Mälardalen University Doctoral Dissertation 280

Contracts-based Maintenance of Safety Cases

Omar Jaradat

MÄLARDALEN UNIVERSITY
SWEDEN

Mälardalen University Press Dissertations
No. 280

CONTRACTS-BASED MAINTENANCE OF SAFETY CASES

Omar Jaradat

2018

MÄLARDALEN UNIVERSITY
SWEDEN

School of Innovation, Design and Engineering

ISBN 978-91-7485-417-6
ISSN 1651-4238
Printed by E-Print AB, Stockholm, Sweden

Mälardalen University Press Dissertations
No. 280

CONTRACTS-BASED MAINTENANCE OF SAFETY CASES

Omar Jaradat

Akademisk avhandling

som för avläggande av teknologie doktorsexamen i datavetenskap vid Akademin för innovation, design och teknik kommer att offentligen försvaras måndagen den 3 december 2018, 09.30 i Kappa, Mälardalens högskola, Västerås.

Fakultetsopponent: Senior Lecturer Mark Nicholson, University of York

MÄLARDALEN UNIVERSITY
SWEDEN

Akademin för innovation, design och teknik

Abstract
Safety critical systems are those systems whose failure could result in loss of life, significant property damage, or damage to the environment. System safety is a major property that shall be adequately assured to avoid any severe outcomes in safety critical systems. Safety assurance should provide justified confidence that all potential risks due to system failures are either eliminated or acceptably mitigated. System developers in many domains (e.g., automotive, avionics, railways) should provide convincing arguments regarding the safe performance of their systems to a national or international regulatory authority and obtain approvals before putting the system into service. Building 'Safety cases' is a proven technique to argue about and communicate systems' safety and it has become a common practice in many safety critical system domains. System developers use safety cases to articulate claims about how systems meet their safety requirements and objectives, collect and document items of evidence, and construct a safety argument to show how the available items of evidence support the claims.

Safety critical systems are evolutionary and constantly subject to preventive, perfective, corrective or adaptive changes during both the development and operational phases. Changes to any part of those systems can undermine the confidence in safety since changes can refute articulated claims about safety or challenge the supporting evidence on which this confidence relies. Hence, safety cases need to be built as living documents that should always be maintained to justify the safety status of the associated system and evolve as these systems evolve. However, building safety cases are costly since they require a significant amount of time and efforts to define the safety objectives, generate the required evidence and conclude the underlying logic behind the safety case arguments. Safety cases document highly dependent elements such as safety goals, assumptions and evidence. Seemingly minor changes may have a major impact. Changes to a system or its environment can necessitate a costly and painstaking impact analysis for systems and their safety cases. In addition, changes may require system developers to generate completely new items of evidence by repeating the verification activities. Therefore, changes can exacerbate the cost of producing and maintaining safety cases.

Safety contracts have been proposed as a means for helping to manage changes. There have been works that discuss the usefulness of contracts for reusability and maintainability. However, there has been little attention on how to derive them and how exactly they can be utilised for system or safety case maintenance.

The main goal of this thesis is to support the change impact analysis as a key factor to enhance the maintainability of safety cases. We focus on utilising safety contracts to achieve this goal. To address this, we study how safety contracts can support essential factors for any useful change management process, such as (1) identifying the impacted elements and those that are not impacted, (2) minimising the number of impacted safety case elements, and (3) reducing the work needed to make the impacted safety case elements valid again. The preliminary finding of our study reveals that using safety contracts can be promising to develop techniques and processes to facilitate safety case maintenance. The absence of safety case maintenance guidelines from safety standards and the lack of systematic and methodical maintenance techniques have motivated the work of this thesis. Our work is presented through a set of developed and assessed techniques, where these techniques utilise safety contracts to achieve the overall goal by various contributions. We begin by a framework for evaluation of the impact of change on safety critical systems and safety cases. Through this, we identify and highlight the most sensitive system components to a particular change. We propose new ways to associate system design elements with safety case arguments to enable traceability. How to identify and reduce the propagation of change impact is addressed subsequently. Our research also uses safety contracts to enable through-life safety assurance by monitoring and detecting any potential mismatch between the design safety assumptions and the actual behaviour of the system during its operational phase. More specifically, we use safety contracts to capture thresholds of selected safety requirements and compare them with the runtime related data (i.e., operational data) to continuously assess and evolve the safety arguments.

In summary, our proposed techniques pave the way for cost-effective maintenance of safety cases upon preventive, perfective, corrective or adaptive changes in safety critical systems thus helping better decision support for change impact analysis.

ISBN 978-91-7485-417-6
ISSN 1651-4238

Abstract

Safety critical systems are those systems whose failure could result in loss of life, significant property damage, or damage to the environment. System safety is a major property that shall be adequately assured to avoid any severe outcomes in safety critical systems. Safety assurance should provide justified confidence that all potential risks due to system failures are either eliminated or acceptably mitigated. System developers in many domains (e.g., automotive, avionics, railways) should provide convincing arguments regarding the safe performance of their systems to a national or international regulatory authority and obtain approvals before putting the system into service. Building 'Safety cases' is a proven technique to argue about and communicate systems' safety and it has become a common practice in many safety critical system domains. System developers use safety cases to articulate claims about how systems meet their safety requirements and objectives, collect and document items of evidence, and construct a safety argument to show how the available items of evidence support the claims.

Safety critical systems are evolutionary and constantly subject to preventive, perfective, corrective or adaptive changes during both the development and operational phases. Changes to any part of those systems can undermine the confidence in safety since changes can refute articulated claims about safety or challenge the supporting evidence on which this confidence relies. Hence, safety cases need to be built as living documents that should always be maintained to justify the safety status of the associated system and evolve as these systems evolve. However, building safety cases are costly since they require a significant amount of time and efforts to define the safety objectives, generate the required evidence and conclude the underlying logic behind the safety case arguments. Safety cases document highly dependent elements such as safety goals, assumptions and evidence. Seemingly minor changes may have a major impact. Changes to a system or its environment can necessitate a costly and painstaking impact analysis for systems and their safety cases. In addition, changes may require system developers to generate completely new items of evidence by repeating the verification activities. Therefore, changes can exacerbate the cost of producing and maintaining safety cases.

Safety contracts have been proposed as a means for helping to manage changes. There have been works that discuss the usefulness of contracts for reusability and maintainability. However, there has been little attention on how to derive them and how exactly they can be utilised for system or safety case maintenance.

The main goal of this thesis is to support the change impact analysis as a key factor to enhance the maintainability of safety cases. We focus on utilising safety contracts to achieve this goal. To address this, we study how safety contracts can support essential factors for any useful change management process, such as (1) identifying the impacted elements and those that are not impacted, (2) minimising the number of impacted safety case elements, and (3) reducing the work needed to make the impacted safety case elements valid again. The preliminary finding of our study reveals that using safety contracts can be promising to develop techniques and processes to facilitate safety case maintenance. The absence of safety case maintenance guidelines from safety standards and the lack of systematic and methodical maintenance techniques have motivated the work of this thesis. Our work is presented through a set of developed and assessed techniques, where these techniques utilise safety contracts to achieve the overall goal by various contributions. We begin by a framework for evaluation of the impact of change on safety critical systems and safety cases. Through this, we identify and highlight the most sensitive system components to a particular change. We propose new ways to associate system design elements with safety case arguments to enable traceability. How to identify and reduce the propagation of change impact is addressed subsequently. Our research also uses safety contracts to enable through-life safety assurance by monitoring and detecting any potential mismatch between the design safety assumptions and the actual behaviour of the system during its operational phase. More specifically, we use safety contracts to capture thresholds of selected safety requirements and compare them with the runtime related data (i.e., operational data) to continuously assess and evolve the safety arguments.

In summary, our proposed techniques pave the way for cost-effective maintenance of safety cases upon preventive, perfective, corrective or adaptive changes in safety critical systems thus helping better decision support for change impact analysis.

Swedish Summary

Säkerhetskritiska system är system där fel kan resultera i förlust av människoliv, betydande skada på egendom, eller skador på miljön. Systemsäkerhet är en viktig egenskap som måste säkerställas för att minimera riskerna för allvarliga fel i säkerhetskritiska system. Säkerställande av systemsäkerheten bör resultera i väl underbyggda argument för att alla potentiella risker som orsakas av systemfel eliminerats eller reducerats till en acceptabel nivå. Systemutvecklare inom många områden (t.ex. bilar, flyg, järnvägar) behöver tillhandahålla information om systemens säkerhetsstatus till en nationell eller internationell tillsynsmyndighet för att denna ska kunna bedöma systemet kan sättas i drift eller inte. Mer specifikt bör systemutvecklare skapa ”säkerhetsfall” bestående av (1) krav som om de är uppfyllda leder till att systemen är tillräckligt säkra och (2) bevis för att dessa krav är uppfyllda, i form av väl dokumenterade bevismaterial och säkerhetsargument som tydligt visar att detta bevismaterial innebär att kraven är uppfyllda. Att på detta sätt konstruera säkerhetsfall är en beprövad teknik för att argumentera för och kommunicera systemens säkerhet och är praxis inom många säkerhetskritiska områden.

Många säkerhetskritiska system är under ständig utveckling och föremål för förebyggande, förbättrande, och korrigerande förändringar under såväl utvecklings- som driftsfasen. ändringar av någon del av dessa system kan undergräva förtroendet för säkerheten, eftersom de kan ändra förutsättningarna för påståenden om säkerhet eller utmana de stödjande bevis som detta förtroende bygger på. Därför är säkerhetsfall byggda som levande dokument som ständigt behöver hållas uppdaterade för att motivera säkerhetsstatus för systemet. Konstruktion och underhåll av säkerhetsfall är dock kostsamt eftersom det krävs betydande tid och ansträngningar för att definiera säkerhetskraven, generera de nödvändiga bevisen och utforma den underliggande logiken bakom säkerhetsargumenten. Säkerhetsfall dokumenterar starkt ömsesidiga beroenden (t.ex. mellan säkerhetskrav, bevis och antaganden) och även mindre förändringar kan ha stor inverkan. Förändringar i ett system eller dess miljö kan kräva en dyr och noggrann säkerhetsanalys för system och dess säkerhetsfall. Dessutom kan ändringar kräva att systemutvecklare genererar helt nya bevismaterial. Därför kan förändringar väsentligt öka kostnaden för att producera och bibehålla säkerhetsfall.

