *Solution 2:* The requirements on joined departments could be written into the procedures to normize GSA.

4. During safety analysis, beside developers and safety experts, the other members keep silence.

Description In GSA 3, only developers and safety experts join the discussion. This is called "self-censorship" in groupthink. One sales manager mentioned: "*They have more expertise and take the responsibility for this.*" Hence, the sales manager follows other technical members.

Reasons and solutions *Reason 1: Temporarily low self-esteem.* Safety analysis is a window into systems, everyone could see and connect hazards in the daily work. The members from the sales department have more practical experiences on products. They can see the possible hazards from the end-users' viewpoint. The members from the purchasing department know

more about the purchased components from suppliers. The possible hazards from these components and purchasing procedures could cause further accidents during the development. However, most employees, especially in functional safety, QA or functional development department, think safety analysis more technical in-depth. The non-technical members are not confident. One production manager said: "*What we mentioned during safety analysis is too high-level, which cannot really help the development of products and even the safety analysis. Even more, what they talked is apart from our knowledge areas.*" *Solution 1.1: Assign critical evaluators.* Each one should be given his/her responsibilities to raise questions in his/her specific areas. Everyone's voice deserves respect. An "ad-hoc" brainstorming sessions are possible. *Solution 1.2: Provide a second-chance meeting or feedback channels.* Someone, who does not want to raise questions in the meetings, could have a second chance through trustful feedback channels to provide suggestions. *Reason 2: High stress from external threats.* A non-professional voice will raise doubts on the employee's core competence. People like to join the discussion in his or her area of expertise. Especially the employees who are already in the senior level. The fresh man also does not want to show his cons in front of his or her leader and other colleagues. *Solution 2: Improve feedback channels.* One interviewee mentioned: "*It would be great, if we could provide suggestions anonymously.*" For example, an anonymous mailbox has been tried by some departments.

5. During safety analysis, the team members always keep consistence with the opinions from senior safety experts.

Description In GSA 3, at the beginning, almost all the members join the discussion. When senior managers or safety experts express their ideas or suggestions, they seek consistence with them. It is called "unanimity" in groupthink.

Reasons and solutions *Reason 1: Impartial leader.* The execution of safety analysis is guided by a moderator, who guides the procedures as well as controls the time. Mostly, to limit the discussion about safety analysis, the

senior managers or safety experts stand out and express their opinions to end the discussion. It makes the decision-making procedures partial. As one moderator mentioned: "*Sometimes there is a tough discussion between different opinions, we have to stop them, because of the time plan. I try to get in touch with some leaders or experts maybe through eye contact. They will provide a decision and we use it to finish this part.*" That causes the decision overly relying on personal decision-making capability and integrating more non-technical considerations from managers, such as cost and schedule. *Solution 1: Make key member impartial.* First, a normized safety analysis procedure with a defined, transparent and explicit decision-making procedure helps the moderator and experts to be impartial. One moderator suggested: "*We always like to use "high-risk priority" principle rather than directly adopting key member's decision.*" That could avoid the bias from positions and thus concentrate on risks. Second, we have to ensure that those who are making safety-related decisions are fully informed and skilled [Lev11]. Third, the decision could be assessed and improved. *Reason 2: Cohesion.* This cohesion is between team members and senior managers or safety experts. Some team members were hired by the senior managers. They have been working for him or her for a long time and give the same voice with their managers. *Solution 2: Split groups.* The practitioners could spit the high-cohesive members into separate groups when performing safety analysis.

6. When providing safety analysis solutions, the team members prefer explaining the rationality of the existing solutions.

Description In GSA 4, the team members prefer using existing solutions for mitigating risks. As one interviewee said: "*When we got a risk, we would like to use the existing methods from the former projects or similar products. It saves more efforts and is more reliable to validate its feasibility of the existing one rather than to propose a new one.*" We can see three groupthink symptoms from this case: "rationale", "morality" and "direct pressure".

Reasons and solutions *Reason 1: Lack of norms on change management procedure.* We reviewed an internal change management manual and ad-

vised one senior quality manager. He mentioned: "*We do the impact analysis when performing change management. Most of the time, we concentrate on the changed parts. Based on the experiences and expert opinions, we analyse the possible impacts. In safety analysis, we notice the requirements on change management in ISO 26262-8, part 8, but there is no mention of systematic technique.*" The members believe in the morality of the former solutions for the unchanged parts and try to rationalise them. *Solution 1: Enhance change management.* Establishing the management of change requirements for evaluating all changes for their impacts on safety systematically. *Reason 2: High stress from external threats.* The delivery deadline cases stress. That causes the team members using the existing solutions for mitigating risks and trying to rationalise them. One interviewee said: "*Before a deadline, we won't switch to developing new solutions. That increases risks on delivery. When we have an existing solution, we use it. The worst case, we could write the possible risks on the internal (for developers) or external (for customers) reports.*" It shows that the existence of a solution is more important than the effectiveness of a solution under delivery pressure. *Solution 2: Devote a block of time to provide alternative scenarios.* Some alternatives or new solutions could be reserved and reconsidered. *Reason 3: Temporarily low self-esteem.* The team members are afraid of undertaking responsibilities. They do not believe in their capability to propose a better solution than the former one. Furthermore, they have to invest more effort for new solutions. When there is an accident during the execution, they have to be responsible for it. A "blame culture" exists. *Solution 3: (Refer to solution 2) Devote a block of time to provide alternative scenarios.* In addition, the proposal of the opinions and the execution of them shall be divided. After accepting the new solutions, the team members shall share responsibilities. To this end, the practitioners need consider anonymity.

7. The safety analysts spontaneously freeze the internal safety-related documents.

Description In GSA 5, the safety analysts take the responsibility for the documents and freeze them to avoid review and modification. One intervie-

wee mentioned: "*Sometimes I have a question about the results what we have discussed in the meeting. Later, I am not able to open such files anymore. When I ask the relevant person, he said that we have already discussed and decide in the meeting. The results are documented. If there are questions, I should ask other people.*" That is called "mindguards" in groupthink symptoms.

Reasons and solutions *Reason 1: Lack of norms on the authority of safety-related documents.* The norms require a lot of documents for each activity. Yet, there is no description of authority. The documentation engineers protect the documents by themselves as to protect the decisions of groups. *Solution 1: Make key members impartial.* The generation of documents should be transparent. The documents should be appropriately open. To manage safety, there is a safety information system including safety analysis related documentation. They are not only for maintenance, but also for guiding safety audit and assessment. *Reason 2: Group insulation.* The management of safety-related documents including project related documents, organisation related documents and process management documents is separate. That causes too many limitations on authority. *Solution 2: Discuss with trusted people.* In documentation management, the industry could consider some online documents management tools. The non-confidential documents are preferred to be open to project-related people.

8. During safety verification, the safety analysts set themselves as a "police" to blame the development team.

Description In GSA 6, the safety analysts sometimes overestimate their authority on a "police" role and neglect their responsibility as a "teacher" or a "doctor". They shall work on finding safety vulnerabilities and guide the development to ensure the safety. However, some of them are concerned only with finding vulnerabilities and even blaming. As one developer mentioned: "*They always point out our failures. That gives us a bad mood. When we ask further for a suggestion, they said that they are not clear about the detailed development, they care about if the development can satisfy these safety requirements. We cannot get any help.*" It shows a groupthink symptom

"invulnerability".

Reasons and solutions *Reason 1: Group insulation.* The functional safety department and the development team are isolated. *Solution 1: Invite external expert.* The safety verification is preferred to be happening during the development phase. The development team could invite relevant members to take part in their meetings or daily work. Effective communication channels and feedback channels are useful. *Reason 2: Temporarily low self-esteem.* A strong attitude and blame are sometimes because of lacking confidence. The reason of the safety analysts taking the role as police to blame could be that they temporarily lack technical capability to teach and cure them. *Solution 2: Devote a block time to discuss the solutions.* A preparation time slot and a harmonious environment for cross-functional communication are vital.

9. During safety validation, the internal safety expert takes the role as a "lawyer" to rationalise their safety assurance to a third party.

Description In GSA 7, the internal safety expert shows partiality as a lawyer. He or she sometimes does not aim to find the hazards rather to rationalise the safety assurance capability and pass a third party validation. That is a symptom called "rationale" in groupthink.

Reasons and solutions *Reason: Cohesion.* The internal safety expert is a member in the company. As one safety expert said: "*I am working for the company, what I should do is to validate that our process is safe enough to deliver a safe product.*" The organisational cohesiveness changes the objectiveness of safety validation. *Solution: Invite external expert.* One senior manager suggested inviting an external safety expert to validate the process and product before third party validation. The external safety expert should be independent and might come from the same company, but different subsidiaries.

10. The team performs safety analysis aiming to provide the required evidence for certification (paperwork culture).

Description During the whole GSA, the team members perform safety anal-

ysis aiming to provide evidence for certification. That is called "rationale" in groupthink.

Reasons and solutions *Reason: Impartial leader.* The leader takes safety superficially. That causes the groupthink symptom that no other member is concerned with it. As one senior manager mentioned: "*We do safety analysis is because the norm requires it. By following the norms, we can get a certification in parallel with our products. That helps us to sale more products.*" Another feedback from the developers is: "*We have no idea about the norms. We just follow the requirements from our leader. He takes care about the norm-related issues.*" However, employees need to feel that they will be supported if they show concern for safety.

Solution: Make key members impartial. A manager's open and sincere concern for safety in everyday can have a major impact on the reception given to safety analysis [Lev11]. First, open norm is important, as well as open communication on the norms. Each team member needs to know "how" and even "what". "Leadership-collaboration" should replace "command-control management" [Coc02]. Second, a thorough integration of safety culture is necessary from three levels: Surface level cultural artifacts; Organisational rule, values, practices; Values and deep cultural assumptions [Sch10]. The surface level includes everyday practices. In the middle, it states the rules like policy, norms and guidelines. Lastly, values and deep cultural assumptions is used for making leaders with more emphasis on safety.

8.1.6 Discussion

We note the consideration on groupthink in safety analysis. From our viewpoint, the occurrence of conflict, together with an effective decision-making, is positive for safety analysis. However, the occurrence of groupthink is potentially fatal. We investigate seven GSA according to the norms. We calculate the occurrence and frequential of groupthink symptoms in the GSA. We propose Top 10 phenomena, which happen in safety analysis and cause the results lacking group considerations. We find out the possible reasons and solutions which are specific for safety analysis to avoid groupthink

referring to Janis's seven antecedents and nine recommendations. Eight symptoms have all been found in our Top 10 phenomena. Six of seven antecedents are found with a specific form in GSA. "Lack of norms" concerning various aspects is the most proposed cause, which has been five times mentioned in the Top 10 phenomena. "Cohesion", "Group insulation" and "Temporarily low self-esteem" show three times. Yet, "homogeneous" has not been found in our cases. "Homogeneous" means the similarities of members' social backgrounds and ideology. In most safety-critical industries, they have a normised organisation development procedure. They have a regular hiring process according to the requirements from organisational structure . The phenomenon of a "homogeneous" team is rare. They range the members from new employees to experts. The knowledge background is also diverse. We map eight of the nine recommendations in the proposed solutions. "Invite external expert" is the most popular one, while "make key members impartial" shows also its importance. Only the recommendation on "setting a devil role" has not been mentioned by the interviewees. It has some difficulties. Each one has his core area. The devil role is hard to cover a wide range of topics. In addition, the interviewees show also unwillingness. More than that, the interviewees mentioned two other suggestions: Improve the communication/feedback channels and regular procedure requirements on joined members or departments and a systematical change management.

8.1.7 Threats to Validity

8.1.7.1 Internal Validity

First, the seven GSA cover only the major parts of safety analysis activities. We walked-through the norms and relevant literature to summarise the GSA that happen in a group. Yet, such activities were observed more than thirty types [Vil+17]. Some of them are integrated into the development process. Thus, we use five main activities when performing safety analysis by using various techniques and two activities including safety verification and safety validation, which are directly relevant to the safety analysis. Other activities

like safety audit, which concerns more on the safety assurance process, is not included in this article. Second, the survey questions could be incomplete. Groupthink is not a popular word in software engineering. To make it more understandable to the participants, we simplified the original descriptions from Janis and set the questions in an easy way. Thus, we propose the survey available online for review. During the interviews, it was possible to explain them in-depth. Third, the influence from culture can be a limitation. We performed the survey in four Europe companies. Two of them are international companies covering Asia subsidiaries. The participants are from both Europe and Asia. However, it is not enough to generalise the result. We hope to expand this study in a wider scope.

8.1.7.2 Construct Validity

First, since the interviews were conducted with industries, the audio recordings were not allowed. We recorded the results only relying on the transcript by the first author. Concerning some missing points and confidentiality, we contacted the participants again and provided our protocol to ensure the results. Second, the results have many inconsistent terminologies. Some of them are internal terms in each company. We discussed our results and determined them with the primary company A.

8.1.7.3 External Validity

First, the primary company is in the automotive industry. We generated our results for RQ 1 mainly from the automotive area functional safety norms ISO 26262. We referred to the safety norms ISO 14971 and IEC 60601 in medical equipment area as well. However, these two areas cannot stand for all the safety-critical industries. Second, the attitudes toward groupthink also rely on different roles. We aim to cover as many roles as possible. Most of the participants are from quality assurance departments, development

teams and relevant managers. Some of them are from sales, purchasing and production. The amount of participation is limited.