Säkerhetskontrakt har föreslagits som ett medel för att hjälpa till att hantera förändringar. Det finns forskning som diskuterar användbarheten av kontrakt för återanvändning och underhåll, men hur de ska härledas och exakt hur de kan användas vid systemunderhåll har fått mindre uppmärksamhet.

Huvudsyftet med denna avhandling är att ge stöd för analys av de effekter som systemförändringar har på systemsäkerheten. Vi använder säkerhetskontrakt för att uppnå detta mål. Specifikt studerar vi hur säkerhetskontrakt kan stödja analys av väsentliga faktorer i förändringshanteringen, såsom (1) identifiering av vilka delar som påverkas, respektive inte påverkas, (2) hur antalet påverkade delar kan minimeras och (3) hur det arbete som behövs för att göra säkerhetsfallet giltiga igen kan minimeras. Våra resultat indikerar att användandet av säkerhetskontrakt är en lovande metod för att utveckla tekniker och processer som underlättar underhåll av säkerhetsfall. Frånvaron av stöd och riktlinjer för detta i säkerhetsstandarder och brist på systematiska och metodiska underhållstekniker har motiverat denna avhandling. Vårt arbete presenteras i form av en uppsättning nyutvecklade och utvärderade metoder som använder säkerhetskontrakt för att uppnå det övergripande målet.

Den första metoden utgörs av ett ramverk för utvärdering av förändringars inverkan på säkerhetskritiska system och deras säkerhetsfall, vilket låter oss identifierar de systemkomponenter som är mest känsliga för en viss förändring. För att öka spårbarheten föreslår vi även nya sätt att associera systemkomponenter till specifika delar av motsvarande säkerhetsfall. Vårt nästa bidrag fokuserar på hur spridningen av effekterna av en förändring kan minskas. Vi använder säkerhetskontrakt för att säkerställa säkerheten under systemets hela livscykel. Genom övervakning kan vi upptäcka brister i överensstämmelsen mellan säkerhetsantaganden och systemets faktiska beteende under drift. Mer specifikt använder vi säkerhetskontrakt för att identifiera kritiska trösklar för utvalda säkerhetskrav och jämför dessa med motsvarande data (dvs operativa data) under drift för att kontinuerligt utvärdera och skapa förutsättningar för utveckling av säkerhetsfallen.

Sammanfattningsvis visar våra föreslagna metoder på en väg mot kostnadseffektivt underhåll av säkerhetsfall vid förebyggande, korrigerande eller adaptiv förändring i säkerhetskritiska system, vilket bidrar till bättre stöd för beslut i förändringsarbetet.

“O’ Lord! Increase me in knowledge”
Holy Quran (20:114)

Acknowledgments

First and foremost, I am deeply grateful to my supervisors, Sasikumar Punnekkat, Iain Bate and Hans Hansson. Without your continuous help and support this thesis would not be possible. Sasikumar, you are a big source of hope and talking to you is always a successful way for me to think positively and make more educated decisions. Iain, you always help me to build a stronger self confidence and never underestimate what I can do, I owe you a debt of gratitude for all you have done for me. I want to express my gratitude to Kristina Lundqvist for her encouragement, recommendations and support during my master and PhD studies. Next, I want to thank Patrick Graydon[1], your patience, discussions and opinions are truly constructive and appreciated. Thank you all for supporting me in taking this PhD and for believing in me.

This thesis is the culmination of a long journey which was just like climbing a high peak step by step. This journey was accompanied with hardship, stress and frustration, and without the endless love, support and continuous encouragement of my parents, the peak was never reachable. Many thanks to my strong father and to my wonderful mother, I love you and always will do. Rawan, you are a great wife who is always, without hesitation, ready to motivate me whenever I am down, thanks for having my back! My sons Rayyan and Ibrahim, you are the secret of my patience to keep moving forward. I am sorry guys for ruining many of your weekends and school breaks. I promised you before to try not to do it again and I failed but I am asking for one more chance now. Special thanks to my lovely sister Arwa and my dear brothers Mohammad, Ahmad, Abdallah and Ali you are always there when I need you. I will probably be in trouble if I forget to thank my parents-in-law, thanks a lot for your continuous support, encouragement and delicious food.

In the same day when I set off on my PhD journey, Irfan Šljivo was another

[1]Patrick was my advisor since I started my PhD studies in Sep 2012 until Nov 2014

candidate who was setting off on his PhD journey at the same office. Since then Irfan and I became journey companions, officemates, project mates and friends or even brothers. We shared unforgettable good and tough times, stress, frustration, project trips, conferences, etc. A tremendous thank you goes to you Irfan for the memorable companionship. You have always been there to lend a helping hand when I stumble (I hope I did the same for you). Of course, I cannot forget one of my best friends Gabriel Campeanu who joined two journeys with me, a colleague during our MSc studies and an officemate during the PhD work. A very big thank you goes to you Gabriel, you never hesitate to help your friends whenever they need you.

I further thank all my co-authors and colleagues with whom I had the pleasure to work with during this time: Sasikumar Punnekkat, Iain Bate, Irfan Šljivo, Ibrahim Habli, Richard Hawkins, Abdallah Salameh, Svetlana Girs, Elena Lisova, Mohammad Ashjaei, Kester Clegg , Lorenzo Corneo, Vincenzo Gulisano and Yiannis Nikolakopoulos.

Next, I would like to thank the head of our division Radu Dobrin for his tips and support. I also want to thank the administrative staff, Malin Rosqvist, Carola Ryttersson, Sofia Jäderén, Susanne Fronnå, et al., for facilitating all paperworks and routines. I would like to thank all researchers at Mälardalen University for the wonderful moments we have shared in lectures, meetings and fika time (coffee breaks). I also owe a great debt of gratitude to my project mates (members of SYNOPSIS, SafeCOP and FiC) for fruitful meetings, discussions, disputes and support. I cannot leave out my office mates and friends, Husni Khanfar, Irfan Šljivo, Gabriel Campeanu, Filip Markovic and Julieth Patricia Castellanos Ardila. I want to thank the football gang who was warming up the cold and lazy weekends. Special thank you goes to Radu Dobrin for organising the games and for my brother-in-law Zaid Darwish for motivating me every week to join.

The work in this thesis has been supported by the Swedish Foundation for Strategic Research (SSF) via the projects SYNOPSIS[2] and FIC[3] as well as EU and VINNOVA via SafeCOP[4] project.

Omar T. Jaradat
October, 2018
Västerås, Sweden

[2]http://www.es.mdh.se/SYNOPSIS/
[3]http://www.es.mdh.se/fic
[4]http://www.safecop.eu/

List of Publications

Papers Included in the PhD Thesis

Paper A *Using Sensitivity Analysis to Facilitate The Maintenance of Safety Cases*, Omar Jaradat, Iain Bate, Sasikumar Punnekkat, In Proceedings of the 20th International Conference on Reliable Software Technologies (Ada-Europe), June 2015.

Paper B *Deriving Hierarchical Safety Contracts*, Omar Jaradat, Iain Bate, In Proceedings of the 21st IEEE Pacific Rim International Symposium on Dependable Computing (PRDC), Nov 2015.

Paper C *Using Safety Contracts to Guide the Maintenance of Systems and Safety Cases*, Omar Jaradat, Iain Bate, In Proceedings of the 13rd European Dependable Computing Conference (EDCC), Sep 2017.

Paper D *Using Safety Contracts to Verify Design Assumptions During Runtime*, Omar Jaradat, Sasikumar Punnekkat, In Proceedings of the 23rd International Conference on Reliable Software Technologies (Ada-Europe), June 2018.

Paper E *A Safety-Centric Change Management Framework by Tailoring Agile and V-Model Processes*, Abdallah Salameh and Omar Jaradat, In Proceedings of the 36th International System Safety Conference (ISSC), Aug 2018.

Related Papers Not Included in the PhD Thesis

1. *Automated Verification of AADL-Specifications Using UPPAAL*, Andreas Johnsen, Kristina Lundqvist, Paul Pettersson, Omar Jaradat, In Proceedings of the 14th IEEE International Symposium on High Assurance Systems Engineering (HASE 2012).

2. *Towards a Safety-oriented Process Line for Enabling Reuse in Safety Critical Systems Development and Certification*, Barbara Gallina, Irfan Sljivo, Omar Jaradat, In Proceedings of the 35th Annual IEEE Software Engineering Workshop (FedCSIS Conference) (SEW-36 2012).

3. *The Role of Architectural Model Checking in Conducting Preliminary Safety Assessment*, Omar Jaradat, Patrick Graydon, Iain Bate, In Proceedings of the 31st International System Safety Conference (ISSC 2013).

4. *An Approach to Maintaining Safety Case Evidence After A System Change*, Omar Jaradat, Patrick Graydon, Iain Bate, In Proceedings of the 10th European Dependable Computing Conference (EDCC 2014)).

5. *Deriving Safety Contracts to Support Architecture Design of Safety Critical Systems*, Irfan Sljivo, Omar Jaradat, Iain Bate, Patrick Graydon, In Proceedings of the 16th IEEE International Symposium on High Assurance Systems Engineering (HASE 2015).

6. *Facilitating the Maintenance of Safety Cases*, Omar Jaradat, Iain Bate, Sasikumar Punnekkat, In Proceedings of the 3rd International Conference on Reliability, Safety and Hazard - Advances in Reliability, Maintenance and Safety (ICRES-ARMS 2015).

7. *Systematic Maintenance of Safety Cases to Reduce Risk*, Omar Jaradat, Iain Bate, In Proceedings of the 4th International Workshop on Assurance Cases for Software-intensive Systems (ASSURE 2016).

8. *Challenges of Safety Assurance for Industry 4.0*, Omar Jaradat, Irfan Sljivo, Ibrahim Habli, Richard Hawkins, In Proceedings of the 13rd European Dependable Computing Conference (EDCC 2017).

9. *Contract-Based Assurance for Wireless Cooperative Functions of Vehicular Systems*, Svetlana Girs, Irfan Sljivo, Omar Jaradat, In Proceedings of the 43rd Annual Conference of the IEEE Industrial Electronics Society (IECON 2017).

10. *Service Level Agreements for Safe and Configurable Production Environments*, Mohammad Ashjaei, Kester Clegg , Lorenzo Corneo , Richard Hawkins , Omar Jaradat, Vincenzo Gulisano , Yiannis Nikolakopoulos, In Proceedings of the 23rd International Conference on Emerging Technologies and Factory Automation (ETFA 2018).

11. *Using Safety Contracts to Guide the Maintenance of Systems and Safety Cases: An Example*, Omar Jaradat, Iain Bate, MRTC technical report, Mälardalen University, April 2017.

Contents

I

Thesis

Chapter 1

Introduction

Safety critical systems are those systems whose failure could result in loss of life, significant property damage or damage to the environment [1]. Assuring safety for such systems should provide justified confidence that all potential risks due to system failures are either eliminated or acceptably mitigated. Hence, all failures which might expose the manufacturing processes to hazards shall be analysed and controlled as part of pre-deployment safety assurance and monitored and controlled as part of operational phase.

The size and complexity of safety critical systems are considerable. Without adequate evidence to support the safe performance and clear demonstration of that performance, it is difficult for safety assessors or system developers themselves to build sufficient confidence in their safety critical systems. Therefore, developers of safety critical systems in several domains are required to demonstrate the safe performance of their systems through a reasoned argument that justifies why the system in question is acceptably safe (or will be so) [2], in the light of the available evidence. This argument is communicated via an artefact that is known as a *safety case*. Typically, a safety case comprises both safety evidence (e.g. safety analyses, software and hardware inspection reports, or functional test results) and a safety argument (i.e., reasoning) explaining that evidence. The safety argument shows how developers use available evidence to support safety claims and how those claims, in turn, support broader claims about system behaviour, hazards addressed, and, ultimately, acceptable safety [3].

An organisation building a safety case should be accountable for the ownership of the risks to be controlled by adopting an appropriate safety management system, performing a hazard assessment, selecting appropriate controls, and implementing them [4]. In order to help building a sufficient and credible (i.e., on a scientific basis) confidence in the safe performance of a system, its safety case shall always communicate the safe performance of the system, and shall always contain only acceptable items of evidence that the system meets its safety requirements.

Moreover, safety critical systems can be evolutionary as they are subject to changes due to perfective, corrective or adaptive maintenance or through technology obsolescence [5]. An item of evidence is valid only in the operational and environmental context in which it is obtained or to which it applies. Since changes to a system during or after its development may add, remove or modify its operational or environmental assumptions, changes may undermine the collected items of evidence and thus defeat articulated safety claims. Evidence might no longer support the developers' claims because it reflects old development artefacts or old assumptions about operation or the operating environment. After a change, original safety claims might be nonsense, no longer reflect operational intent, or be contradicted by new data [3]. Eventually, the real system will have diverged so far from that represented by the safety case argument and the latter is no longer valid or useful [6]. Hence, it is almost inevitable that the safety case will require updating throughout the operational lifetime of the system. In addition, any change that might compromise system safety involves repeating the certification process (i.e., re-certification) and repeating the certification process necessitates an updated and valid safety case that considers the changes. For example, the UK Ministry of Defence Ship Safety Management System Handbook JSP 430 requires that *"the safety case will be updated ... to reflect changes in the design and/or operational usage which impact on safety, or to address newly identified hazards. The safety case will be a management tool for controlling safety through life including design and operation role changes"* [7, 8]. Similarly, the UK Health and Safety Executive (HSE) — Railway safety case regulations 1994 — states in regulation 6(1) that *"a safety case to be revised whenever, appropriate that is whenever any of its contents would otherwise become inaccurate or incomplete"* [9, 8].

Furthermore, change to a safety case may necessitate many other consequential changes — creating a ripple effect [5]. Any improper maintenance in a safety argument might cause unforeseen violations of the acceptable safety limits, which will negatively impact the system safety performance conveyed by the safety case. A step to assess the impact of this change on the safety argument is crucial and highly needed prior to updating a safety argument after a system change. Despite the significance of how to deal with changes to the system or its operational environment, as well as the clear recommendations to adequately maintain and review safety cases by safety standards, existing standards offer little advice on how such operations can be carried out [5]. More clearly, some safety standards provide generic guidance of the change management and impact analysis on the system level, and briefly describe the expected outputs.

Using contracts has been around for a few decades in the system development domain [10]. Generally speaking, contracts are intended to describe functional and behavioural properties for each design component in the form of *assumptions* and *guarantees*. Contracts have been discussed as a means for helping to manage system changes in the system domain or in its corresponding safety case [11, 12, 13]. The cost of maintaining, reusing and changing software components is lessened when using contracts as developers may rework components with knowledge of the constraints placed upon them [14]. However, deriving safety contracts and their contents have received little support yet [15]. Also, most of the works that discuss contracts for system maintenance are limited to component-to-component contracts with not much focus on the dependencies between a component and its operational environment, related safety requirements and relevant safety argument.

In this thesis, we consider a safety case as a living document that should always be maintained to correctly portray the safety of a system [5], and evolves as the system evolves. The overall goal of this thesis is *to introduce new change management techniques in order to facilitate the maintenance of safety cases due to system or environment changes.* Our work focuses on: (1) How and where to derive safety contracts and their contents as a supportive means to our suggested change management techniques, (2) using the derived contracts to support the decision as to whether or not apply changes, (3) using the derived contracts to guide developers to the parts in the safety case that might be affected after applying a change, and (4) using the derived contracts to enable through-life safety assurance.

The rationale of our suggested techniques is to determine, for each component, the allowed range for a certain parameter within which a component may change before it compromises a certain system property (e.g., safety, reliability, etc.). We use sensitivity and importance analyses to define safety thresholds (i.e., the maximum allowed change). Subsequently, we derive safety contract where the guarantees represent these thresholds. We also include in each contract whatever assumption that can violate a guaranteed threshold. We use the derived safety contracts to facilitate the accommodation of system changes in safety cases to ultimately support the maintainability of safety cases.

1.1 Thesis Outline

The thesis report is organised into two main parts. **Part I** includes five chapters. Chapter 1 provides an introduction to the thesis where an overview of the research problem, motivation and the thesis contributions are presented. In Chapter 2, we present background information and overview of the most prominent terms that appear frequently in this thesis. In Chapter 3, we describe the research scope, state-of-the-art, research problem, and derive the research goal and research questions. We also describe the overall methodology that is adopted to perform the research. In Chapter 4, we present the contributions of the research and reflect how they address the research questions. In Chapter 5, we draw the conclusion and describe possible directions for future work. **Part II** contains the research papers included in the thesis.

Paper A (Chapter 6): Using Sensitivity Analysis to Facilitate The Maintenance of Safety Cases, Omar Jaradat, Iain Bate, Sasikumar Punnekkat.
Abstract: *"A safety case contains safety arguments together with supporting evidence that together should demonstrate that a system is acceptably safe. System changes pose a challenge to the soundness and cogency of the safety case argument. Maintaining safety arguments is a painstaking process because it requires performing a change impact analysis through interdependent elements. Changes are often performed years after the deployment of a system making it harder for safety case developers to know which parts of the argument are affected. Contracts have been proposed as a means for helping to manage changes. There has been significant work that discusses how to represent and to use them but there has been little on how to derive them. In this paper, we propose a sensitivity analysis approach to derive contracts from Fault Tree Analyses and use them to trace changes in the safety argument, thus facilitating easier maintenance of the safety argument."* [15]

Status: Published in Proceedings of the 20th International Conference on Reliable Software Technologies, Ada-Europe 2015.

My contribution: I was the main contributor of the work under supervision of the co-authors. My contributions include combining the results of sensitivity analysis together with the concept of contracts to identify the sensitive parts of a system and highlight these parts to help the experts to make an educated decision as to whether or not apply changes.

Paper B (Chapter 7): Deriving Hierarchical Safety Contracts, Omar Jaradat, Iain Bate, Sasikumar Punnekkat.
Abstract: *"Safety cases are costly since they need significant amount of time and efforts to produce. This cost can be dramatically increased (even for already certified systems) due to system changes as they require maintaining the safety case before it can be submitted for certification. Anticipating potential changes is useful since it reveals traceable consequences that will eventually reduce the maintenance efforts. However, considering a complete list of anticipated changes is difficult. What can be easier though is to determine the flexibility of system components to changes. Using sensitivity analysis is useful to measure the flexibility of the different system properties to changes. Furthermore, contracts have been proposed as a means for facilitating the change management process due to their ability to record the dependencies among system's components. In this paper, we extend a technique that uses a sensitivity analysis to derive safety contracts from Fault Tree Analyses and uses these contracts to trace changes in the safety argument. The extension aims to enabling the derivation of hierarchical and correlated safety contracts. We motivate the extension through an illustrative example within which we identify limitations of the technique and discuss potential solutions to these limitations."*

Status: Published in Proceedings of the 21st IEEE Pacific Rim International Symposium on Dependable Computing, PRDC 2015.

Main contribution: I was the main contributor of the work under Bate's supervision. My contribution comprises: (1) identifying possible limitations for the proposed technique in Paper A and (2) suggesting an extension to the technique to resolve the identified limitations.

Paper C (Chapter 8): Using Safety Contracts to Guide the Maintenance of Systems and Safety Cases, Omar Jaradat and Iain Bate.
Abstract: *"Changes to safety critical systems are inevitable and can impact the safety confidence about a system as their effects can refute articulated claims about safety or challenge the supporting evidence on which this confidence relies. In order to maintain the safety confidence under changes, system developers need to re-analyse and re-verify the system to generate new valid items of evidence. Identifying the effects of a particular change is a crucial step in any change management process as it enables system*

developers to estimate the required maintenance effort and reduce the cost by avoiding wider analyses and verification than strictly necessary. This paper presents a sensitivity analysis-based technique which aims at measuring the ability of a system to contain a change (i.e., robustness) without the need to make a major re-design. The proposed technique exploits the safety margins in the budgeted failure probabilities of events in a probabilistic fault-tree analysis to compensate for unaccounted deficits or changes due to maintenance. The technique utilises safety contracts to provide prescriptive data for what is needed to be revisited and verified to maintain system safety when changes happen. We demonstrate the technique on an aircraft wheel braking system."

Status: Published In Proceedings of the 13rd IEEE European Dependable Computing Conference (EDCC), Geneva, Switzerland, December 2017.

Main contribution: I was the main contributor of the work under Bate's supervision. My contributions include introducing a new technique besides SANESAM and SANESAM+, which focuses on containing (i.e., localising) the potential changes in the smallest possible part of a system. More clearly, the technique in this paper makes use of the margins in the failure probability of some FTA events to backup the violated MAFP (Maximum Allowed Failure Probability) of other affected events due to a change.