8.1.7.4 Reliability

Some contents concerning groupthink are sensitive, such as "impartial leadership". We designed and collected the survey anonymously. Yet, the participants may also reserve some results.

8.2 Mapping Groupthink in S-Scrum

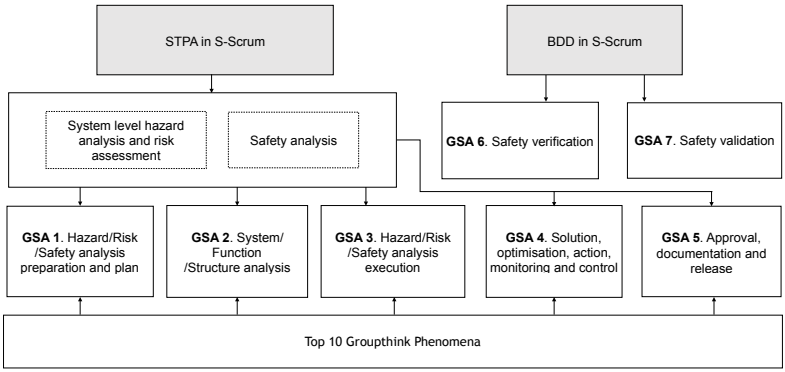


Figure 8.4: The Top 10 Phenomena in Preliminary S-Scrum

Now we describe how the aforementioned Top 10 phenomena occur in the preliminary S-Scrum. As we can see in Figure 8.4, GSA 1 to GSA 5 happen during STPA safety analysis, while GSA 6 and GSA 7 happen during BDD safety verification. In these Top 10 groupthink phenomena, phenomenon 1 and phenomenon 2 happen in GSA 1, phenomenon 3 happens in GSA 2, phenomenon 4 and phenomenon 5 happen in GSA 3, phenomenon 6 happens

in GSA 4 and phenomenon 7 happens in GSA 5. When performing BDD in preliminary S-Scrum, we should consider phenomenon 8 and phenomenon 9. During the whole preliminary S-Scrum, we should consider phenomenon 10. Now we describe them in detail.

1. The managers do not plan safety analysis activities in detail.

In preliminary S-Scrum, safety planning is facilitated in pre-planning meeting and sprint planning meeting, which is insufficient concerning the technical in-depth considertations. To overcome this phenomenon, (1) S-Scrum should assign responsibilities, authorities, execution methodologies and corresponding accountability of the internal and external safety expert concerning safety planning. (2) S-Scrum could establish a two-part sprint planning [Rub12]. The first part focuses on "what", while the second part focuses on "how". (3) Preliminary S-Scrum prefers the invitation of external experts, such as external safety experts. Here we recommend that S-Scrum could invite experts from the procedure design department to take part in the sprint planning meeting to help the plan of safety analysis and verification.

2. The technical members overestimate their capability in functional development and ignore the importance of safety analysis.

In preliminary S-Scrum, the developers given a higher priority to functional development than safety analysis and verification. Preliminary S-Scrum prefers that the team members are kept as long as possible [Coh10]. A negative organisation cohesion exists. This makes the team members over believing in their team capability. To overcome this phenomenon, (1) S-Scrum could split the discussion of safety analysis in small groups. (2) The interfaces between functional development and safety analysis are important in S-Scrum. Besides an invitation of external safety expert, an effective communication channel should be determined in the sprint planning meeting, such as the execution of daily chats or regular safety meetings.

3. During system/function/structure analysis, the non-functional departments are under stereotype and are always absent.

In preliminary S-Scrum, this phenomenon might happen throughout daily work. The non-functional departments should take part in the daily or weekly scrum meeting. Yet, it happens occasionally. To overcome this phenomenon, (1) S-Scrum should regulate responsibility, time, channel and complementary methods to invite non-functional members in or before the sprint planning meeting. (2) S-Scrum should regulate joined departments in some key milestones.

4. During safety analysis, beside developers and safety experts, the other members keep silence.

In preliminary S-Scrum, discussion happens frequently between developers and safety experts. However, the safety experts guide the discussion most of time. In addition, the product owner, scrum master and other stakeholders prefer keeping silence and not expressing their own opinions. To overcome this phenomenon, (1) S-Scrum should assign critical responsibilities for each person to express his viewpoint in his area. For example, the product owner could express his opinions concerning the relation between safety analysis and products, while scrum master gives the opinions on the execution of safety analysis. (2) S-Scrum could set an anonymous feedback channel, such as an anonymous mailbox.

5. During safety analysis, the team members always keep consistence with the opinions from senior safety experts.

In preliminary S-Scrum, the team members prefer keeping consistence with the opinions of product owner or safety experts. To overcome this phenomenon, (1) S-Scrum should regulate a procedure with a defined, transparent, explicit and professional decision-making mechanism. That helps the managers to be impartial. (2) S-Scrum could also split the high cohesion people in different groups. This step is able to be combined with

safety analysis discussion in the solution 1 of phenomenon 2.

6. When providing safety analysis solutions, the team members prefer explaining the rationality of the existing solutions.

In preliminary S-Scrum, the safety requirements, which have existing solutions, are given a higher priority. For example, "Smart Home" has similar sub-systems. Some safety requirements are similar with a same solution. To overcome this phenomenon, (1) STPA is a system-theory based safety analysis method, which analyses the whole system rather than the single changing part. Thus, the existing solutions cannot be used for new requirements. A rational changing management procedure exists in preliminary S-Scrum. (2) When there is an alternative solution, S-Scrum should provide a block of time to consider it. The block of time can be several hours to several days depending on the length of each iteration. (3) Each team member could provide solutions. When the solution is accepted, the whole team should share responsibilities.

7. The safety analysts spontaneously freeze the internal safety-related documents.

In preliminary S-Scrum, the safety product backlog with safety stories and story map with safety epics are open in Jira. Yet, the execution documents of STPA are kept privately by safety experts. To overcome this phenomenon, (1) the permission of safety documents in S-Scrum should be determined. (2) Transparency is important in ASD [Rub12]. S-Scrum encourages to make the documents as transparent as possible. The STPA report and the agile safety plan could be internally open online.

8. During safety verification, the safety analysts set themselves as a "police" to blame the development team.

In preliminary S-Scrum, the external safety expert and product owner review the product in the sprint review meeting. The examination occupies

more time than providing guidances. Yet, preliminary S-Scrum integrates safety analysis and verification in short iterations, which supports guiding the development rather than only examining the products at the end of the sprint. In addition, an effective communication between the internal safety expert and the external safety expert in regular safety meetings enhances the understanding of the deliverable product, which reduces the possibility of misunderstandings on the product, which might cause blaming.

9. During safety validation, the internal safety expert takes the role as a "lawyer" to rationalise their safety assurance to a third party.

Preliminary S-Scrum has not been used in a real-world industrial project with a third party assessment or audit. Yet, we facilitated an assessment by walking through the norm ISO 26262 by an independent safety expert. This phenomenon occurred. However, when the purpose of such assessment changes from process assessment to process improvement, we are able to avoid trying to rationalise the process, rather, to find out problems.

10. The team performs safety analysis aiming to provide the required evidence for certification (paperwork culture).

In preliminary S-Scrum, we reduce this phenomenon effectively. Safety analysis and verification are integrated into iterations and happen in parallel with the development. Preliminary S-Scrum does not produce additional documents to satisfy the requirements from norms, rather using existing scrum artifacts to, on the one side, fulfill the requirements from norms, on the other side, to support communications. However, to satisfy the certification, some documents are still necessary, such as an agile safety plan. The improvements of other documentation in S-Scrum remain as the future work.

8.3 Conclusion

In this chapter, we solve the problem concerning requirements prioritisation in the preliminary S-Scrum through improving group work and avoiding groupthink. We investigate groupthink in safety analysis and verification, and map the findings into S-Scrum. We conduct a case study in seven safety-critical companies. We investigate seven GSA concerning groupthink. We statistically summarise the most frequent symptoms through 39 surveys. We investigate further the Top 10 groupthink phenomena through 17 interviews. We investigate and map the reasons with Janis's theory. We generate further the solutions combining Janis' theory and safety system management. We find that the groupthink does exist in safety analysis. The practitioners should find phenomena and consider solutions. We map the phenomena, reasons into S-Scrum and propose solutions, "second-chance meeting" and "cross-functional meeting", in the final S-Scrum. Our studies show limitations mainly on the research domains and countries.

Chapter 9

S-Scrum

*In this chapter, after fixing the four challenges in the preliminary S-Scrum (verification, planning, communication and requirements prioritisation), we summarise and propose a final "**S-Scrum**".*

The main contributions of this chapter are:

- **We propose the final S-Scrum.** *We propose the final S-Scrum in Figure 9.1 and illustrate it in this chapter.*
- **We illustrate S-Scrum from 3 dimensions: Activities; Roles; Documentation.** *These 3 dimensions are further divided into 3, 2 and 2 sub-dimensions respectively. Activities are described from tasks, input and output, and communication. Roles are described from responsibilities and competences. Documentation is described from contents and communication.*
- **We evaluate S-Scrum by reviewing the norm ISO 26262 in automotive area.** *We conducted 3 rounds of evaluation. First, we evaluate S-Scrum informally by reviewing ISO 26262 by the author. Second, we interview 15 senior-level experts to evaluate S-Scrum in practical safety-critical industries. Third, we do a walkthrough based on ISO 26262 formally by one certified safety expert and one safety expert with more than 20 years working experiences in safety-critical industries.*

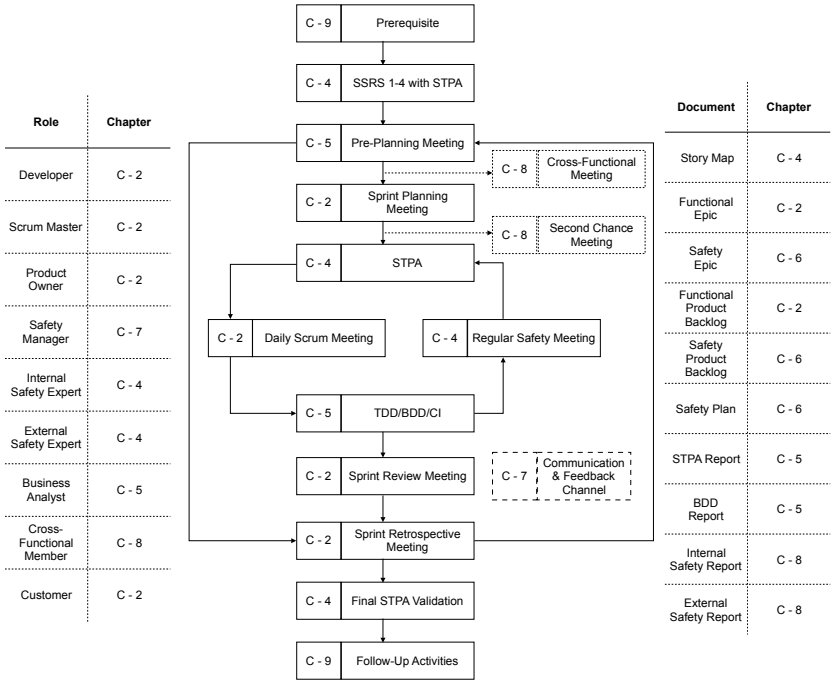


Figure 9.1: S-Scrum

9.1 Activities

As we can see in Figure 9.2, we illustrate the activities in S-Scrum in 3 sub-dimensions: Tasks; Input and output; Communication[1].

9.1.1 Prerequisite

Before starting S-Scrum, the organisation should have the following prerequisites: (1) An operational quality management system complying with a quality management norm, such as ISO/TS 16949, ISO 9001 or equivalent. (2) When the functional safety concepts rely on other technologies or external measures, they should be discussed referring to ISO 26262-3, 8.4.3.2 and 8.4.3.3. (3) Configuration management should be according to ISO 26262-6, Annex C and ISO 26262-8, 7.4. (4) Embedded software in terms of ASIL definition should refer to ISO 26262-6, 7.4.10, 7.4.11 and 7.4.17. (5) When the project includes distributed development, ISO 26262-8, 5.4.3 and 5.4.4

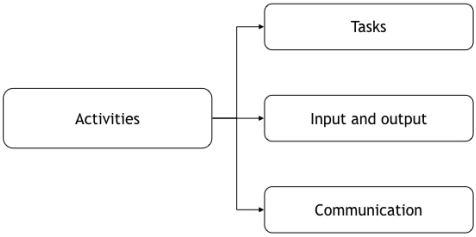


Figure 9.2: Dimensions to describe Activities in S-Scrum

[1]We explain only the major communication purpose and challenges. Other communication purposes may also occur throughout the iterations.

should be followed. (6) Functional safety assessment and functional safety audit should have been done for suppliers before starting S-Scrum. (7) The "confidence in the use" of software tools is prepared. (8) The qualification of reused software components is prepared.

9.1.2 SSRS 1-4 with STPA

The detailed description is shown in Chapter 4.
Tasks: SSRS 1-4 is a set of classic activities in functional safety norm IEC 61508 [Com11b]. It includes: (1) Item definition; (2) Initiation of safety lifecycle; (3) Hazard analysis and risk assessment; (4) Functional safety concept. S-Scrum uses the pre-steps of STPA including system description and control structure to perform hazard analysis. HARA is used for classifying hazardous events.
Input and output: The input is a general overview of the system. The outputs are the descriptions of the system functional goal, system safety goal, system scope, safety scope, system requirements, hazards and safety requirements.
Communication: In SSRS 1-4 with STPA, the team members communicate primarily for deriving high-level safety issues. The communication challenge can be a lack of documentation to support communication concerning STPA and HARA.