Paper D (Chapter 9): Using Safety Contracts to Verify Design Assumptions During Runtime, Omar Jaradat and Sasikumar Punnekkat.
Abstract: *"A safety case comprises evidence and argument justifying how each item of evidence supports claims about safety assurance. Supporting claims by untrustworthy or inappropriate evidence can lead to a false assurance regarding the safe performance of a system. Having sufficient confidence in safety evidence is essential to avoid any unanticipated surprise during operational phase. Sometimes, however, it is impractical to wait for high quality evidence from a system's operational life, where developers have no choice but to rely on evidence with some uncertainty (e.g., using a generic failure rate measure from a handbook to support a claim about the reliability of a component). Runtime monitoring can reveal insightful information, which can help to verify whether the preliminary confidence was over- or underestimated. In this paper, we propose a technique which uses runtime monitoring in a novel way to detect the divergence between the failure rates (which were used in the safety analyses) and the observed failure rates in the operational life. The technique utilises safety contracts to provide prescriptive*

data for what should be monitored, and what parts of the safety argument should be revisited to maintain system safety when a divergence is detected. We demonstrate the technique in the context of Automated Guided Vehicles (AGVs).”

Status: Published In Proceedings of the 23rd International Conference on Reliable Software Technologies (Ada-Europe), Lisbon, Portugal, June 2018.

Main contribution: I was the main contributor of the work under Punnekkat’s supervision. My main contributions includes: (1) A novel technique to continuously reassess the failure rates and use the results to suggest system changes or maintenance, (2) a new way to derive safety contracts to facilitate the traceability between the system design, safety analysis and the safety case, (3) an example of how to argue more compelling over the failure rate in the light of the derived evidence from the operational phase, and (4) an example of how to carry out a through-life safety assurance.

Paper E (Chapter 10): A Safety-Centric Change Management Framework by Tailoring Agile and V-Model Processes, Abdallah Salameh and Omar Jaradat.
Abstract: *“Safety critical systems are evolutionary and subject to preventive, perfective, corrective or adaptive changes during their lifecycle. Changes to any part of those systems can undermine the confidence in safety since changes can refute articulated claims about safety or challenge the supporting evidence on which this confidence relies. Changes to the software components are no exception. In order to maintain the confidence in the safety performance, developers must update their system and its safety case. Agile methodologies are known to embrace changes to software where agilists strive to manage changes, not to prevent them. In this paper, we introduce a novel framework in which we tailor a hybrid process of agile software development and the traditional V-model. The tailored process aims to facilitate the accommodation of non-structural changes to the software parts of safety critical systems. We illustrate our framework in the context of ISO 26262 safety standard.”*

Status: Published In Proceedings of the 36th International System Safety Conference (ISSC), Arizona, USA, August 2018.

Main contribution: Myself and Salameh were the main drivers. The main contribution in the paper is the introduction of *XP-Kan-Safe* as a novel maintenance framework to facilitate the accommodation process of software non-structural changes in safety critical systems by utilising the strengths of agile methods and the V-model. My contribution focused on deriving safety contracts from safety analyses and associate them with test cases and safety cases to enable a tri-directional change management process.

Chapter 2

Background

In this chapter, we provide background and overview of the most prominent terms that appear frequently in this thesis.

2.1 Safety Critical Systems

The word 'safety' means: *"The condition of being protected from or unlikely to cause danger, risk, or injury"* [16]. *Safety critical "is a term applied to a condition, event, operation, process or item that is essential to safe system operation or use, e.g., safety critical function, safety critical path, and safety critical component"* [17]. ***Safety critical systems*** *are those systems whose failure might endanger human life, lead to substantial economic loss, or cause extensive environmental damage* [1]. The operation of safety critical systems should be safe and, ideally, never cause severe consequences. However, developing an absolutely safe system is unattainable even if the project has an open budget. This is because severe consequences are typically linked to system faults and we cannot be 100% certain that a system is fault free. However, this shall not discourage the efforts that aim at assuring systems' safety.

The key to assuring safety is to eliminate hazards or to ensure that the consequences of these hazards are minimal. The word ***hazard*** in English is defined as: *"a potential source of danger"* [16]. In the context of safety critical systems, there are different suggestions to explain what the word *hazard* means. Some definitions suggest that a hazard is simply a system state that could lead to accidents. For example, Knight [18] indicates that the word *hazard* is an abbreviation of ***hazard state*** and it means: *"a system state that could lead to an unplanned loss of life, loss of property, release of energy, or release of dangerous materials"*. Some other definitions suggest to consider the potential environmental conditions in the definition to clarify the relationship between hazards and accidents. For example, Leveson [19] define *hazard* as: *"a state or condition of a system that, combined with certain environmental conditions, could lead to accidents"*. Anyway, all definitions agree that a hazard is a system state in which an accident might occur. An ***accident*** can be defined as: *"An unplanned event or sequence of events which results in human death or injury, damage to property, or to the environment"* [20]. The measure of the probability that a system will cause an accident is referred to as ***risk***. The risk is assessed by considering the probability that someone/something will be harmed if exposed to a hazard (also known as ***exposure***) and the severity of that hazard.

Hazards are caused by malfunctioning behaviours (i.e., failures) [21]. A ***Failure*** is defined as: *"an event that occurs when the delivered service deviates from correct service"* [22]. Failures are caused by errors. An ***error*** is defined as: *a part of the total state of the system that may lead to its subsequent service failure* [22]. Finally, errors are caused by faults. A ***fault*** is defined as: *an adjudged or hypothesized cause of an error* [22].

Figure 2.1 illustrates some of the system safety concepts and how they relate to each other. The figure uses a scenario from an adaptive cruise control system[1] to exemplify these concepts.

Any process or activity that aims at assuring or improving systems' safety should identify and eliminate potential hazards of those systems. If the elimination is not possible then they should be mitigated to an acceptable level. *Preliminary Hazard Analysis (PHA)* can be done with only a description of the system's concept and functions. That is, PHA is typically used in the early stages of the system's lifecycle where not enough design details are available. PHA consists of four main tasks as follows [23]:

- Identify system hazards
- Translate system hazards into high-level system safety design constraints
- Assess hazards if required to do so
- Establish the hazard log

[1]Adaptive Cruise Control (ACC) system is an optional system for road vehicles that automatically adjusts the vehicle speed to maintain a safe distance from vehicles ahead.

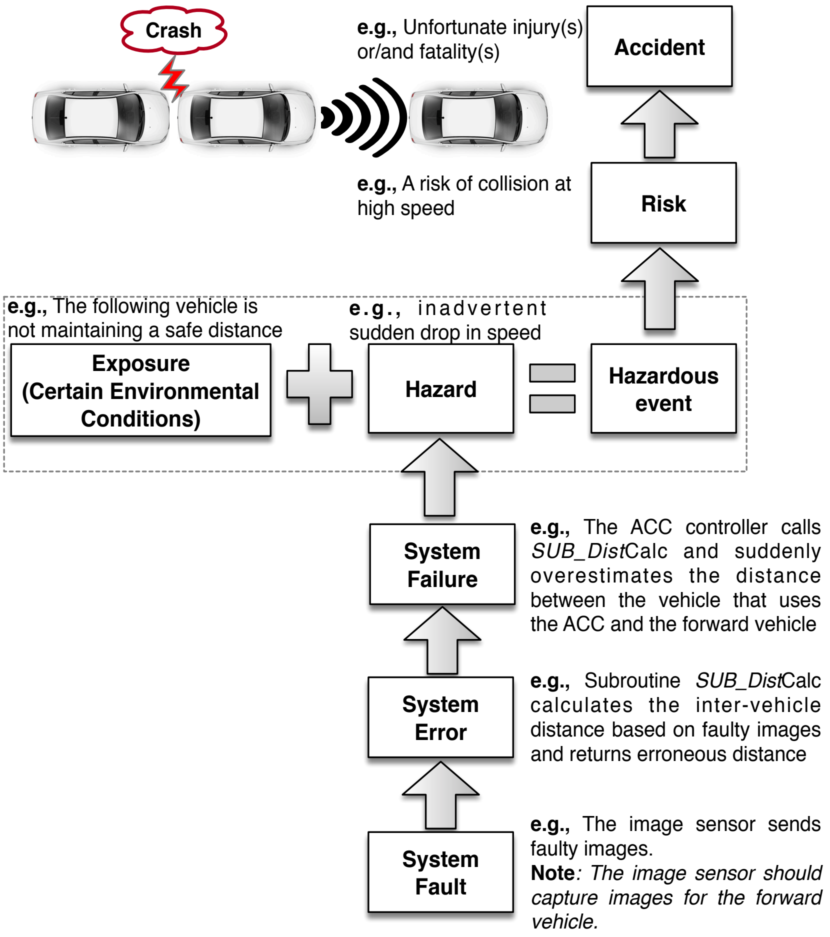

Crash
e.g., Unfortunate injury(s) or/and fatality(s)
Accident
e.g., A risk of collision at high speed
Risk
e.g., The following vehicle is not maintaining a safe distance
e.g., inadvertent sudden drop in speed
Exposure (Certain Environmental Conditions)
Hazard
Hazardous event
System Failure
e.g., The ACC controller calls SUB_DistCalc and suddenly overestimates the distance between the vehicle that uses the ACC and the forward vehicle
System Error
e.g., Subroutine SUB_DistCalc calculates the inter-vehicle distance based on faulty images and returns erroneous distance
System Fault
e.g., The image sensor sends faulty images.
Note: The image sensor should capture images for the forward vehicle.

Figure 2.1: Overview of basic system safety concepts

Safety functions (also know as *safety barriers*) shall be identified, implemented and verified to achieve or maintain the safe state of a system (with respect to the identified hazards). These functions can be safeguards, countermeasures, or protection layers (e.g., car collision avoidance system, fire and gas detection system, pressure relief system, emergency shutdown system). The reliability of such functions are crucial to achieve safety. Reliability here means *"the ability of the item to perform a required function, under given environmental and operational conditions and for a stated period of time"* [24]. However, safety should not be confused with reliability. A reliable system can be unsafe and vice versa. The software of a reliable system may still behave in such a way that the resultant system behavior leads to an accident.

The Lufthansa flight 2904 accident can be an example of how a reliable system may be unsafe. The plane was landing at Warsaw airport in Poland when the computer-controlled braking system did not work. While landing, the braking system did not recognise that the plane touched the ground and assumed that the aircraft was still airborne. A safety feature on the aircraft had stopped the deployment of the reverse thrust system, which slows down the aircraft. The plane ran off the end of the runway, hit an earth bank, and caught fire [20]. The investigations revealed that the braking system software was reliable and had operated according to its specification, but this did not lead to a safe system [20].

2.2 Safety Analysis

In order to assure that their systems perform as needed and provide acceptable levels of safety, safety engineers need to find the causal dependencies between system level hazards and failures of individual components. The automotive safety standard ISO 26262, for instance, states that the objectives of safety analyses are to 1) examine the consequences of faults and failures on the functions, behaviour and design of items and elements, and 2) provide information on conditions and causes that could lead to the violation of a safety goal or safety requirement [21]. Hence, all potential failures that can contribute to identified hazards shall be identified, analysed and managed. To this end, a wide variety of safety analysis techniques with different methodologies, are available. Moreover, there are different criteria used to distinguish these techniques (e.g., qualitative vs. quantitative, formal vs. informal, deductive vs. inductive). The work in this thesis, employs, to a large extent, the fault tree analysis technique. The work also employs the failure mode and effects analysis technique but to a smaller extent.

2.2.1 Failure Mode and Effects Analysis (FMEA)

The first formal FMEAs were conducted in the aerospace industry in the mid-1960s and were specifically focused on safety issues [25]. The goal with safety

FMEAs was, and remains today, to prevent safety accidents and incidents from occurring [25]. FMEA is a bottom-up analytical technique, and it usually depends on architectural system design or a functional block diagram to identify failure modes for each function in a system. The effects of the identified failure modes are described and, in some cases, assigned a probability based on the failure rate and failure mode ratio of the function or component. It is worth mentioning that performing FMEA is recommended by many safety standards from different domains. However, FMEA is not the best technique to investigate the effects of multiple failures. Hence, safety analysts usually combine it with other techniques (e.g., FTA) to perform more complete safety analysis.

2.2.2 Fault Tree Analysis (FTA)

In 1962, Bell Telephone Laboratories introduced the fault tree technique as a means to evaluate safety in the launching system of the intercontinental *Minuteman* missile [26]. The Boeing Company improved the technique and introduced computer programs for both qualitative and quantitative fault tree analysis. Today FTA is the most commonly used technique for safety and reliability studies.

FTA is a top-down failure analysis method which focuses on one particular undesired event and provides a method for determining causes of this event [27]. In other words, FTA is used to specify the occurrence of critical states (from a safety or reliability standpoint). These states might be associated with component hardware failures, human errors, software errors, or any other pertinent events. FTA helps safety engineers to identify plausible causes (i.e., faults) of undesired events [28].

A fault tree illustrates the logical interrelationships of the system's components (***Basic Events***) that lead to the undesired event or the system's state (***Top Event***) [28, 26]. These logical interrelationships are called ***Logical Gates***. Figure 2.2 shows the most commonly used FTA symbols.

Similar to FMEA, Performing FTA is recommended by many safety standards from different domains. FTA is also used as a method to achieve Probabilistic Safety Analysis (PSA) 2.2.3. More specifically, it is used to quantify system failure probability. Quantitative FT evaluation techniques produce three types of results: (1) numerical probabilities, (2) quantitative importance, and (3) sensitivity evaluations [27]. In this thesis, we exploit the results obtained by sensitivity evaluations to measure how sensitive a system design is to a particular aspect of individual event. Section 2.2.4 provides more details about sensitivity analysis.

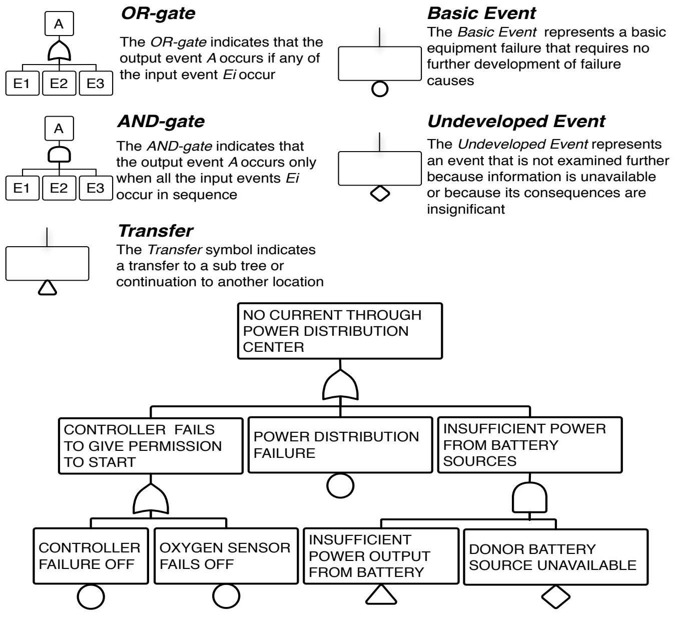

A
OR-gate
The OR-gate indicates that the output event A occurs if any of the input event Ei occur
E1
E2
E3
Basic Event
The Basic Event represents a basic equipment failure that requires no further development of failure causes
AND-gate
The AND-gate indicates that the output event A occurs only when all the input events Ei occur in sequence
Undeveloped Event
The Undeveloped Event represents an event that is not examined further because information is unavailable or because its consequences are insignificant
Transfer
The Transfer symbol indicates a transfer to a sub tree or continuation to another location
NO CURRENT THROUGH POWER DISTRIBUTION CENTER
CONTROLLER FAILS TO GIVE PERMISSION TO START
POWER DISTRIBUTION FAILURE
INSUFFICIENT POWER FROM BATTERY SOURCES
CONTROLLER FAILURE OFF
OXYGEN SENSOR FAILS OFF
INSUFFICIENT POWER OUTPUT FROM BATTERY
DONOR BATTERY SOURCE UNAVAILABLE

Figure 2.2: The principal FTA symbols and an instantiation [26, 29]

2.2.3 Probabilistic Safety Assessments (PSA)

PSA is a qualitative safety analysis to evaluate the probability that an accident might occur. More clearly, PSA is a technique used to numerically quantify risk measures by determining undesired scenarios as well as their likelihoods and consequences. Safety engineers typically starts PSAs by identifying undesired risk or scenario as a top event and investigate the causes that may lead to it. These causes include system failures that need to be identified and quantified using some models like FTA. Failure rates are typically used to quantify failures and they can be assigned to the basic events of the FTAs.

Failure rate is defined as: "*The total number of failures within an item population, divided by the total number of life units expended by that population, during a particular measurement period under stated conditions*" [30].

$$\lambda(t) = \frac{R(t) - R(t + \Delta t)}{\Delta t R(t)} \tag{2.1}$$

where $R(t)$ is the probability of a device not failing prior to some time t, t = t_1 and t_2 = t + Δt. FR is also the inverse of the Mean Time Between Failure (MTBF = $1/\lambda$) , which is generally measured as follows:

$$MTBF = \frac{C_n * T}{F_n} \tag{2.2}$$

where C_n = number of components, T = time period, F_n = number of failures.

The assigned failure rates to the basic events propagate up through the FTA logic to reach the failure rate of the top event.

The result of a PSA is important because it determines the minimum required levels of fault tolerance (e.g., redundancy). However, since the quality of PSA's results is dependent on the quality of the λ estimates, safety analysts should have clear confidence in the used λs in PSAs. For example, the IEC 61508 standard [31] requires failure rates with a 90% confidence level (λ 90%) in order to minimise the requirements for hardware fault tolerance. The standard accepts failure rates with only a 70% confidence level (λ 70%) but a fault tolerance mechanism should be considered.

It is not realistic to assume that all components' failure rates used in a PSA come from the operating experience of a specific system in a statistically meaningful way [32]. Typically, λs come from generic data sources (OREDA, SINTEF, SERH), where a generic failure rate (λ_G) of a particular device might vary from a source to another. The variation depends largely on the service, operating environment and maintenance practices [33]. Imprecise λ can contribute to cognitive impairment of safety. Nevertheless, generic λs are indispensable in any PSA and they are used in practice but the degree in which they are used varies from case to case [32]. More clearly, some PSAs are totally based on λ_G estimates while others use generic estimates as preliminary data that should be replaced later with more specialised data (system specific estimates). This implies that the quality of λ estimates should be reconsidered for a specific system in a specific environment.

Moreover, the failures of components in safety critical systems are typically divided into four modes, namely, Safe Detected (SD), Safe Undetected (SU), Dangerous Detected (DD), and Dangerous Undetected (DU) [34]. DD and DU failures can cause a loss of a safety function while we believe that we are protected and this might happen in fraction of diagnostic interval in case of DD failures or during the unknown downtime in case of DU failures [35]. DU failures are typically due to either random or systematic failures. However, we are after dangerous failures DD and DU. For SD and SU, on the other hand, we will be aware that the safety function is unavailable to whatever reason (e.g., testing, repair, planned preventive maintenance) and we can take precautions to avoid hazardous situations [35]. Whenever FTAs are constructed to evaluate hazards, the basic event failure data must describe only failures that contribute to that hazard and thus only dangerous failure rates (λ_D) should be included for the basic events, where $\lambda_D = \lambda_{DD} + \lambda_{DU}$.

However, the reliability measure of any hazard that is being investigated by a FTA is not determined by simply calculating its λ_D. In fact, the Average Probability of Dangerous Failure on Demand (PFD$_{Avg}$) is used to determine the reliability measure of the top event (i.e., hazard). The PFD$_{Avg}$ is basically a number from 0 to 1 which indicates the likelihood of a safety function to fail to operate on demand, where λ_D is just a variable within the PFD$_{Avg}$ equation. Typically, safety standards specify the minimum allowable PFD$_{Avg}$ of a safety function in order to meet its safety requirement based on SIL, ASIL, DAL, etc. More clearly, the failure measure is defined by PFD$_{Avg}$ of a safety function and the result is compared to predefined target PFD$_{Avg}$ for determined SIL of that function in safety standards. There are different formulae used to calculate PFD$_{Avg}$ depending on different factors, such as system's structure (K-out-of-N structures), Common Cause Factor (CCF), operational maintenance, etc.