9.1.3 Pre-Planning Meeting

The detailed description is shown in Chapter 4.
Tasks: A pre-planning meeting is arranged before the sprint planning meeting. It includes: (1) Change impact analysis; (2) Tailoring of safety activities; (3) Decomposition of ASIL; (4). Defining tools, methods and languages in each lifecycle phase; (5) Supplier and customer relationship; (6) Solving conflicts between safety requirements and functional requirements, as well as their acceptance criteria (including HW and SW interfaces), for the coming sprint.

Input and output: The inputs are: (1) The output from the former activity; (2) A project plan; (3) A safety plan; (4) A functional product backlog; (5) A safety product backlog. The outputs are (1) A revised safety plan; (2) A revised functional product backlog; (3) A revised safety product backlog; (4) Team members' competence evaluation (team velocity and team capability).
Communication: In the pre-planning meeting, the team members communicate primarily for planning and solving conflicts. The communication challenge can be a lack of opinions from multiple functional departments and recording processes with appropriate tools.

9.1.4 Sprint Planning Meeting

The detailed description is shown in Chapter 2.
Tasks: The sprint planning meeting is a regular activity in normal scrum. The team agrees on a goal. In addition, the team performs safety planning concerning the determination of safety product backlog items, which can be aligned with safety goals, together with the determination of their acceptance criteria.
Input and output: The input is: (1) The output from the previous activity. The outputs are: (1) A sprint backlog; (2) A revised safety plan.
Communication: In the sprint planning meeting, in addition to planning, the members aim to establish commitments for realising safety goals in the coming sprint. The storage of the recording documents for commitments remains challenges. The These recording documents should support communication as well.

9.1.5 STPA

The detailed description is shown in Chapter 4.
Tasks: STPA is performed iteratively and incrementally in parallel with the development to derive safety requirements for the safety-guided design. The safety experts and developers execute STPA step 1 and step 2 through a constant communication and cooperation during development.

Input and output: The inputs are: (1) The outputs from the previous activity; (2) An incremental system architecture. The outputs are: (1) Incrementally derived accidents, hazards, UCA and safety requirements; (2) System design decisions. (3) An STPA report.
Communication: During STPA safety analysis, the team aims to derive and implement safety requirements. The communication challenge can be "a misunderstanding of the safety requirements between developers and safety analysts".

9.1.6 Daily Scrum Meeting

The detailed description is shown in Chapter 2.
Tasks: The daily scrum meeting is a regular activity in normal scrum. It focuses on getting people together and sharing the big picture of what is happening among team members. Additionally, in S-Scrum, the internal safety expert is a member in the development team. He or she should join the daily scrum meeting and answer three questions. Safety-related issues are discussed during daily scrum meeting in S-Scrum.
Input and output: The inputs are: (1) The outputs from the previous activity; (2) Individual work state. (3) Possible conflicts, obstacles and problems during development. The outputs are: (1) A transmitted state of work of each team member; (2) Possibly fixed conflicts, obstacles and problems.
Communication: In the daily scrum meeting, the communication occurs as a regular discussion or monitoring the status. The primary challenge can be "a misunderstanding during the discussion among multiple functional members or between safety experts and developers".

9.1.7 BDD

The detailed description in shown in Chapter 5.
Tasks: BDD is performed iteratively and incrementally in parallel with the STPA and the development during each iteration. It needs the decision on

acceptance criteria for the safety requirements in the pre-planning meeting . The execution of BDD is combined with TDD and CI.
Input and output: The inputs are (1) The outputs from the previous activities; (2) Source code. The outputs are (1) BDD test suites; (2) Safe code snippets; (3) A BDD report.
Communication: During the BDD safety verification, the team aims to derive test suites for verifying safety requirements and implementing them in the code. The challenge can be "a misunderstanding among business analyst, developer and safety analyst when generating test scenarios".

9.1.8 Regular Safety Meeting

The detailed description is shown in Chapter 4.
Tasks: In the regular safety meeting, the internal safety expert and external safety expert exchange their state and incoming requirements of safety issues. External experts can be invited when there are other specific issues related to safety issues.
Input and output: The inputs are (1) The outputs from the previous activities; (2) Safety-related work state; (3) Possible safety-related conflicts, obstacles and problems. The outputs are (1) An exchanged work state; (2) Fixed safety-related conflicts, obstacles and problems.
Communication: In the regular safety meeting, the members aim to monitor the state of safety issues and share safety knowledge. The different knowledge backgrounds of customers (transferred through the external safety expert) and developers (transferred through the internal safety expert) might cause misunderstandings on the exchanged information.

9.1.9 Sprint Review Meeting

The detailed description is shown in Chapter 2.
Tasks: The sprint review meeting is a regular activity in normal scrum. In the sprint review meeting, the practitioners inspect and adapt the results of the work or the potentially shippable product increment. The additional

tasks in S-Scrum include the review for safety product backlog items and their acceptance criteria. A periodical safety assessment on the shippable product can be performed in a specific milestone.

Input and output: The inputs are: (1) The outputs from the previous activities; (2) A shippable product; (3) An STPA report; (4) A BDD report. The outputs are (1) A safe shippable product; (2) A closed sprint backlog; (3) Unfinished user stories for the coming sprints.

Communication: In the sprint review meeting, the team members aim to demonstrate their results. The challenge is the possibly fragmented and inconsistent information during demonstration.

9.1.10 Sprint Retrospective Meeting

The detailed description is shown in Chapter 2.

Tasks: The sprint retrospective meeting is a regular activity in normal scrum. The team has an opportunity to examine what is happening, analyse the way they work, identify ways to improve and make plans to implement these improvements. The additional tasks include the retrospective concerning safety assurance processes. A periodical safety audit can be performed before and discussed in a sprint retrospective meeting in a specific milestone depending on the project plan.

Input and output: The inputs are: (1) The outputs from the previous activities; (2) An executed sprint. The output is: (1) Possible improvements on the processes.

Communication: In the sprint retrospective meeting, the team members aim to improve the processes and techniques. The safety experts and developers are expected to share their knowledge with non-functional departments, so that the members in non-functional departments can contribute to the improvement.

9.1.11 Final STPA Validation

The original description is shown in Chapter 4.
Tasks: The final STPA validation is a replacement of the original RAMS validation from the functional safety norm EN 50128 [MSL15] and focuses on safety validation. The safety experts perform STPA step 1 and step 2 on the final product.
Input and output: The inputs are: (1) The outputs from the previous activities; (2) A final product. The outputs are: (1) A safe product; (2) An internal safety report; (3) An external safety report.
Communication: In the final STPA validation, the team members aim to review their product and demonstrate their results. The challenge can be "the fragmented and inconsistent information in the high-level safety requirements when generating internal safety report and external safety report".

9.1.12 Follow-Up Activities

The follow-up activities are not regular activities in S-Scrum. The practitioners can use them according to the specific project settings. They include: (1) Confirmation review. (2) Functional safety audit. (3) Functional safety assessment.

9.1.13 *Cross-Functional Meeting

This meeting is an optional activity in S-Scrum. We describe it in chapter 8. S-Scrum recommends a cross-functional meeting to enhance the cooperation among multiple functional departments. It is scheduled after the pre-planning meeting and before the sprint planning meeting. The safety experts have a commitment concerning the initial plan. The members or representatives from multiple functional departments will participate in and provide suggestions for this initial plan before the starting of a sprint. In principle, the inputs for the sprint planning meeting, such as an initial

plan or product backlog, should have been discussed by multiple functional departments.

9.1.14 *Second-Chance Meeting

This meeting is an optional activity in S-Scrum. We describe it in chapter 8. S-Scrum recommends a second-chance meeting to enhance an insufficient plan. It is scheduled after the sprint planning meeting and before the execution of the sprint. There are practitioners who use two-staged sprint planning meetings. The second-chance meeting can be combined with the second stage of a two-staged sprint planning meeting. The tasks should include the assignment of responsibilities and the distribution of detailed tasks. In principle, all the team members should be clear about "what to do" and "how to do" after the second-chance meeting.

9.2 Roles

As we can see in Figure 9.3, we illustrate the roles in S-Scrum in 2 sub-dimensions: Responsibilities and competences. We describe the roles, which are already in normal scrum as well due to the mixed responsibilities and competences.

9.2.1 Developers

The detailed description is shown in Chapter 2.
Responsibilities: (1) The developers should perform the creative work of designing, building[1], integrating and testing (HW and SW interfaces are included). (2) Assisting safety experts for safety-related issues including safety analysis, safety verification, safety validation, allocation of safety requirements to functional development. (3) Writing BDD test cases. (4) Participating in organisation activities, such as planning, grooming the product backlog, inspecting and adapting the product and process.

[1] To support the correctness of the design and implementation, the design principles and coding guidelines should be used.

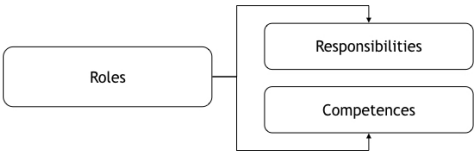


Figure 9.3: Dimensions to describe Roles in S-Scrum

Competences: (1) Domain knowledge. (2) Expertise in common safety practices and cross-functionality skills. (3) The capability for team work including self-organisation and transparent communication. (4) Knowledge of functional safety practices.

9.2.2 Scrum Master

The detailed description is shown in Chapter 2.
Responsibilities: (1) Assisting the product owner and safety experts to facilitate and maintain the functional backlog and the safety backlog to ensure all the user stories can be understood. (2) Coaching, promoting and educating the team and stakeholders within scrum principles and functional safety norms. (3) Helping the team to avoid or remove impediments to its functional goals and safety goals. (4) Helping the team to follow a regular process. (5) Coaching and guiding multiple functional departments.
Competences: (1) Expertise in a scrum master role including characteristics and skills. (2) Expertise in domain-specific SCS development processes.

9.2.3 Product Owner

The detailed description is shown in Chapter 2.
Responsibilities: (1) Arranging the project plan. (2) Ensuring that the products satisfy customers' requirements. (3) Updating work products. (3) Demonstrating the products. (4) Cooperating with team members. (5) Negotiating the priorities between the functional product backlog and the safety product backlog. (6) Controlling the scope, economics and schedule of the project. (7) Defining the SIL with safety experts. (8) Collaborating with stakeholders.
Competences: (1) Expertise in domain and product knowledge. (2) Expertise in management including people skills, decision-making skills and accountability skills. (3) Knowledge of functional safety.

9.2.4 Safety Manager

The detailed description is shown in Chapter 8.
Responsibilities: (1) Planning functional safety activities. (2) Coordinating functional safety activities. (3) Maintaining the safety plan. (4) Monitoring the progress of safety activities. (5) Ensuring compliance with ISO 26262. (6) Planning the resources for supporting functional safety activities. (7) Confirming the safety plan.
Competences: (1) Expertise in functional safety and QA. (2) Expertise in domain and product knowledge. (3) Expertise in management.

9.2.5 External Safety Expert

The detailed description is shown in Chapter 4.
Responsibilities: (1) The safety manager can take over the role of external safety expert in certain circumstances. (2) Transferring business values from safety manager to development team. (3) Planning the safety product backlog. (4) Cooperating with internal safety expert. (5) Reviewing the safety product backlog. (6) Planning the resources for functional safety activities. (7) Managing the confirmation review, safety audit and safety assessment.

(8) Tailoring safety requirements and allocating safety requirements into functional development. (9) Collaborating with safety manager.
Competences: (1) Expertise in functional safety and QA. (2) Product and domain knowledge.

9.2.6 Internal Safety Expert

The detailed description is shown in Chapter 4.
Responsibilities: (1) Performing STPA. (2) Performing BDD. (3) Planning, managing and maintaining safety product backlog. (4) Negotiating the priorities between the functional product backlog and the safety product backlog. (5) Tailoring safety requirements and allocating safety requirements to software components. (6) Cooperating with development team in terms of designing, building, integrating and testing (HW and SW interfaces are included). (7) Generating and maintaining STPA and BDD reports.
Competences: (1) Expertise in STPA. (2) Expertise in BDD. (3) Expertise in domain and product knowledge. (4) Capability for team work.

9.2.7 Business Analyst

The detailed description is shown in Chapter 5.
Responsibilities: (1) Transferring business value. (2) Performing BDD. (3) Cooperating and communicating with the safety manager or the external safety expert.
Competences: (1) Knowledge of domain and product. (2) Knowledge of safety verification. (3) Expertise in marketing and business values development.

9.2.8 Suppliers

The detailed description is shown in Chapter 7.
Responsibilities: (1) Providing evidence of the execution of safety assurance on the products, components and services. (2) Cooperating and communicating with the safety manager or the external safety expert.

Competences: (1) Capability to provide safe products, components and services. (2) A professional safety assurance process in the organisation. (3) A strong capability for market competition.

9.2.9 Cross-Functional Members

The original description is shown in Chapter 8.
Responsibilities: (1) Providing safety information from their knowledge areas, such as sales, purchasing or production. (2) Cooperating and communicating with the safety manager or the external safety expert.
Competences: (1) Expertise in specific aspects of products. (2) Knowledge of functional safety.

9.2.10 Customers

The original description is shown in Chapter 7.
Responsibilities: (1) Providing clear and reliable requirements. (2) Possibility to be "on-site" and keep constant communication with development team.
Competences: (1) A higher reliability. (2) Accountability.