2.2.4 Sensitivity Analysis

Sensitivity analysis can be defined as: *"The study of how uncertainty in the output of a model (numerical or otherwise) can be apportioned to different sources of uncertainty in the model input"* [36]. The analysis helps to establish reasonably acceptable confidence in the model by studying the uncertainties that are often associated with variables in models. Many variables in system analysis or design models represent quantities that are very difficult, or even impossible to measure to a great deal of accuracy. In practice, system, developers are usually uncertain about variables in the different system models and they estimate those variables. Sensitivity analysis allows system developers to determine what level of accuracy is necessary for a parameter (variable) to make the model sufficiently useful and valid [37].

There are different purposes for using sensitivity analysis. The analysis can be used to provide insight into the robustness of model results when making decisions [38]. Also, the analysis can be used to enhancing communication from modelers to decision makers, for example, by making recommendations more credible, understandable, compelling or persuasive [39]. In safety domains, sensitivity analysis can be used in risk analysis models to determine the most significant exposure or risk factors so to speak, and thus, it can support the prioritisation of the risk mitigation. Sensitivity analysis methods can be classified in different ways such as mathematical, graphical, statistical, etc. In this paper we use the sensitivity analysis to identify the safety argument parts (i.e., sensitive parts) that might require unneeded painstaking work to update with respect to the benefit of a given change. The results of the analysis should be presented in the safety argument so that it is always available up front to get developers' attention.

In this thesis, we apply the sensitivity analysis on FTAs to measure the sensitivity of outcome A (e.g., a safety requirement being true) to a change in a parameter B (e.g., the failure probability in a component). The sensitivity is defined as $\Delta B/B$, where ΔB is the smallest change in B that changes A (e.g., the smallest increase in failure probability that makes safety requirement A false). The failure probability values that are attached to FTA's events are considered input parameters to the sensitivity analysis. A sensitive part of a FTA is defined as one or multiple FTA events whose minimum changes (i.e., the smallest increase in its failure probability due to a system change) have the maximal effect on the FTA, where effect means exceeding failure probabilities (reliability targets) to inadmissible levels. A sensitive event is an event whose failure probability value can significantly influence the validity of the FTA once it increases. In this this, system components whose failure rates correspond to FTA's events whose likelihoods are sensitive are referred to as sensitive components. Hence, changes to a sensitive component cause a great impact to system design [15].

We use the sensitivity analysis as a method to determine the range of failure probability parameter for each event. Hence, our work assumes the existence of a probabilistic FTA where each event in the tree is specified by an actual (i.e., current) failure probability $FP_{Actual|event(x)}$. In addition, our work assumes the existence of the required failure probability for the top event $FP_{Required(Topevent)}$, where the FTA is considered unreliable if:
$FP_{Actual(Topevent)} > FP_{Required(Topevent)}$.

2.3 Safety Assurance and Certification

It is common to adopt a highly prescriptive approach to assure system safety, where safety assurance is demonstrated by showing compliance with the requirements set out as a prescribed process in a safety standard [40]. One purpose of safety assurance is to ensure that a rigorous process has been followed to build a system and an adequate evidence from the audit, inspection and assessment activities, is available. Safety assurance should also continuously assess the adequacy of the identified risk controls and identify additional or modify existing controls to ensure the safety performance and, ultimately, maintain the acceptable safety levels.

2.3.1 Safety Case

In 1965, section 14 of the UK Nuclear Installations Act states:

"Without prejudice to any other requirements of the conditions attached to this license the licensee shall make and implement adequate arrangements for the production and assessment of safety cases consisting of documentation to justify safety during the design, construction, manufacture, commissioning, operation and decommissioning phases of the installation." [41]

Hence, the notion of building safety cases to justify safety is not new and it has been around for almost fifty years. In 1989, the British chemical industry requested from nuclear sites (according to the Control of the Industrial Major Accident Hazards (CIMAH) regulations) to generate a written report that should contain (1) facts about the site, and (2) reasoned arguments about the hazards and risks from the site [42]. This report was also known as a safety case. The objective of the report was to demonstrate to the UK Health and Safety Executive (HSE) that the site is satisfactory by listing the major hazards and risks and shows how they are adequately mitigated.

The development of the safety case as a means of demonstrating acceptable risk began in the nuclear industry but the application of this means was uncommon in other industries. For example, in the Clapham junction accident in Chapter 1, there was no safety case and although the transport system was allegedly mature, regulated and safe, British Rail could not demonstrate why their system was acceptably safe to operate [43]. From 1990s onwards the development of safety cases spread across many other major safety critical industries, such as the railways, offshore oil and gas facilities. [44].

2.3.2 Safety Case Definition

It is worth mentioning that in addition to the term 'safety case', there are different other terms such as '*Assurance Case*' and '*Safety Assurance Case*' that are, sometimes, used interchangeably. An assurance case is defined as: "*A* *reasoned and compelling argument, supported by a body of evidence, that a system, service or organisation will operate as intended for a defined application in a defined environment"* [45]. It is also defined as: *A collection of auditable claims, arguments, and evidence created to support the contention that a defined system/service will satisfy the particular requirements* [46]. As observed from the former or latter definitions, the term 'assurance case' is generic and does not necessarily indicate safety as the property to be assured. Hence, the term 'assurance case' by its own has no particular focus, but if safety is the intended property to be assured, then using terms such as 'safety case' or 'safety assurance case' is more precise where both can be thought of as an instance of an assurance case.

Although the term 'safety case' has become popular today in many safety critical system domains, but its precise meaning is dependent on the purpose that the safety case is intended to satisfy [6]. This raises the question of: *Why do industries need a safety case?* During this work, different purposes that safety cases can satisfy are observed. A safety case is built as a tool:

- To manage residual risks [47]
- To record engineering practices [6]
- In a court of law to address and reduce legal liability [48, 6]
- For marketing

However, before any safety case is attempted, the rationale and purpose of it must be clearly understood. This is vitally important, because if the specific requirements for compiling a safety case are not clear, then the following safety case will also be not clear [6].

There are different definitions of safety case [49, 6]. Most of the available definitions indicate the consensus that a safety case is oriented to demonstrate how a system reduces risk of specific losses to an acceptable level and thus enable a regulator to assess whether the system is acceptably safe to operate. It is worth pointing out that the definition of safety case by the UK Defence Standard 00-56 [50] is the most common. The standard defines the safety case as: "*A structured argument, supported by evidence, intended to justify that a system is acceptably safe for a specific application in a specific operating environment*".

$$\text{Safety Argument} \in \text{Safety Case}$$

The work presented in the two parts of this thesis assumes that the main purpose of a safety case is to justify safety and it refers to the safety case definition by the UK Defence Standard 00-56 wherever the term 'safety case' appears.
A safety case comprises elements as follows:

- *Safety requirements or objectives* that are mainly derived to eliminate or mitigate hazards (also known as goals)
- *Lifecycle artefacts* (also know as work products) which are basically the results of each development phase (e.g. safety analyses, software inspections, or functional tests)
- a *Safety argument* explaining how safety goals (in form of safety claims) are supported by available artefacts (in form of safety evidence)
- *Context* and *Assumptions* about the operating environment and usage

Figure 2.3 shows an overview of the safety case elements and the relationships between them.

2.3.3 Safety Argument

The main purpose of a safety case is to communicate an argument. The argument demonstrates how someone can reasonably conclude that a system is acceptably safe from the evidence available [51]. In English, the word 'argument' is defined as: *"A reason or set of reasons that somebody uses to show that something is true or correct"* [16]. A more technical definition for the word 'argument' is: *A body of information presented with the intention to establish one or more claims through the presentation of related supporting claims, evidence, and contextual information.* [46]. An argument in the safety case definition is called a 'safety argument' or 'safety case argument' and it can be defined as a hierarchically connected series of claims supported by evidence. Safety arguments are intended to demonstrate to the reader that a system is acceptably safe as an overall claim. The claim is defined as: *A proposition being asserted by the author or utterer that is a true or false statement* [46]. The evidence is defined as: *Information or objective artifacts being offered in support of one or more claims* [46].

In order for safety cases to be developed, discussed, challenged, presented and reviewed amongst stakeholders, as well as maintained throughout the product lifecycle, it is necessary for the (1) argument to be clearly structured

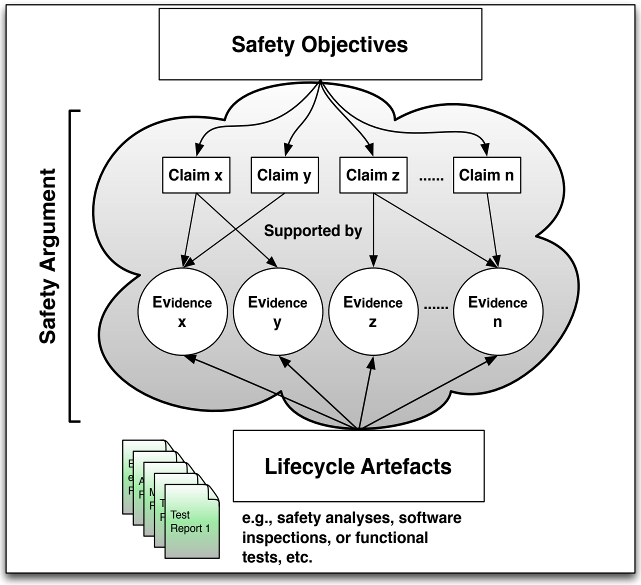


Figure 2.3: Overview of a safety case and its elements

and (2) items of evidence to be clearly asserted to support the argument [45]. There are several ways to represent safety arguments. Safety arguments might be represented by:

- **Prose:** Safety arguments are typically communicated in existing safety cases through free text [51]. Perhaps this is the easiest way to represent safety arguments but not necessarily the most efficient. There are several problems observed while reviewing real safety cases written in prose. We describe some of them. Noticeably, not all engineers who are involved in writing a safety case can write clear and well-structured English [51]. An instance of this problem is, probably, the unconscious use of the ellipsis process in natural languages when authors, unconsciously, leave some non-described crucial details in some statements because they assume that the readers are aware of the context of these statements. For example, the following text describes an evidence item used to support some claims about a bug tracking system of software failures:

"Here we provide evidence of bug tracking for the software. 'XXXXX' is the database that is used to track all issues regarding this system. It has full visibility and is extremely detailed." What does 'all issues' mean? Is it the safety, software or bug issues? How about 'visibility'? Does the writer mean the visibility of the software or the visibility of the bug information? [6]

There are more problems in the description above but the idea is to give an example of the text quality problem.