9.3 Documents

As we can see in Figure 9.4, we illustrate the documents in S-Scrum in 2 sub-dimensions: Contents and communication. We exclude the documents which are already described in normal scrum. They are: Functional product backlog; Sprint backlog; Functional epics; Functional user stories.

9.3.1 Story Map

The detailed description is shown in Chapter 2.
Contents: A story map consists of ordered user stories along two independent dimensions. The horizontal axis represents ordered user activities as

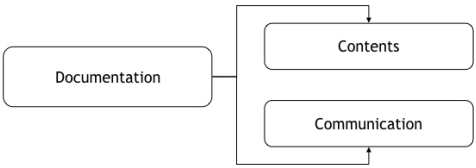


Figure 9.4: Dimensions to describe Documents in S-Scrum

functional epics, while the vertical axis represents an increasingly sophisticated implementation as functional user stories. We add additional items on the horizontal axis as safety epics, while after each safety epic, we write safety stories. We assign SIL for each safety epic and safety story referring to at least one safety goal.
Communication: A story map can be used for transferring safety requirements to the external stakeholders with a clear overview. Thus, the sensitive and confidential information should be avoided showing in the story map.

9.3.2 Safety Epic

The detailed description is shown in Chapter 6.
Contents: The safety epic represents high-level safety requirements in the form of: *To satisfy <the overall safety needs>, the system must <always be able to reach a safe state>*. SIL can be recorded in each safety epic.
Communication: Safety epics are used primarily for clarifying, implementing, planning and tracing safety requirements. These safety epics show the safety requirements between high-level and low-level. Thus, a shared

understanding between customers and developers is important.

9.3.3 Safety Story

The detailed description is shown in Chapter 6.
Contents: The safety story represents low-level safety requirements in the form of: *To keep <the control action> safe, the system must <achieve or avoid something>*. SIL is recorded in each safety story.
Communication: The safety story is mainly for transferring and implementing low-level safety requirements. The storage and the monitoring of technical in-depth information should be ensured.

9.3.4 Safety Product Backlog

The detailed description is shown in Chapter 4.
Contents: A safety product backlog represents a list of safety stories with prioritisation. Each change request is recorded with a unique identifier, date, reason and description.
Communication: A safety product backlog is used for planning the coming sprint. Multiple functional departments should participate in the discussion of the safety product backlog.

9.3.5 Safety Plan

The detailed description is shown in Chapter 6.
Contents: A safety plan is used for managing and guiding the execution of safety activities in a project including dates, milestones, tasks, deliverables, responsibilities and resources. The safety analysis plan and safety verification plan are included. The added contents are: (1) Summary of changes. (2) Tailoring of safety activities. (3) Overview of findings. (4) Safety requirements. (5) Test suites.
Communication: The safety plan is proposed for planning. Its contents should be transparent, open and easy to understand to relevant members.

9.3.6 STPA Report

The detailed description is shown in Chapter 5.
Contents: An STPA report is generated after the execution of the STPA safety analysis including system descriptions, system goals, safety goals, accidents, hazards, safety requirements, UCA and unsafe scenarios.
Communication: An STPA report aims for demonstrating safety analysis results. Fragmented and inconsistent information should be avoided. More importantly, the contents should be easy to understand by non-safety experts.

9.3.7 BDD Report

The detailed description is shown in Chapter 5.
Contents: A BDD report is generated after the execution of the BDD safety verification including test scenarios, test cases, test results and code. The test scenarios, test cases and test results include unique identifiers.
Communication: The BDD report aims for demonstrating safety V&V results. Fragmented and inconsistent information should be avoided. The test suites should support communication among stakeholders.

9.3.8 Internal Safety Report and External Safety Report

The original description is shown in Chapter 8.
Contents: An internal safety report records the final results of safety activities during projects, while an external safety report provides additional guidelines for customers or end-users to keep a safe operation.
Communication: The internal safety report aims for tracing safety requirements and sharing safety knowledge. A suitable tool is necessary to maintain the internal safety report. The contents should be easy to understand after a long time, especially among different cultures and languages. The external safety report aims for external demonstration. Sensitive and confidential information should be recorded appropriately.

9.4 Evaluation

The evaluation of the final *S-Scrum* is according to an iterative review of ISO 26262. We choose the *automotive domain* for the final evaluation.

9.4.1 An Overview of ISO 26262

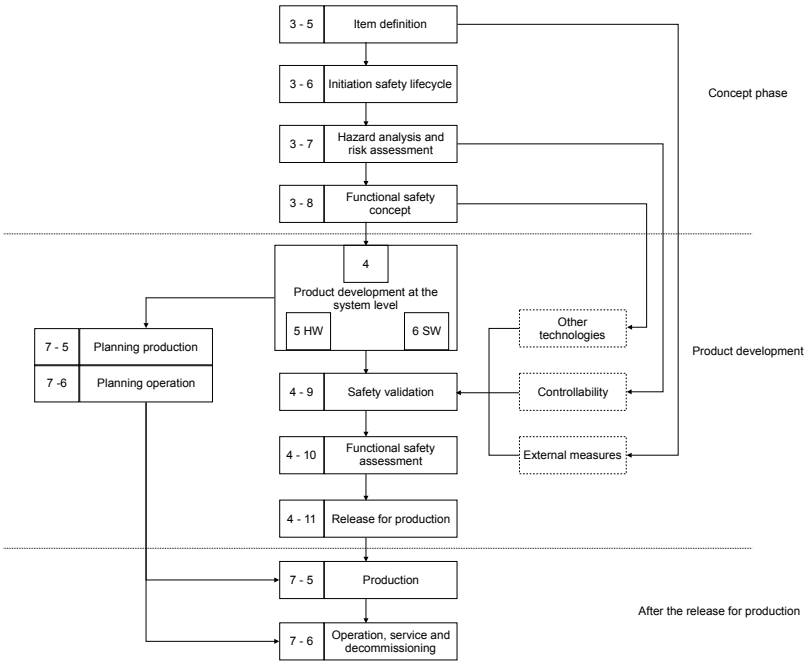


Figure 9.5: An Overview of ISO 26262

ISO 26262, Road vehicles - Functional safety, is an international norm for functional safety of electrical and/or electronic systems in the production of automobiles defined by ISO in 2011. The norm ISO 26262 is an adaptation

of the Functional Safety norm IEC 61508 for Automotive Electric/Electronic Systems. ISO 26262 defines functional safety for automotive equipment applicable throughout the lifecycle of all automotive electronic and electrical safety-related systems.

The goals of ISO 26262 are: (1) To provide an automotive safety lifecycle (management, development, production, operation, service, decommissioning) and to support tailoring the necessary activities during these lifecycle phases. (2) To cover the functional safety aspects of the entire development process (including such activities as requirements specification, design, implementation, integration, verification, validation, and configuration). (3) To provide an automotive-specific risk-based approach for determining ASILs. (4) To use ASILs for specifying the item's necessary safety requirements for achieving an acceptable residual risk. (5) To provide requirements for validation and confirmation measures to ensure a sufficient and acceptable level of safety is achieved.

ISO 26262 consists of ten parts: (1) Vocabulary. (2) Management of functional safety. (3) Concept phase. (4) Product development at the system-level. (5) Product development at the hardware level. (6) Product development at the software level. (7) Production and operation. (8) Supporting processes. (9) ASIL-oriented and safety-oriented analysis. (10) Guideline on ISO 26262. We demonstrate it partially in Figure 9.5.

We review S-Scrum through part 2, part 3, part 6, part 8 and part 9. S-Scrum is not yet suitable for the hardware level as well as the system-level. S-Scrum is used for software development and not extended to production and operation. Thus, we exclude part 4, part 5 and part 7.

9.4.1.1 Operation

As we can see in Figure 9.6, we conducted an informal review of ISO 26262. Second, we conducted 15 interviews with senior-level experts in diverse safety-critical domains including Bosch (Germany), Roche (Germany), Daimler (Germany), General Motors (USA), Boeing (USA), Continental (Ger-

many). Third, we conducted a walkthrough by one TÜV Rheinland[1] certified safety expert and one safety expert who has more than 20 years working experiences concerning ISO 26262.

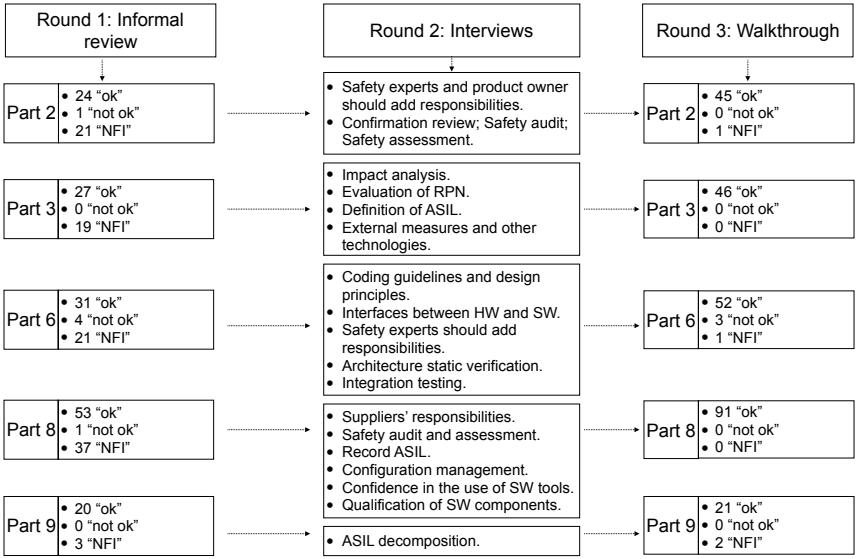


Figure 9.6: Evaluation Results

9.4.2 Results

As we can see in Figure 9.6, we generate the review results from 3 stages. In terms of round 1 and round 3, we compare S-Scrum with each item in ISO 26262 to check if they are "ok", "not ok" and "need further investigation". We derive unclear topics from round 1 and design them as interview questions in round 2. Based on the recommendations from round 2, we revise S-Scrum and evaluate it using the same metrics "ok", "not ok" or "need further investigation".

In round 1, 154 (59.2%) items are "ok", 6 (2.3%) items are "not ok", and

[1] TÜV is a technical inspection association in Germany established in 1800, which provides inspection and product certification services.

102 (38.9%) items "need further investigation".

In round 2, we summarise the vague issues: (1) Blurred/Missed responsibilities of the product owner and safety experts. (2) No action of confirmation review, safety audit and safety assessment. (3) Missed consideration of "change impact analysis". (4) No methods to evaluate RPN. (5) No definition of ASIL. (6) No consideration of external measures or other technologies. (7) No clear requirements on the use of code guidelines and design principles. (8) No specific actions on HW/SW interfaces. (9) The difficulty to perform architecture static verification. (10) No consideration of integration testing. (11) No definition about suppliers' responsibilities. (12) No recordings of ASIL in documentation. (13) No consideration of configuration management. (14) Lack of the way to perform "confidence in use of SW tools". (15) Lack of the way to perform "qualification of SW components". (16) No specific time to perform ASIL decomposition.

Based on the results from the interviews, we investigate the existing solutions in industry and map them into our S-Scrum. For (1), they do split responsibilities for product owners and safety experts, since the safety experts are not full-time employed in one project. The product owner has more responsibilities. For (2), they perform confirmation review, safety audit and safety assessment before a product delivery, normally 2 to 3 times per year. The execution does not rely on iterations. For (3), they perform impact analysis before starting the development. For (4) and (5), a combination of STPA and HARA is proposed to possibly define RPN. For (6), (7) and (8), we can directly assign them in S-Scrum. For (9) and (10), S-Scrum, at this stage, has no solutions. This remains as a limitation in S-Scrum. For (11), since the investigated industries are large companies with a lot of distributed projects and various suppliers, the responsibilities must be clear when establishing a project. For (12), ASIL is labeled in each safety requirement. For (13), CI is preferred in some agile projects for configuration management with appropriate tools, such as Jenkins. For (14), the practitioners evaluate "confidence in use" whenever there is a new tool in use. Between projects, they change the tools slightly. For (15), few companies do that in our investigated context. Only one OEM mentioned that

the suppliers will perform the qualification of SW themselves with provided evidence. For (16), the development team conducts the decomposition under the guidance of the safety expert.

In round 3, we revise our S-Scrum with the aforementioned recommendations. The evaluation results show that 255 (97.3%) items are "ok", 3 (1.1%) items are "not ok" and 4 (1.5%) items "need further investigation". The "not ok" items are the architecture design as well as the execution of static architecture analysis and verification. In addition, "Does ISO 26262 have to be based on the reference model (V-model)?" and "how to use STPA for performing quantitative analysis for HW parts in embedded systems?" need further investigation. We consider them as our future work.

The evaluation of S-Scrum excludes the requirements that are highly correlated with hardware as well as the specific requirements for the systems with ASIL C and ASIL D.

9.5 Conclusion

In this chapter, we propose the final S-Scrum. We describe it in 3 dimensions: Activities; Roles; Documentation. We integrate our results from groupthink into S-Scrum activities, while we enhance communication channels by analysing the communication purposes and paying attention to their challenges. The evaluation is based on ISO 26262 in automotive industries, while the senior expert-level interviews are conducted in 6 large SCS companies. After revising S-Scrum, the formal review shows a positive result by achieving 97.3% requirements from ISO 26262. However, there are three limitations: (1) Our evaluation does not contain the automotive systems with ASIL C and ASIL D. (2) Hardware systems and embedded software systems with highly correlated hardware are not suitable for S-Scrum. (3) Large-scale or distributed projects may use S-Scrum, but extensions are necessary. The evaluation in other safety-critical domains as well as using S-Scrum in real automotive industrial projects are our potential future work.

Chapter 10

Discussion and Conclusion

In this chapter, we summarise this dissertation with implications, limitations and future work.

The main contents of this chapter are:

- **We summarise this dissertation.** *We provide a short summary of this dissertation concerning the methods we used and the results we obtained.*
- **We highlight the implications of this dissertation.** *S-Scrum shows implications for practitioners concerning a possible process together with individual techniques and for researchers concerning a new way to investigate ASD for developing SCS.*
- **We identify limitations of S-Scrum and its overall research.** *Although we believe that the research of S-Scrum is based on robustly empirical results, several known limitations cannot be avoided in this dissertation.*
- **We propose future work.** *On one hand, the immediate indication for further research is to narrow down our limitations. For example, to evaluate S-Scrum in a real automotive project or to expand the evaluation in other safety-critical domains. On the other hand, we expect to open up avenues for research to face today's complex SCS, such as a consideration of other critical aspects including information security.*

10.1 Discussion

In this dissertation, we propose an S-Scrum to apply ASD for developing SCS. However, *we recommend practitioners and researchers focusing more on our standalone chapter's contributions rather than this entire S-Scrum*.

Despite S-Scrum is proposed as a general development process, the implementation of an entire ASD process, in developing SCS, still has some context-specific problems. Currently, to facilitate individual techniques in the existing processes seems more feasible than an entire process.

In Chapter 4, we begin this dissertation by proposing the use of STPA in a scrum development process and name it "*S-Scrum*". We aim to face the changing architectures in ASD when performing safety analysis. We demonstrate our concept of the use of STPA, illustrate the concept with an example - Airbag System, and investigate it in a one-year student project "Smart Home" with 14 participants. The preliminary S-Scrum contains challenges in *requirements prioritisation, communication, planning and verification.* Some initial solutions are proposed through interviews and questionnaires, such as a pre-planning meeting or separated safety experts. We facilitate the initial optimisations in the preliminary S-Scrum within the same project. The process is more agile, while the products are safer than before.

The results show a possibility to use STPA in a scrum development process, as well as the feasibility of this preliminary S-Scrum. Even though the agility reduces slightly compared with original scrum, team members can derive safety requirements during development and implement safety requirements in the product. However, the initial solutions remain in organisation management. A technical in-depth investigation of these challenges is necessary. *Thus, we improve our preliminary S-Scrum with Chapter 5, Chapter 6, Chapter 7 and Chapter 8.*

In Chapter 5, we propose the use of BDD in S-Scrum. It aims to face the challenges in *verification* and *communication* in S-Scrum. The preliminary S-Scrum uses UAT for safety verification. However, UAT works well for high-level requirements and is not specific for safety verification. In addition, UAT does not support an effective communication between technical members

and customers. BDD relies on verifying system behaviours. Safety is to ensure a safe behaviour. Therefore, we believe that BDD is suitable for verifying safety requirements. In addition, BDD is an agile technique. It generates test suites in natural language with a clear glue for tracing the relevant safety requirements.

We combine BDD with STPA in S-Scrum and conduct a controlled experiment with 44 participants to evaluate the use of BDD. The participants are assigned in 4 groups relying on 2 different methods (BDD and UAT) and 2 systems (APS and CSGS). They portray as a developer and a customer in 3 sessions (30 minutes/session). The measures we used are productivity, test thoroughness, fault detection effectiveness and communication effectiveness, as well as other test-driven development experiments.

Only the communication effectiveness shows a significant difference between BDD and UAT. All other measures show no statistically significant differences. That contradicts our original opinion that BDD should show better effects. We observe that the arrangement of hierarchy and name conventions when writing BDD cost too much time. Therefore, we develop a semi-automated tool for helping to generate BDD test suites. We replicate the experiment with 11 additional participants using BDD with the semi-automated tool in APS system. The productivity is 7 times greater, the test thoroughness is 1.5 times greater and the fault detection effectiveness is 2 times greater than the use of BDD without the semi-automated tool and UAT. In conclusion, the results support the use and show the effectiveness of using BDD for safety verification in S-Scrum.

In Chapter 6, we adapt and develop three documents in S-Scrum, namely safety story, safety epic and agile safety plan. We aim to face the challenges in *communication* as well as *planning*. We adapt three templates from Safe Scrum with an integration of STPA. Since the safety analysis method differs from FMEA in Safe Scrum to STPA in S-Scrum, the safety stories are changing from restricting function's and component's failures to UCA. Also, the agile safety plan needs the consideration on the plan as well as outputs from STPA.

To evaluate these three documents, we use them in S-Scrum as a case study

in the one-year student project with 14 participants. By using questionnaires and interviews, we obtain the results that safety story and safety epic can support an effective communication, while the agile safety plan contributes more to planning and certification, such as safety process overview, priority management, providing safety backup knowledge as well as having a safety assessment report.

In Chapter 7, we investigate communication channels during safety analysis and verification in 7 small to large safety-critical industries with 39 experts. We aim to face the challenges specifically on *communication*. We found 9 popular communication channels when performing safety analysis and verification. The practitioners have normally 28 purposes for communication, while we explored Top 10 challenges in communication. We highlight the importance of communication purposes - what or why to communicate. In S-Scrum, when performing safety analysis and verification, meetings and documentation can achieve more communication purposes than in our research context. For instance, "daily scrum meeting" and "regular safety meeting" in S-Scrum can achieve an additional communication purpose "provide feedback and comments". In the demonstration of S-Scrum, we include clear communication purposes in each activity, based on our results, the practitioners can select possible channels, be clear about the challenges and avoid the challenges in specific contexts.

In Chapter 8, we explore groupthink during safety analysis and verification in 7 small to large safety-critical industries with 60 experts. We aim to face the challenge primarily on *requirements prioritisation*, together with group work. We notice that there do exist groupthink during safety analysis and verification. We list the Top 10 phenomena. The antecedents and recommendations are proposed specifically based on Janis's psychological theory model. "Lack of norms" is the most proposed antecedent, while "invite external experts" is the most popular recommendation.

In S-Scrum, safety analysis and verification happen with an increasing amount of group work, the Top 10 phenomena do exist in S-Scrum and happen more frequently in short iterations. Based on the proposed recommendations in our research context, "cross-functional meeting" and "second-chance

meeting" are possibly to be established in S-Scrum to avoid groupthink during safety analysis and verification.

In Chapter 9, we generate our final S-Scrum by including the overall technical in-depth optimisations, such as the basic optimisations from Chapter 4, the use of BDD from Chapter 5, the safety story, safety epic and agile safety plan from Chapter 6, enhanced communication channels from Chapter 7, a second-chance meeting and cross-functional meeting from Chapter 8.

Finally, we allocate S-Scrum to the automotive domain and evaluate the consistency of the final S-Scrum with an automotive functional safety norm - ISO 26262. We conduct 3 rounds of evaluations, i.e. one round of informal review by the author, one round of interviews with senior-level experts and one round as a formal walkthrough by one certified safety expert and one safety expert with more than 20 years' experience on functional safety in the automotive area. The results are increasing from a satisfaction rate of 59.2% (the first round of informal review) to 97.3% (the third round of formal walkthrough) to the norm based on experts' recommendations (the second round of interviews). So far, S-Scrum is validated to be feasible for both student projects and projects in automotive companies with ASIL A and ASIL B.

In summary, the contributions of S-Scrum are from two sides. From the technical side, we notice the importance of a lack of integrated *safety analysis and verification* in the existing ASD due to a lack of a stable architecture. STPA and BDD can solve these challenges and are thought to be more suitable for today's sophisticated SCS. From the management side, we notice the conflicts between *communication and documentation*, as well as an increasing number of *group work*, which increases the risk of the occurrence of groupthink. We adapt and develop three documents in S-Scrum to support communication and investigate challenges when using existing communication channels in S-Scrum. We investigate reasons and propose solutions in S-Scrum to avoid groupthink by referring to a psychological theory model. To this end, we summarise the existing activities, responsibilities and documents, as a comprehensive S-Scrum, to achieve a norm's requirements.

10.2 Implications

Currently, most of the existing studies prefer a hybrid model to integrate ASD with traditional development processes. Safe Scrum is a representative, which is initially proposed in 2012 [SMH12] and successfully used in 2018 [MS18], while the HELENA survey [Kuh+17] is another research group investigating ASD in SCS with a focus on a hybrid model. The HELENA survey research group starts publishing results from 2017. They seek consistency with the functional safety norms and keep using the traditional safety assurance activities.

Safe Scrum has been reviewed by several domain norms and been allocated to railway and the metro domains with complaints to the norm EN 50129 (IEC 62425). Safe Scrum proposes to use such as SSRS phase 1-4 and RAMS validation outside the iteration, and add a role with safety responsibilities as well as documents based on agile artifacts. However, Safe Scrum is mainly developed aiming to satisfy norms. It either uses recommended techniques or organisational management from norms in scrum. The running of a sprint is considered as a mini Waterfall or mini V-model.

The HELENA survey puts its emphasis on hybrid methods between traditional development processes and ASD from practice. It starts from industrial experiences to investigate the state-of-the-art about the use of hybrid methods. Based on the recent results [Kuh+17], the HELENA survey has addressed six major motivations to use hybrid methods rather than pure ASD, such as a need for a stepwise transition or client constraints. Since the HELENA survey has just started one year ago and is mainly spreading in European countries, it is still at the stage of exploring motivations and problems rather than proposing solutions.

We believe in the value of the HELENA survey for enhancing the chain of evidence of using ASD in SCS within a large scope, S-Scrum, however, provides solutions, together with evaluations. Compared with Safe Scrum, *S-Scrum advocates the use of systems theory (or systems theory-based methods), which promotes the system thinking in ASD.*

For researchers, we have three major implications. (1) The first implication

lies in the novel technique in S-Scrum, *STPA-BDD*. It successfully solved the problem of a lack of safety analysis and verification during each iteration. (2) The second implication is that we propose another possible way for the researchers in the area of using ASD for development SCS: *We claim that seeking for using traditional safety assurance techniques (or purely following norms) in ASD is* ***not*** *the only way to make ASD applicable for developing SCS. To investigate some good (agile) practices, which are satisfied with agile principles, is more effective to solve the root problems, which may be caused by the nature of ASD that has an instinct conflict with traditional safety assurance.* (3) The third implication is that our research *links safety assurance in process management with an attention on human aspects.* We notice that there are rich studies of process improvement in safety or (x)-critical domains. They either put their attention on norms, such as CMM and ISO, or derive experiences through specific project settings. On the other hand, human aspects in organisational management arouse increasingly more attention in software engineering, such as "agile methods", "behaviour software engineering" or "psychoempirical software engineering". Yet, in safety or (x)-critical domains, human aspects in organisational management might produce various effects, as well as encompass diverse causalities. For instance, "groupthink" and "communication channels" seem to have normal effects in regular projects. Yet, in SCS, they are dangerous.

For practitioners, we have three major implications. (1) We propose an executable process "*S-Scrum*" both for academic and automotive industrial projects with ASIL A and ASIL B. It includes a general execution process with descriptions of activities, roles and documents, which can be used in practice. (2) In addition, when the project already has a running process, we recommend the use of individual techniques and artifacts, such as *STPA* and *BDD* and their relevant documents to face the probable challenges. (3) Our results have implications for *safety management*. In SCS, we provide recommendations for managing an effective group work during safety analysis and verification in order to avoid groupthink. Also, we propose the existing challenges in communication when performing safety analysis and verification that the management should notice. S-Scrum compensates the

weakness of the conflicts between documentation and communication when performing STPA and BDD. Finally, S-Scrum is evaluated at the management level to be satisfied with the functional safety norm ISO 26262 and validated as a feasible ASD in the automotive domains for developing the systems with ASIL A and ASIL B.

In this dissertation, we argue that no matter practitioners or researchers, in the area of using ASD for development SCS, should *jump out of* the framework of norms, but still *not deviate from* norms.

10.3 Limitations

As a dissertation in terms of process improvement, we agree with the opinions of Humphrey [Hum89] and Dybå [Dyb05] concerning the importance of specific contexts and specific business values. However, given the context in SCS, **we have the limitation on the evaluation in a real-world industrial project**, as same as other research [GPM10], which remains in theory. The practical evaluation is a challenge for investigating ASD in SCS. More importantly, SCS projects have different SIL. Even though the evaluation could be facilitated in a project developing SCS, the results are not generalisable.

In 2017, Francisco et al. [Vas+17] conducted a systematic literature review about the approaches to software process improvement. Academic validation is supported by recent research [Gor+06] [Fal+18b], which we used for Chapter 4, Chapter 5 and Chapter 6. Static validation in industries is also postponed, such as interviews, surveys and norm reviews, which we used for Chapter 7, Chapter 8 and Chapter 9. However, dynamic validation has not been found in recent studies. It shows the gap in recent empirical methods for evaluating software process.

In our research context, we conjecture the reason as *the motivation (of using ASD for development SCS) is still not strong enough to persuade industrial practitioners to take risks.* The running projects are working, even with some delays, overspending or dissatisfaction of customers. Yet, a use of a new process, which is still immature and incomplete, might cause even worse

results. Changing a process seems taking over more responsibilities and needs more efforts than a technique. A stepwise transition is preferable. That is the reason why we notice the HELENA survey, which starts from 2017, and consider how to push S-Scrum further and meet industrial needs.