Another problem observed in safety cases written in prose is the cross-references among texts. Multiple cross-references in texts can be awkward and disrupt the flow of the main argument [51].

- **Tabular notations:** This way to demonstrate safety arguments is not common. The idea, however, is to arrange an argument claim together with its supportive items of evidence in rows and columns.

- **Graphical notations:** This way represents the individual elements of safety arguments (e.g., safety goals, items of evidence and assumptions) using graphical symbols (e.g., squares, circles, parallelograms, etc.). The Goal Structuring Notation (GSN), as well as, the Claims Argument Evidence (CAE) notation are two examples of this way.

Discussing the advantages and disadvantages of the three ways listed above is not an objective of this thesis. We do not claim that a problem in one way can not apply to other ways. However, we use the graphical notation since it can clearly represent the elements of safety arguments and their relationships. Moreover, almost all of the related works to this thesis use GSN thus adopting GSN can make the discussions, comparisons and explanations of our work more clear with respect to other works.

2.3.4 The Goal Structuring Notation (GSN)

GSN is a graphical argument notation which can be used to document explicitly the elements and structure of an argument and the argument's relationship to evidence [45]. GSN's notations are used as a means for communicating (1) safety argument elements, claims, argument logic, assumptions, context, evidence and (2) the relationships between these elements [15].

A goal structure shows how goals are successively ***solved by*** sub-goals until a point is reached where claims can be supported by direct reference to evidence. Using GSN, it is also possible to clarify the argument strategies adopted (i.e., how the premises imply the conclusion), the rationale for the approach (assumptions, justifications) and the context in which goals are stated [15].

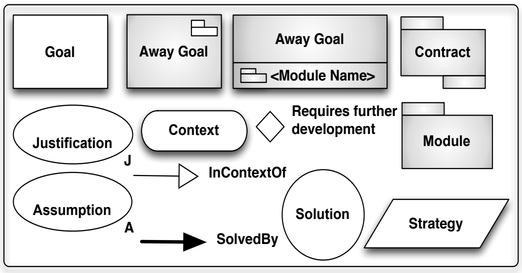


Figure 2.4: Overview of the GSN principal elements

In GSN, rectangles are used to present the argument's claims (***Goals*** in GSN). Parallelograms is used to present the argument's logic (***Strategies*** in GSN). Circles are used to present items of evidence (***Solutions*** in GSN). Ovals with the letter 'J' at the bottom-right are used to present a statement of rationale (***Justifications*** in GSN). Ovals with the letter 'A' at the bottom-right are used to present an intentionally unsubstantiated statement (***Assumptions*** in GSN) [45]. Squashed rectangles are used to present a reference to contextual information or a statement (***Context*** in GSN). Hollow diamonds are applied to the centre of an element (e.g., goal, assumptions, context, etc.) to indicate that a line of argument has not been developed (***Undeveloped <element name>*** in GSN) [45]. ***SupportedBy*** is an evidential relationship which declares the link between a goal and the evidence used to substantiate it [45]. Permitted supported by connections are: goal-to-goal, goal-to-strategy, goal-to-solution, strategy-to-goal. ***InContextOf*** is a link that declares a contextual relationship [45]. Permitted connections are: goal-to-context, goal-to-assumption, goal-to-justification, strategy-to-context, strategy-to-assumption and strategy-to-justification [45].

Figure 2.4 shows the principal GSN elements, and Figure 2.5 shows an example of a safety argument represented by those elements.

GSN has been extended to enable modularity in a safety case (i.e., module-based development of safety cases). Hence, modular GSN enables the partitioning of a safety case into an interconnected set of modules [52].

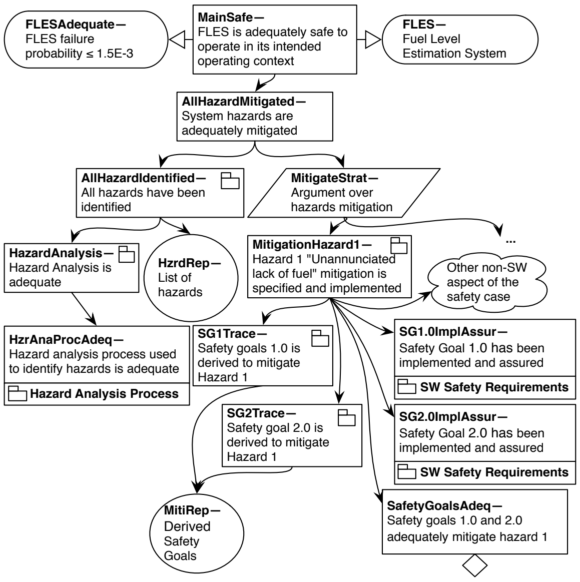

FLESAdequate—
FLES failure probability ≤ 1.5E-3
MainSafe—
FLES is adequately safe to operate in its intended operating context
FLES—
Fuel Level Estimation System
AllHazardMitigated—
System hazards are adequately mitigated
AllHazardIdentified—
All hazards have been identified
MitigateStrat—
Argument over hazards mitigation
...
HazardAnalysis—
Hazard Analysis is adequate
HzrdRep—
List of hazards
MitigationHazard1—
Hazard 1 "Unannunciated lack of fuel" mitigation is specified and implemented
Other non-SW aspect of the safety case
HzrAnaProcAdeq—
Hazard analysis process used to identify hazards is adequate
Hazard Analysis Process
SG1Trace—
Safety goals 1.0 is derived to mitigate Hazard 1
SG1.0ImplAssur—
Safety Goal 1.0 has been implemented and assured
SW Safety Requirements
SG2Trace—
Safety goal 2.0 is derived to mitigate Hazard 1
SG2.0ImplAssur—
Safety Goal 2.0 has been implemented and assured
SW Safety Requirements
MitiRep—
Derived Safety Goals
SafetyGoalsAdeq—
Safety goals 1.0 and 2.0 adequately mitigate hazard 1

Figure 2.5: A safety argument example represented by GSN [53]

Figure 2.4 presents the principal notations of GSN after the extension in gray. An ***Away Goal*** with a bisecting line in the lower half of it the repeats a claim presented in another argument module which is used to support the argument in the local module [45]. The Module Identifier provides a reference to the module that presents the original claim. A ***Module*** reference presents a reference to a module containing an argument. A ***Contract*** module reference presents a reference to a contract module containing definition of the relationships between two modules, defining how a claim in one supports the argument in the other [45].

2.3.5 Confidence in Safety

The safety case (according to our adopted definition) should provide information about what does it mean for a system to be safe and how the safety is measured and assured. Reviewers (or assessors) of the safety case use that information to assess the safety of the whole system by looking into what the system is made of and whether the failures of the different system components are managed in a way that the safety requirements are still respected. Based on the confidence in the adequacy of the articulated claims, in the soundness of the structured argument, and in the trustworthiness, suitability and completeness of the gathered evidence, the reviewers build an overall confidence in the safety performance of a system. That is, the overall confidence in safety is a superset that is composed of and influenced by different subset confidences. The trustworthiness of evidence is one subset that affect the reviewers' overall confidence in safety. This raises two questions:

- What do we mean by evidence?
- What might affect the confidence in trustworthiness of an item of evidence?

Although the common definition of the safety case by many safety standards in different domains requires the existence of a body of *evidence* to support the safety argument, there is little agreement about what evidence is [54]. However, some standards explicitly explain what the evidence should look like or the forms of evidence. For example, the UK defence standard 00-56 describes that the generated item of evidence should consist of one or more forms of four types, namely, **direct, quantitative, process, and qualitative evidence** [50].

To narrow down the focus of this thesis, we point out that we are particularly interested of any item of evidence that supports articulated claims about the failure rates of hardware components. Hence, we define an item of evidence, in our particular context, as any available body of fact or information (source of knowledge) which indicates whether our belief in failure rate measures of hardware components is true or false. Figure 2.6 shows potential sources from which we can support a claimed failure rate.

As for the question: What might affect our confidence in trustworthiness of the items of evidence? In fact, for evidence to be relied on as honest or truthful depends on how complete and accurate is our knowledge about it. Judgment on the trustworthiness of an item of evidence depends on our conception and epistemology (e.g., what we know about the evidence, the context it was obtained for, and the environment it was tested within, etc.). Claiming the completeness of our epistemology is perfect but it is impossible to be proven. However, we shall not make the perfect as the enemy of the good by asking either for everything (including what is beyond our knowledge) or nothing. Alternatively, we should accept practical epistemology by verifying that the evidence exists and adequately supports the claim [54], given our knowledge. Even claims that are supported by expert judgements, they should not be dealt with as axioms but they should be subject to verification whenever possible.

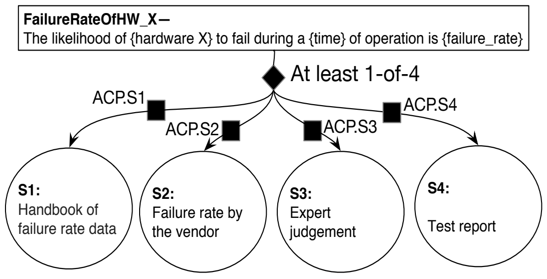

FailureRateOfHW_X—
The likelihood of {hardware X} to fail during a {time} of operation is {failure_rate}
At least 1-of-4
ACP.S1
ACP.S2
ACP.S3
ACP.S4
S1: Handbook of failure rate data
S2: Failure rate by the vendor
S3: Expert judgement
S4: Test report

Figure 2.6: Possible items of evidence to support failure rate measures

Since the realisation of the uncertainty and confidence in safety cases is a key to assess their sufficiency and trustworthiness, it is important to clearly identify, explicitly represent and qualitatively or quantitively assess the confidence and the uncertainty. Some researchers suggested quantitative models to assess the confidence in safety arguments (e.g., [55, 56, 57, 58]). However, adopting the current proposed techniques to quantitatively assess the confidence in safety cases is debatable. This is because quantitative confidence techniques require further validation to prove their efficacy [4]. Other researchers suggested to increase the clarity of the contained confidence in safety cases by developing explicit and separate confidence argument [59]. Assessing the confidence quantitively is beyond the scope of our thesis. However, we are particularly interested of developing explicit and separate confidence arguments and then describe how obtaining information about the system behaviour from its operational life might impact these arguments.