Therefore, we recommend the practitioners noticing our standalone chapter's contributions with individual techniques. The evaluation of the individual techniques is based on method triangulation, such as case studies in a student project, controlled experiments in academic settings and industrial case studies. The limitations are discussed at the end of each chapter.

Nevertheless, S-Scrum is entirely evaluated through reviewing the norm. For an entire process, a norm review is preferable and reliable before a real-world implementation [SG+17] [SMH12]. To enhance the robustness, we conducted one round of informal review, one round of formal review, together with interviews with senior-level experts. Thus, although we lack an evaluation of S-Scrum in a real-world industrial project, we encompass as many empirical evaluation methods as possible to validate S-Scrum as a whole as well as each technique individually.

In addition, **we allocate the SCS domain of S-Scrum only in automotive industry**. There are scarce studies addressing domain-specific challenges in terms of the use of ASD in SCS. To strengthen S-Scrum further in industry, *we must determine a domain to go deeper*. Besides the equivalent motivations and the state-of-the-art about using ASD in recent SCS, such as smart home, automotive, medical equipment or aviation, we decide to use S-Scrum in the automotive domain due to *an easy-to-get resource (location and cooperation possibilities)*.

The evaluation shows that S-Scrum can achieve the automotive systems with ASIL A and ASIL B, which actually encompass 80% of automotive systems. Additionally, given that the evaluation of S-Scrum is through reviewing norms, we can also easily extend S-Scrum to other safety-critical domains, since the domain-specific norms show similarities [Bau+10].

Furthermore, **S-Scrum, at the current stage, is not suitable for developing hardware systems**, as well as highly hardware-correlated embedded systems. This is due to a historical problem - the limitations of original scrum.

The practitioners show difficulties in adopting scrum for developing hardware systems, due to a high cost of changing or managing dependencies of hardware components as well as a time limitation for customers to approve prototypes [BHL17]. Thus, we propose S-Scrum only for software systems and embedded software systems.

Finally, **S-Scrum has not been evaluated for large-scale or distributed organisations**. This might cause worries for practitioners in SCS. The organisation of SCS is possible on a large-scale with distributed locations and management. A scrum team seems more popular for small organisations. However, recent studies show the possible way with a plenty of evidence to organise scrum in large-scale or distributed projects [EP17] [KLM17] [Din+17], such as a use of "scrum of scrums" [All17]. Thus, we strongly believe in the possibilities to extend S-Scrum into large-scale or distributed organisations.

10.4 Future Work

The research area of using ASD in developing SCS is novel in this decade and contains many uncultivated lands in the future.

On the one hand, to narrow down the limitations of this dissertation is our immediate future work. (1) We expect to *execute S-Scrum in one (or more) real-world project(s) in automotive industry*. Some research or platform projects assigned with low ASIL and on a small scale can be considered rather than customer projects with high ASIL. (2) S-Scrum can *be evaluated through other domain-specific norms* by domain experts as a pre-step to strengthen S-Scrum into wide-ranging safety-critical domains. Domain-specific differences can be used for extending S-Scrum. (3) *Theoretical research on a large-scale or distributed S-Scrum* may arouse interest. The cooperation among cross functional departments, especially in different time zones, seems important in industry.

On the other hand, to use ASD for developing SCS, safety assurance should consider other critical aspects, which may influence the safety of systems and

indirectly cause hazards. *Information security* is the most influential, such as in autonomous driving, smart home or aviation management. To *consider information security when performing safety analysis* is important for today's SCS. STPA can encompass information security causalities, which lead to an unsafe event. Yet, information security itself also causes serious losses. To perform *security analysis interleaved with safety analysis* in S-Scrum is more suitable for today's SCS. A combination between existing security analysis and safety analysis methods are more practical in industries to pursue than an integrated safety and security method.

Bibliography

[AD17] J. D. Arthur, J. B. Dabney. “Applying standard independent verification and validation (IV&V) techniques within an agile framework: Is there a compatibility issue?” In: *Proceedings of the Annual IEEE International Systems Conference*. 2017, pp. 1–5 (cit. on p. 52).

[Adl77] R. B. Adler. *Confidence in communication: A guide to assertive and social skills*. Harcourt School, 1977 (cit. on p. 101).

[Adz09] G. Adzic. *Bridging the communication gap: Specification by example and agile acceptance testing*. Neuri Limited, 2009 (cit. on p. 94).

[Aer12] R. T. C. for Aeronautics. “Software considerations in airborne systems and equipment certification.” In: *DO-178C* (2012) (cit. on pp. 26, 42).

[Alj+09] H. Aljazzar, M. Fischer, L. Grunske, M. Kuntz, F. Leitner-Fischer, S. Leue. “Safety analysis of an airbag system using probabilistic FMEA and probabilistic counterexamples.” In: *Proceedings of the 6th International Conference on Quantitative Evaluation of Systems*. 2009, pp. 299–308 (cit. on p. 64).

[All] A. Alliance. *What is agile software development?* 2013. Retrieved 2015. (cit. on p. 34).

[All01] A. Alliance. “Agile manifesto.” In: *http://www.agilemanifesto.org* 6.1 (2001) (cit. on pp. 34, 44).

[All17] A. Alliance. *Scrum of scrums*. 2017 (cit. on p. 227).

[Ant13] B. Antoine. “Systems-theoretic hazard analysis (STPA) applied to the risk review of complex systems: An example from the medical device industry.” PhD thesis. Massachusetts Institute of Technology, 2013 (cit. on p. 27).

[AW14] A. Abdulkhaleq, S. Wagner. “A software safety verification method based on system-theoretic process analysis.” In: *Proceedings of the International Conference on Computer Safety, Reliability, and Security*. 2014 (cit. on p. 39).

[Ban+12] A. Banerjee, K. K. Venkatasubramanian, T. Mukherjee, S. K. S. Gupta. “Ensuring safety, security, and sustainability of mission-critical cyber-physical systems.” In: *Proceedings of the IEEE* 100.1 (2012), pp. 283–299 (cit. on p. 146).

[Bas92] V. R. Basili. *Software modeling and measurement: The Goal/Question/-Metric paradigm*. Tech. rep. 1992 (cit. on p. 71).

[Bau+10] P. Baufreton, J. Blanquart, J. Boulanger, H Delseny, J. Derrien, J Gassino, G Ladier, E Ledinot, M Leeman, P Quéré, et al. “Multi-domain comparison of safety standards.” In: *Proceedings of the 5th International Conference on Embedded Real Time Software and Systems*. Toulouse, France, 2010, pp. 13–25 (cit. on p. 226).

[Bec00] K. Beck. *Extreme programming explained: Embrace change*. Canada: Addison-Wesley, 2000 (cit. on pp. 26, 34, 35).

[Bec04] K. Beck. *Extreme programming explained: Embrace change*. Canada: Second Edition, Addison-Wesley, 2004 (cit. on p. 35).

[BH06] J. Bowyer, J. Hughes. “Assessing undergraduate experience of continuous integration and test-driven development.” In: *Proceedings of the 28th International Conference on Software Engineering*. 2006, pp. 691–694 (cit. on p. 110).

[BHL17] A. I. Böhmer, P. Hugger, U. Lindemann. “Scrum within hardware development insights of the application of scrum for the development of a passive exoskeleton.” In: *Proceedings of the International Conference on Engineering, Technology and Innovation*. IEEE. 2017, pp. 790–798 (cit. on p. 227).

[Bow93] J. Bowen. “Formal methods in safety-critical standards.” In: *Proceedings of the Software Engineering Standards Symposium*. 1993, pp. 168–177 (cit. on pp. 26, 39).

[Bre15] E. Brechner. *Agile project management with kanban*. Pearson Education, 2015 (cit. on pp. 34, 35).

[Bro+10] B. K. Brockman, M. E. Rawlston, M. A. Jones, D. Halstead. “An exploratory model of interpersonal cohesiveness in new product development teams.” In: *Journal of Product Innovation Management* 27.2 (2010), pp. 201–219 (cit. on p. 47).

[Bro14] M. Brown. “Groupthink in software engineering.” In: 5 (2014) (cit. on p. 45).

[BS93] J. Bowen, V. Stavridou. “Safety-critical systems, formal methods and standards.” In: *Software Engineering Journal* 8.4 (1993), pp. 189–209 (cit. on p. 44).

[BT05] B. Boehm, R. Turner. “Management challenges to implementing agile processes in traditional development organisations.” In: *IEEE Software* 22.5 (2005), pp. 30–39 (cit. on p. 42).

[BZL04] N. K. Baym, Y. B. Zhang, M.-C. Lin. “Social interactions across media: Interpersonal communication on the internet, telephone and face-to-face.” In: *New Media & Society* 6.3 (2004), pp. 299–318 (cit. on p. 137).

[CCA13] S. Coyle, K. Conboy, T. Acton. “Group process losses in agile software development decision making.” In: *International Journal of Intelligent Information Technologies* 9.2 (2013), pp. 38–53 (cit. on p. 56).

[CG09] L. Crispin, J. Gregory. *Agile testing: A practical guide for testers and agile teams*. Pearson Education, 2009 (cit. on p. 91).

[CGP99] E. M. Clarke, O. Grumberg, D. Peled. *Model checking*. Cambridge, MA: MIT Press, 1999 (cit. on p. 39).

[CH17] J. Cleland-Huang. “Safety stories in agile development.” In: *IEEE Software* 34.4 (2017), pp. 16–19 (cit. on p. 53).

[Che+07] M. Cherubini, G. Venolia, R. DeLine, A. J. Ko. “Let’s go to the whiteboard: How and why software developers use drawings.” In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 2007, pp. 557–566 (cit. on pp. 139, 158).

[CHR17] J. Cleland-Huang, M. Rahimi. “A case study: Injecting safety-critical thinking into graduate software engineering projects.” In: *Proceedings of the 39th International Conference on Software Engineering: Software Engineering and Education Track*. 2017, pp. 67–76 (cit. on pp. 39, 103).

[CLC04] D. Cohen, M. Lindvall, P. Costa. “An introduction to agile methods.” In: *Advances in Computers* 62.03 (2004), pp. 1–66 (cit. on p. 34).

[CMS09] K. J. Cruickshank, J. B. Michael, M.-T. Shing. “A validation metrics framework for safety-critical software-intensive systems.” In: *Proceedings of the IEEE International Conference on System of Systems Engineering*. 2009, pp. 1–8 (cit. on p. 71).

[Coc02] A. Cockburn. *Agile software development*. Boston, MA: Addison-Wesley, 2002 (cit. on pp. 26, 183).

[Coh10] M. Cohn. *Succeeding with agile: Software development using scrum*. Pearson Education, 2010 (cit. on pp. 40, 187).

[Com11a] I. E. Commission. “Functional safety of electrical/electronic/programmable electronic safety related systems.” In: *IEC 61508* (2011) (cit. on pp. 26, 42).

[Com11b] I. E. Commission. *IEC 61508: Functional safety of electrical/electronic/programmable electronic safety-related systems*. International Electrotechnical Commission, 2011 (cit. on pp. 142, 197).

[Con+11] K. Conboy, S. Coyle, X. Wang, M. Pikkarainen. “People over process: Key challenges in agile development.” In: *IEEE Software* 28 (July 2011) (cit. on p. 156).

[Cre09] J. W. Creswell. *Research design: Qualitative, quantitative, and mixed method approaches*. 3rd ed. Vol. 2. Thousand Oaks, California: Sage Publications, Thousand Oaks, California, 2009 (cit. on pp. 160, 161).

[Cut+02] R. Cutler, Y. Rui, A. Gupta, J. J. Cadiz, I. Tashev, L. He, A. Colburn, Z. Zhang, Z. Liu, S. Silverberg. “Distributed meetings: A meeting capture and broadcasting system.” In: *Proceedings of the 10th International Conference on Multimedia*. 2002, pp. 503–512 (cit. on p. 136).

[Dab+05] L. A. Dabbish, R. E. Kraut, S. Fussell, S. Kiesler. “Understanding email use: Predicting action on a message.” In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 2005, pp. 691–700 (cit. on p. 138).

[DD08] T. Dybå, T. Dingsøyr. “Empirical studies of agile software development: A systematic review.” In: *Information and Software Technology* 50.9-10 (2008), pp. 833–859 (cit. on p. 35).

[DGCA17] M. L. Drury-Grogan, K. Conboy, T. Acton. “Examining decision characteristics & challenges for agile software development.” In: *Journal of Systems and Software* 131 (2017), pp. 248–265 (cit. on p. 55).

[Din+17] T. Dingsøyr, K. Rolland, N. B. Moe, E. A. Seim. “Coordination in multi-team programmes: An investigation of the group mode in large-scale agile software development.” In: *Procedia Computer Science* 121 (2017), pp. 123–128 (cit. on p. 227).

[DMN14] K. Dobson, A Moors, B. Norris. “Literature review of safety critical communication methodologies.” In: *ERA* 01 (2014) (cit. on pp. 44, 156).

[DN02] Y. K. Djamba, W. L. Neuman. “Social research methods: Qualitative and quantitative approaches.” In: *Teaching Sociology* 30.3 (July 2002), p. 380 (cit. on p. 161).

[DSVWJ11] C. De Snoo, W. Van Wezel, R. J. Jorna. “An empirical investigation of scheduling performance criteria.” In: *Journal of Operations Management* 29.3 (2011), pp. 181–193 (cit. on p. 47).

[Dyb05] T. Dybå. “An empirical investigation of the key factors for success in software process improvement.” In: *IEEE Transactions on Software Engineering* 31.5 (2005), pp. 410–424 (cit. on p. 225).

[ELS05] J. Erickson, K. Lyytinen, K. Siau. “Agile modelling, agile software development, and extreme programming: The state of research.” In: *Journal of Database Management* 16.4 (2005), p. 88 (cit. on p. 34).

[EMT05] H. Erdogmus, M. Morisio, M. Torchiano. “On the effectiveness of the test-first approach to programming.” In: *IEEE Transactions on Software Engineering* 31.3 (2005), pp. 226–237 (cit. on p. 93).

[EP17] C. Ebert, M. Paasivaara. “Scaling agile.” In: *IEEE Software* 34.6 (2017), pp. 98–103 (cit. on p. 227).

[Eri15] C. A. Ericson. *Hazard analysis techniques for system safety*. NJ, USA: John Wiley & Sons, 2015 (cit. on pp. 36, 37).

[Fal+18a] D. Falessi, N. Juristo, C. Wohlin, B. Turhan, J. Münch, A. Jedlitschka, M. Oivo. “Empirical software engineering experts on the use of students and professionals in experiments.” In: *Empirical Software Engineering* 23.1 (2018), pp. 452–489 (cit. on pp. 85, 103).

[Fal+18b] D. Falessi, N. Juristo, C. Wohlin, B. Turhan, J. Münch, A. Jedlitschka, M. Oivo. “Empirical software engineering experts on the use of students and professionals in experiments.” In: *Empirical Software Engineering* 23.1 (2018), pp. 452–489 (cit. on p. 225).

[Fan+15] Y. Fan, Z. Li, J. Pei, H. Li, J. Sun. “A system-theoretic analysis of the "7.23" Yong-Tai-Wen railway accident.” In: *Safety Science* 76 (July 2015) (cit. on p. 27).

[FC03] C. Ferraris, R. Carveth. “NASA and the Columbia disaster: Decision-making by groupthink?” In: *Proceedings of the 2003 Association for Business Communication Annual Convention*. 2003, p. 12 (cit. on p. 56).

[Fit+13] B. Fitzgerald, K.-J. Stol, R. O’Sullivan, D. O’Brien. “Scaling agile methods to regulated environments: An industry case study.” In: *Proceedings of the 35th International Conference onSoftware Engineering*. IEEE. 2013, pp. 863–872 (cit. on p. 50).

[FL15] *Including safety during early development phases of future air traffic management concepts*. Lisbon, Portugal, 2015 (cit. on p. 27).

[FOC08] R. H. Flin, P. O’Connor, M. Crichton. *Safety at the sharp end: A guide to non-technical skills*. Ashgate Publishing, 2008 (cit. on p. 45).

[Ghe+13] C. Ghezzi, C. Menghi, A. M. Sharifloo, P. Spoletini. “On requirements verification for model refinements.” In: *Proceedings of the 21st IEEE International Requirements Engineering Conference*. 2013 (cit. on p. 52).

[GL13] J Gläser, G Laudel. “Life with and without coding: Two methods for early-stage data analysis in qualitative research aiming at causal explanations.” In: *Forum: Qualitative Social Research* 14 (Mar. 2013) (cit. on p. 161).

[GN16] T. J. Gandomani, M. Z. Nafchi. “Agile transition and adoption human-related challenges and issues: A grounded theory approach.” In: *Computers in Human Behaviour* 62 (2016), pp. 257–266 (cit. on p. 35).

[Gor+06] T. Gorschek, P. Garre, S. Larsson, C. Wohlin. “A model for technology transfer in practice.” In: *IEEE Software* 23.6 (2006), pp. 88–95 (cit. on p. 225).

[GPM10] X. Ge, R. F. Paige, J. A. McDermid. “An iterative approach for development of safety-critical software and safety arguments.” In: *Proceedings of the Agile Conference*. 2010, pp. 35–43 (cit. on pp. 26, 50, 225).

[Gre+16] P. Gregory, L. Barroca, H. Sharp, A. Deshpande, K. Taylor. “The challenges that challenge: Engaging with agile practitioners’ concerns.” In: *Information and Software Technology* 77 (2016), pp. 92–104 (cit. on p. 35).

[Gre12] D. D. Gregorio. “How the business analyst supports and encourages collaboration on agile projects.” In: *Proceedings of the International Systems Conference*. 2012, pp. 1–4 (cit. on p. 102).

[GTF17] L. Gren, R. Torkar, R. Feldt. “Group development and group maturity when building agile teams: A qualitative and quantitative investigation at eight large companies.” In: *Journal of Systems and Software* 124 (2017), pp. 104–119 (cit. on p. 56).

[Gul00] F. W. Guldenmund. “The nature of safety culture: A review of theory and research.” In: *Safety Science* 34.1-3 (2000), pp. 215–257 (cit. on p. 144).

[Ham77] R. G. Hamlet. “Testing programs with the aid of a compiler.” In: *IEEE Transactions on Software engineering* 4 (1977), pp. 279–290 (cit. on p. 93).

[Han+16] G. K. Hanssen, B. Haugset, T. Stålhane, T. Myklebust, I. Kulbrandstad. "Quality assurance in scrum applied to safety-critical software." In: *Proceedings of the 17th International Conference on Agile Software Development*. 2016, pp. 92–103 (cit. on p. 50).

[HH09] L. Huang, M. Holcombe. "Empirical investigation towards the effectiveness of test-first programming." In: *Information and Software Technology* 51.1 (2009), pp. 182–194 (cit. on p. 102).

[Hol+06] H. Holmstrom, E. Ó. Conchúir, J Agerfalk, B. Fitzgerald. "Global software development challenges: A case study on temporal, geographical and socio-cultural distance." In: *Proceedings of the International Conference on Global Software Engineering*. 2006, pp. 3–11 (cit. on p. 150).

[HRH13] M. Hummel, C. Rosenkranz, R. Holten. "The role of communication in agile systems development." In: *Business & Information Systems Engineering* 5.5 (2013), pp. 343–355 (cit. on pp. 55, 135, 156).

[HRW00] M. Höst, B. Regnell, C. Wohlin. "Using students as subjects - A comparative study of students and professionals in lead-time impact assessment." In: *Empirical Software Engineering* 5.3 (2000), pp. 201–214 (cit. on p. 85).

[HS12] B. Haugset, T. Stalhane. "Automated acceptance testing as an agile requirements engineering practice." In: *Proceedings of the 45th Hawaii International Conference on System Science*. 2012, pp. 5289–5298 (cit. on p. 100).

[HS14] M. Hammarberg, J. Sunden. *Kanban in action*. Manning Publications Co., 2014 (cit. on p. 35).

[Hum89] W. S. Humphrey. *Managing the software process*. Boston, MA, USA: Addison-Wesley, 1989 (cit. on p. 225).

[HWS17] G. K. Hanssen, G. Wedzinga, M. Stuip. "An assessment of avionics software development practice: Justifications for an agile development process." In: *Proceedings of the 18th International Conference on Agile Software Development*. 2017 (cit. on p. 50).

[Jan08] I. L. Janis. "Groupthink." In: *IEEE Engineering Management Review* 36.1 (2008), p. 36 (cit. on pp. 45, 46).

[Jan71] I. L. Janis. “Groupthink.” In: *Psychology Today* 5.6 (1971), pp. 43–46 (cit. on p. 150).

[Joh+94] J. D. Johnson, W. A. Donohue, C. K. Atkin, S. Johnson. “Differences between formal and informal communication channels.” In: *The Journal of Business Communication* 31.2 (1994), pp. 111–122 (cit. on p. 44).

[Key17] J. Keyton. “Communication in organisations.” In: *Annual Review of Organisational Psychology and Organisational Behaviour* (2017) (cit. on p. 44).

[Kit+17] B. Kitchenham, L. Madeyski, D. Budgen, J. Keung, P. Brereton, S. Charters, S. Gibbs, A. Pohthong. “Robust statistical methods for empirical software engineering.” In: *Empirical Software Engineering* 22.2 (2017), pp. 579–630 (cit. on p. 103).

[Kit08] B. Kitchenham. “The role of replications in empirical software engineering? A word of warning.” In: *Empirical Software Engineering* 13.2 (2008), pp. 219–221 (cit. on p. 105).

[KLM17] Y. Khmelevsky, X. Li, S. Madnick. “Software development using agile and scrum in distributed teams.” In: *Proceedings of the Annual IEEE International Systems Conference*. IEEE. 2017, pp. 1–4 (cit. on p. 227).

[Kni02] J. C. Knight. “Safety-critical systems: Challenges and directions.” In: *Proceedings of the 24th International Conference on Software Engineering*. 2002, pp. 547–550 (cit. on p. 26).

[Kra+90] R. E. Kraut, R. S. Fish, R. W. Root, B. L. Chalfonte. “Informal communication in organisations: Form, function, and technology.” In: *Human Reactions to Technology: Claremont Symposium on Applied Social Psychology*. 1990, pp. 145–199 (cit. on pp. 138, 155).

[Kra98] R. M. Kramer. “Revisiting the Bay of Pigs and Vietnam decisions 25 years later: How well has the groupthink hypothesis stood the test of time?” In: *Organizational Behavior and Human Decision Processes* 73.2-3 (1998), pp. 236–271 (cit. on p. 45).

[Kru04] P. Kruchten. *The rational unified process: An introduction*. Addison-Wesley Professional, 2004 (cit. on p. 26).

[KS92] S. Kiesler, L. Sproull. “Group decision making and communication technology.” In: *Organisational Behaviour and Human Decision Processes* 52.1 (1992), pp. 96–123 (cit. on p. 137).

[Kuh+17] M. Kuhrmann, P. Diebold, J. Münch, P. Tell, V. Garousi, M. Felderer, K. Trektere, F. McCaffery, O. Linssen, E. Hanser, et al. “Hybrid software and system development in practice: Waterfall, scrum, and beyond.” In: *Proceedings of the International Conference on Software and System Process*. ACM. 2017, pp. 30–39 (cit. on p. 223).

[KW04] T. Kelly, R. Weaver. “The goal structuring notation - A safety argument notation.” In: *Proceedings of the Dependable Systems and Networks*. 2004, p. 6 (cit. on p. 71).

[Lar74] W. Larsen. “Fault Tree Analysis.” In: (1974, Retrieved 2014) (cit. on pp. 27, 37).

[Lev11] N. Leveson. *Engineering a safer world: Systems thinking applied to safety*. Cambridge, MA: MIT Press, 2011 (cit. on pp. 27, 36–38, 54, 62, 115, 116, 144, 148, 152, 175, 176, 179, 183).

[Luc04] T. Lucey. *Management information systems*. London: Thomson Learning, 2004 (cit. on p. 146).

[Mad10] L. Madeyski. “The impact of test-first programming on branch coverage and mutation score indicator of unit tests: An experiment.” In: *Information and Software Technology* 52.2 (2010), pp. 169–184 (cit. on p. 93).

[MAD12] N. B. Moe, A. Aurum, T. Dybå. “Challenges of shared decision-making: A multiple case study of agile software development.” In: *Information and Software Technology* 54.8 (2012), pp. 853–865 (cit. on p. 75).

[Mar99] B. Marick. “How to misuse code coverage.” In: *Proceedings of the 16th International Conference on Testing Computer Software*. 1999, pp. 16–18 (cit. on p. 93).

[MB09] J. McAvoy, T. Butler. “The role of project management in ineffective decision making within agile software development projects.” In: *European Journal of Information Systems* 18.4 (2009), pp. 372–383 (cit. on p. 56).

[MG17] L. E. Martins, T. Gorschek. “Requirements engineering for safety-critical systems: Overview and challenges.” In: *IEEE Software* 34.99 (Jan. 2017), pp. 49–57 (cit. on pp. 28, 52).

[Mon16] D. R. Montes. “Using STPA to inform developmental product testing.” PhD thesis. Massachusetts Institute of Technology, 2016 (cit. on p. 116).

[MPB16] A. Martini, L. Pareto, J. Bosch. “A multiple case study on the inter-group interaction speed in large, embedded software companies employing agile.” In: *Journal of Software: Evolution and Process* 28.1 (2016), pp. 4–26 (cit. on p. 55).

[MS16] T. Myklebust, T. Stålhane. “Safety stories - A new concept in agile development.” In: *Proceedings of the International Conference on Computer Safety, Reliability, and Security*. 2016 (cit. on pp. 53, 78, 115).

[MS18] T. Myklebust, T. Stålhane. *The Agile safety case*. Switzerland: Springer, 2018 (cit. on pp. 50, 223).

[MSL15] T. Myklebust, T. Stålhane, N. Lyngby. “Application of an agile development process for EN 50128/railway conformant software.” In: *Safety and Reliability of Complex Engineered Systems* (2015) (cit. on pp. 50, 202).

[MSL16] T. Myklebust, T. Stålhane, N. Lyngby. “The agile safety plan.” In: *PSAM13* (Oct. 2016) (cit. on pp. 53, 77, 123).

[Mul+94] B. Mullen, T. Anthony, E. Salas, J. E. Driskell. “Group cohesiveness and quality of decision making: An integration of tests of the groupthink hypothesis.” In: *Small Group Research* 25.2 (1994), pp. 189–204 (cit. on p. 45).

[Myk+14] T. Myklebust, T. Stålhane, G. Hanssen, T Wien, B Haugset. “Scrum, documentation and the IEC 61508-3: 2010 software standard.” In: *Proceedings of the International Conference on Probabilistic Safety Assessment and Management*. 2014 (cit. on p. 50).

[NG14] T. Nesheim, L. J. Gressgård. “Knowledge sharing in a complex organisation: Antecedents and safety effects.” In: *Safety Science* 62 (2014), pp. 28–36 (cit. on p. 144).

[NH17] P. A. Nielsen, L. T. Heeager. “The dynamics of agile practices for safety-critical software development.” In: *Proceedings of the XP2017 Scientific Workshops*. 2017, p. 21 (cit. on p. 53).

[Ols+92] G. M. Olson, J. S. Olson, M. R. Carter, M. Storrosten. “Small group design meetings: An analysis of collaboration.” In: *Human Computer Interaction* 7.4 (1992), pp. 347–374 (cit. on p. 136).

[Org13] I. C. A. Organisation. “Safety management manual.” In: *SMM* (2013) (cit. on p. 42).

[Pai+11] R. F. Paige, A. Galloway, R. Charalambous, X. Ge, P. J. Brooke. “High-integrity agile processes for the development of safety critical software.” In: *International Journal of Critical Computer-Based Systems* 2.2 (2011), pp. 181–216 (cit. on p. 53).

[PF01] S. R. Palmer, M. Felsing. *A practical guide to feature-driven development*. Pearson Education, 2001 (cit. on pp. 26, 35).

[PGP08] F. J. Pino, F. García, M. Piattini. “Software process improvement in small and medium software enterprises: A systematic review.” In: *Software Quality Journal* 16.2 (2008), pp. 237–261 (cit. on p. 143).

[Pik+08] M. Pikkarainen, J. Haikara, O. Salo, P. Abrahamsson, J. Still. “The impact of agile practices on communication in software development.” In: *Empirical Software Engineering* 13.3 (2008), pp. 303–337 (cit. on pp. 55, 118).

[Pol+17] A. Poller, L. Kocksch, S. Türpe, F. A. Epp, K. Kinder-Kurlanda. “Can security become a routine?: A study of organisational change in an agile software development group.” In: *Proceedings of the 20th ACM Conference on Computer-Supported Cooperative Work and Social Computing*. 2017, pp. 2489–2503 (cit. on p. 83).

[Pop07] M. Poppendieck. “Lean software development.” In: *Proceedings of the 29th International Conference on Software Engineering Companion*. 2007, pp. 165–166 (cit. on pp. 26, 35).

[Pri10] S. Prineas. “Safety-critical communication.” In: *Handbook of communication in anaesthesia & critical care: A practical guide to exploring the art* (2010), p. 189 (cit. on p. 44).

[Rav98] B. H. Raven. “Groupthink, bay of pigs, and watergate reconsidered.” In: *Organizational Behavior and Human Decision Processes* 73.2-3 (1998), pp. 352–361 (cit. on p. 45).

[RH09] P. Runeson, M. Höst. “Guidelines for conducting and reporting case study research in software engineering.” In: *Empirical Software Engineering* 14.2 (2009), p. 131 (cit. on pp. 68, 116, 167).

[RM13] Y. Rafique, V. B. Mišić. “The effects of test-driven development on external quality and productivity: A meta-analysis.” In: *IEEE Transactions on Software Engineering* 39.6 (2013), pp. 835–856 (cit. on p. 102).

[Ros11] J. D. Rose. “Diverse perspectives on the groupthink theory - A literary review.” In: *Emerging Leadership Journeys* 4.1 (2011), pp. 37–57 (cit. on p. 45).

[RR08] P. A. Rottier, V. Rodrigues. “Agile development in a medical device company.” In: *Proceedings of the Agile Conference*. 2008 (cit. on p. 52).

[Rub12] K. S. Rubin. *Essential scrum: A practical guide to the most popular agile process*. Michigan, USA: Addison-Wesley, 2012 (cit. on pp. 35, 36, 83, 119, 123, 187, 189).

[Run+12] P. Runeson, M. Höst, A. Rainer, B. Regnell. *Case study research in software engineering: Guidelines and examples*. Hoboken, New Jersey: John Wiley & Sons, 2012 (cit. on p. 128).

[SB02] K. Schwaber, M. Beedle. *Agile software development with scrum*. Vol. 1. Upper Saddle River, New Jersey: Prentice Hall, 2002 (cit. on pp. 34, 35, 44).

[SB06] A. M. Saks, M. Belcourt. “An investigation of training activities and transfer of training in organisations.” In: *Human Resource Management* 45.4 (2006), pp. 629–648 (cit. on p. 138).

[SC94] A. Strauss, J. Corbin. “Grounded theory methodology.” In: *Handbook of Qualitative Research* 17 (1994), pp. 273–285 (cit. on p. 133).

[SC97] A. Strauss, J. Corbin. *Grounded theory in practice*. Sage, 1997 (cit. on pp. 72, 169).

[Sca+16] G. Scanniello, S. Romano, D. Fucci, B. Turhan, N. Juristo. “Students’ and professionals’ perceptions of test-driven development: A focus group study.” In: *Proceedings of the 31st Annual ACM Symposium on Applied Computing*. 2016, pp. 1422–1427 (cit. on p. 99).

[Sch04] K. Schwaber. *Agile project management with scrum*. Microsoft Press, 2004 (cit. on p. 35).

[Sch10] E. H. Schein. *Organisational culture and leadership*. Vol. 2. John Wiley & Sons, 2010 (cit. on p. 183).

[Sch95] K. Schwaber. “Scrum development process.” In: *OOPSLA’95 Workshop on Business Object Design and Implementation*. 1995, pp. 117–134 (cit. on pp. 26, 35).

[Sel09] B. Selic. “Agile documentation, anyone?” In: *IEEE software* 26.6 (2009) (cit. on p. 53).

[SG+17] M.-L. Sanchez-Gordon, A. de Amescua, R. V. Oconnor, X. Larrucea. “A standard-based framework to integrate software work in small settings.” In: *Computer Standards & Interfaces* 54 (2017), pp. 162–175 (cit. on p. 226).

[SH11] C. J. Stettina, W. Heijstek. “Necessary and neglected? An empirical study of internal documentation in agile software development teams.” In: *Proceedings of the 29th ACM International Conference on Design of Communication*. 2011, pp. 159–166 (cit. on p. 53).

[Shu+08] F. J. Shull, J. C. Carver, S. Vegas, N. Juristo. “The role of replications in empirical software engineering.” In: *Empirical software engineering* 13.2 (2008), pp. 211–218 (cit. on p. 105).

[SKM13] T. Stålhane, V. Katta, T. Myklebust. “Scrum and IEC 60880.” In: *Enlarged Halden Reactor Project Meeting*. Storefjell, Norway, 2013 (cit. on p. 50).

[SM14] S. Shafiq, N. M. Minhas. “Integrating formal methods in XP - A conceptual solution.” In: *Journal of Software Engineering and Applications* 7.04 (2014), p. 299 (cit. on p. 52).

[SM16a] T. Stålhane, T. Myklebust. “Agile safety analysis.” In: *ACM SIGSOFT Software Engineering Notes* (2016) (cit. on pp. 51, 52).

[SM16b] T. Stålhane, T. Myklebust. “Early safety analysis.” In: *Proceedings of the 17th International Conference on Agile Software Development*. 2016, pp. 1–6 (cit. on p. 115).

[SMH12] T Stålhane, T. Myklebust, G. Hanssen. “The application of Safe Scrum to IEC 61508 certifiable software.” In: *Proceedings of the 11th International Probabilistic Safety Assessment and Management Conference and the Annual European Safety and Reliability Conference*. 2012, pp. 6052–6061 (cit. on pp. 26, 50, 53, 223, 226).

[SS11] K. Schwaber, J. Sutherland. “The scrum guide.” In: *Scrum Alliance* 268 (2011) (cit. on p. 35).

[Sta03a] D. H. Stamatis. *Failure mode and effect analysis: FMEA from theory to execution*. Milwaukee, WI: ASQ Quality Press, 2003 (cit. on pp. 27, 37).

[Sta03b] J. Stapleton. *DSDM: Business focused development*. Pearson Education, 2003 (cit. on p. 35).

[Sta06] E. Standards. “Railway applications - The specification and demonstration of reliability, availability, maintainability and safety.” In: *EN 50126* (2006) (cit. on p. 53).

[Sta11] I. O. for Standardisation. “Functional safety of electrical and/or electronic systems in production automobiles.” In: *ISO 26262* (2011) (cit. on pp. 26, 42, 172).

[Sta97] J. Stapleton. *DSDM, dynamic systems development method: The method in practice*. Cambridge University Press, 1997 (cit. on pp. 26, 34, 35).

[Sto+10] M.-A. Storey, C. Treude, A. van Deursen, L.-T. Cheng. “The impact of social media on software engineering practices and tools.” In: *Proceedings of the FSE/SDP Workshop on Future of Software Engineering Research*. 2010, pp. 359–364 (cit. on pp. 139, 157).

[Sto+17] M.-A. Storey, A. Zagalsky, F. Figueira Filho, L. Singer, D. M. German. “How social and communication channels shape and challenge a participatory culture in software development.” In: *IEEE Transactions on Software Engineering* 43.2 (2017), pp. 185–204 (cit. on pp. 137, 139, 155, 156).

[Stå+14] T. Stålhane, G. K. Hanssen, T. Myklebust, B. Haugset. “Agile change impact analysis of safety-critical software.” In: *Proceedings of the International Conference on Computer Safety, Reliability, and Security*. 2014, pp. 444–454 (cit. on p. 50).

[SW06] W. Schiano, J. W. Weiss. “Y2K all over again: How groupthink permeates IS and compromises security.” In: *Business Horizons* 49.2 (2006), pp. 115–125 (cit. on p. 56).

[TH90] P. T Hart. *Groupthink in government: A study of small groups and policy failure.* Swets & Zeitlinger Publishers, 1990 (cit. on p. 45).

[The+15] G. Theocharis, M. Kuhrmann, J. Münch, P. Diebold. “Is water-scrum-fall reality? On the use of agile and traditional development practices.” In: *Proceedings of the International Conference on Product-Focused Software Process Improvement*. 2015, pp. 149–166 (cit. on p. 85).

[Tic00] W. F. Tichy. “Hints for reviewing empirical work in software engineering.” In: *Empirical Software Engineering* 5.4 (2000), pp. 309–312 (cit. on p. 85).

[Vas+17] F. J. Vasconcellos, G. B. Landre, J. A. O. Cunha, J. L. Oliveira, R. A. Ferreira, A. M. Vincenzi. “Approaches to strategic alignment of software process improvement: A systematic literature review.” In: *Journal of Systems and Software* 123 (2017), pp. 45–63 (cit. on p. 225).

[Vil+17] J. Vilela, J. Castro, L. E. G. Martins, T. Gorschek. “Integration between requirements engineering and safety analysis: A systematic literature review.” In: *Journal of Systems and Software* 125 (2017), pp. 68–92 (cit. on pp. 54, 155, 184).

[Voi+16] S. Voigt, J. von Garrel, J. Müller, D. Wirth. “A study of documentation in agile software projects.” In: *Proceedings of the 10th International Symposium on Empirical Software Engineering and Measurement*. 2016, p. 4 (cit. on p. 53).

[Vuo11] M. Vuori. “Agile development of safety-critical software.” In: *Tampere University of Technology* 14 (2011) (cit. on pp. 50, 53).

[WBW17] Y. Wang, I. Bogicevic, S. Wagner. “A study of safety documentation in a scrum development process.” In: *Proceedings of the XP 2017 Scientific Workshops*. 2017, p. 22 (cit. on p. 152).

[WC03] L. Williams, A. Cockburn. “Agile software development: It’s about feedback and change.” In: *Computer* 36.6 (2003), pp. 39–43 (cit. on p. 34).

[WH12] M. Wynne, A. Hellesoy. *The cucumber book: Behaviour-driven development for testers and developers*. Pragmatic Bookshelf, 2012 (cit. on pp. 41, 77).

[Whe09] S. A. Wheelan. “Group size, group development, and group productivity.” In: *Small Group Research* 40.2 (2009), pp. 247–262 (cit. on p. 176).

[WK02] L. Williams, R. Kessler. *Pair programming illuminated*. Boston, MA, USA: Addison-Wesley, 2002 (cit. on p. 44).

[Woh+12] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, A. Wesslén. *Experimentation in software engineering*. Springer Science & Business Media, 2012 (cit. on p. 91).

[WRC10] L. Williams, K. Rubin, M. Cohn. “Driving process improvement via comparative agility assessment.” In: *Proceedings of the Agile Conference*. 2010, pp. 3–10 (cit. on pp. 71, 84).

[WW18a] Y. Wang, S. Wagner. “Combining STPA and BDD for safety analysis and verification in agile development: A controlled experiment.” In: *Proceedings of the 19th International Conference on Agile Software Development*. 2018, pp. 37–53 (cit. on p. 111).

[WW18b] Y. Wang, S. Wagner. “On groupthink in safety analysis: An industrial case study.” In: *Proceedings of the 40th International Conference on Software Engineering*. 2018 (cit. on p. 150).

[Yin13] R. K. Yin. *Case study research: Design and methods*. Sage, 2013 (cit. on pp. 68, 167).

[YJC09] J. Yoo, E. Jee, S. Cha. “Formal modeling and verification of safety-critical software.” In: *IEEE software* 26.3 (2009), pp. 42–49 (cit. on p. 39).

List of Figures

List of Tables