2.3.6 Assured Safety Argument

Hawkins et al., [59] introduced "An assured safety argument" as a structure for arguing safety in which the safety argument is accompanied by a confidence argument that documents the confidence in the structure and bases of the safety argument. More specifically, the suggested structure explicitly separates the safety case argument into two parts [59]:

1. A safety argument: the safety argument is constructed to argue over risk reduction —anything related to the identification and mitigation of hazards associated with a system— and should not cope with confidence.

2. An accompanying confidence argument: the confidence argument is

structured according to the assertions of the safety argument and it is not allowed to contain general 'confidence raising' arguments that cannot be clearly related to the structures of the core safety argument.

Assertions in a safety argument relate to the sufficiency and appropriateness of the inferences declared in the argument, the context and assumptions used and the evidence cited [59]. For example, when an item of evidence is used to support a claim, it is asserted that this evidence is sufficient to support the claim. However, a simple 'SolvedBy' relation between the evidence and the claim will not satisfy a reviewer's concerns to reach a certain level of confidence, such as, why the reviewer should believe that the evidence is appropriate for the claim? Or whether or not it is trustworthy. Instead of decomposing the arguments further to argue over the appropriateness and trustworthiness of the supporting evidence, an Assurance Claim Points (ACP) can be created to indicate an assertion in the safety argument. An ACP is indicated in GSN with a named black rectangle on the relevant link and a confidence argument should be developed for each ACP [59]. Three types of assertions were defined, where an ACP can be created, as follows (Figure 9.3 instantiates an example of each type):

1. Asserted inference: the ACP for an asserted inference is the link between the parent claim and its strategy or sub-claims (e.g., ACP.I1).

2. Asserted context: the ACP for asserted context is the link to the contextual element (e.g., ACP.C2).

3. Asserted solution: the ACP for asserted solutions is the link to the solution element (e.g., ACP.S3).

Other examples are found in Figure 2.6 where *ACP.S1-S4* are examples of asserted solutions.

Incomplete knowledge that can preclude a perfect confidence is referred to as an *Assurance Deficit (AD)* [59]. Highlighting and managing the ADs in safety cases are essential to build an overall all confidence in the safety case. The main motivation of proposing the "assured safety argument" structure is to identify and manage the ADs in safety cases through the ACPs and apart from the safety argument. Having a separate confidence argument eliminates or reduces the difficulties that can emerge when merging confidence and safety arguments in one argument. Including unnecessary material, excluding necessary material and increasing the size and complexity of safety arguments, are examples of these difficulties [59].

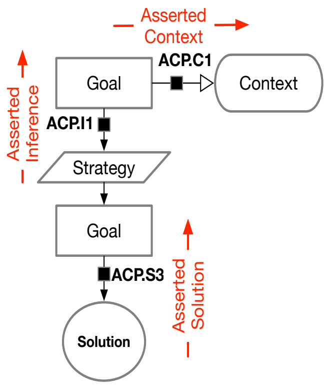


Figure 2.7: Types of ACPs with an example of each usage [59]

2.3.7 Dynamic Safety Case (DSC)

Denney et al., [60] introduced the term "Dynamic Safety Cases (DSCs)" as a novel operationalisation of the concept of through-life safety assurance. The main motivation for introducing DSCs is that the appreciable degree of certainty about the expected runtime behaviour of a system might not be precise or it perhaps over- or underestimate the actual behaviour, which can create deficiencies in the reasoning about the safety performance of that system. Hence, there is a need for a new class of safety assurance techniques that exploit the runtime related data (operational data) to continuously assess and evolve the safety reasoning to, ultimately, provide through-life safety assurance [60].

The suggested lifecycle of DSCs comprises four main activities as follows [60]:

1. *Identify* the sources of uncertainty in a safety case (i.e., the ADs — as described in Subsection 2.3.6).

2. *Monitor* the runtime operation of the related system to collect data about system and environment variables, events, and the ADs in the safety argument(s).

3. *Analyse* the collected operational data from the former activity to examine whether the threshold defined for ADs are met, and to update the confidence in the associated claims

4. *Respond* to operational events that affect safety assurance. Deciding on the appropriate response depends on a combination of factors including the impact of confidence in new data, the available response options already planned, the level of automation provided, and the urgency with which certain stakeholders have to be alerted.

Figure 2.8 shows the four activities of the DSCs' lifecycle.

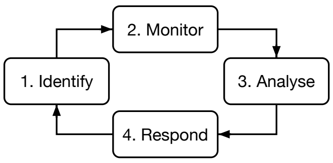


Figure 2.8: An overview of DSCs's lifecycle [60]

2.4 Safety Contracts

The term 'contract' is defined in English as: *"A written or spoken agreement, especially one concerning employment, sales, or tenancy, that is intended to be enforceable by law"* [16]. A contract is intended to (1) establish a binding relationship between one party's offer and the acceptance of that offer by one or more parties, and (2) set out the terms and conditions that constrain this relationship. Using the contracts is familiar in software development. For instance, Design by Contract (DbC) was introduced by Meyer [61, 62] to constrain the interactions that occur between objects. Moreover, contract-based design is an approach where the design process is seen as a successive assembly of components where a component behaviour is represented in terms of assumptions about its environment and guarantees about its behavior [10].

In 1969, Hoare introduced the pre- and postcondition technique to describe the connection (dependency) between the execution results (R) of a program (Q) and the values taken by the variables (P) before that program is initiated [63]. Hoare introduced a new notation to describe this connection, as follows:

$$P\,\{Q\}\,R.$$

This notation can be interpreted as: *"If the assertion P is true before initiation of a program Q, then the assertion R will be true on its completion"* [63].

In the context of contract-based design, a contract is conceived as an extension to the specification of software component interfaces that specifies preconditions and postconditions to describe what properties a component can offer once the surrounding environment satisfies one or more related assumption(s).

Safety and non-safety contracts might describe similar properties; the distinction is whether the guaranteed property is traceable to a hazard [13]. In this thesis, a contract is said to be a *safety contract* if it guarantees a property that is traceable to a hazard. There have been significant works that discuss how to represent and to use contracts [64, 65, 66]. In the safety critical systems domain, researchers have used, for example, assume-guarantee contracts to propose techniques to lower the cost of developing software for safety critical systems. Moreover, contracts have been exploited as a means for helping to manage system changes in a system domain or in its corresponding safety case [11, 12, 13].

The following is an example that depicts the most common used form of contracts:

Guarantee: The WCET of task X is $\leq$ 10 milliseconds
Assumptions:
X is:

1. compiled using compiler [C],
2. executed on microcontroller [M] at 1000 MHz with caches disabled, and
3. not interrupted

In this thesis, we distinguish between safety contracts within the system domain and safety argument contracts in the safety case. The former type of contracts captures the dependencies among the system's components, whereas a safety argument contract captures the dependencies among the safety case modules. More specifically, a safety argument contract describes the connection between a *consumer* goal in one safety case module and a *provider* goal in another module [45].

Chapter 3

Research Overview

This chapter comprises five main sections. The first section describes the research scope. The second section presents literature review of safety maintenance and change management. The third section describes the problem context and provides a motivation of our research goal. The fourth section introduces the research goal and the derived research questions addressed by the thesis. The last section explains the research methodology.

3.1 Research Scope

Safety critical systems are evolutionary and they are always exposed to both predicted and unpredicted changes during the different stages in their lifecycle. System changes are typically introduced to [67]:

1. correct discovered problems (i.e., corrective changes),
2. accommodate changes in the environment in which a system must operate (i.e., adaptive changes),
3. detect and correct latent faults in a system before they are manifested as failures (i.e., perfective changes), or
4. detect and correct latent faults in a system before they become operational faults (i.e., preventive changes).

Changes to a system can negatively affect the gained confidence in the safe performance because they have the potential to compromise the safety evidence which has been already collected. In fact, operational or environmental changes may invalidate a safety case argument for two main reasons [3]:

1. Evidence is valid only in the operational and environmental context in which it is obtained, or to which it applies. During or after a system change, evidence might no longer support the developers' claims because it could reflect old development artefacts or old assumptions about operation or the operating environment.

2. Safety claims, after introducing a change, might be nonsense, no longer reflect operational intent, or be contradicted by new data. Changing safety claims might change the argument structure.

In order to maintain the confidence in the safe performance of a system, safety engineers must maintain the safety case so that it reflects the reality of the current operating status. In practice, this requires (1) identifying, re-analysing, and re-checking the impacted parts of the system, (2) reviewing the safety case and the logic of its argument to evaluate the impacted elements, and (3) correcting the logic of the argument and generating a new valid set of evidence to support any refuted claims.

Maintenance (or maintainability) is not defined in terms of safety cases by safety standards nor the literature. However, an understanding of the term 'safety case maintenance' can be deduced by looking at the definition of 'Maintenance' by relevant standards in computer systems and software engineering domains, and also by considering the objectives of safety cases by the safety standards. For instance, the systems and software engineering standard ISO/IEC 25010 [68] defines the word 'Maintainability' as: *"The degree of effectiveness and efficiency with which a product or system can be modified by the intended maintainers"*. The standard adds a note to the definition:

NOTE 1: Modifications can include corrections, improvements or adaptation of the software to changes in environment, and in requirements and functional specifications. Modifications include those carried out by specialized support staff, and those carried out by business or operational staff, or end users.

On the other hand, the safety standard ISO 26262 [21] states that: *"Throughout the operational life of any system, the corresponding safety case* *might be challenged by additional safety evidence arising from operation, changes and updates to a design, and a shifting regulatory context. In order to maintain an accurate account of the safety of the system, such challenges are assessed for their impact on the original safety argument"*.

Moreover, the regulations of the offshore installations [69] by the Health and Safety Executive (HSE) in the UK state that: *"The safety case is intended to be a living document that reflects the reality of the current operating status on the installation. Changes are likely to occur in the environment, in the activities carried out or in other factors that may affect risks to people. It is therefore important that the safety case is reviewed in the light of any such changes and revised as often as may be necessary to ensure it reflects reality"*. The same regulations also state that *" Safety cases are intended to be living documents, kept up to date and revised as necessary during the operational life of the installation"*.

Also, the UK Ministry of Defence Ship Safety Management System Handbook JSP 430 [7] requires that *"the safety case will be updated ... to reflect changes in the design and/or operational usage which impact on safety, or to address newly identified hazards. The safety case will be a management tool for controlling safety through life including design and operation role changes"*.

From the forgoing, one can deduce that the term 'safety case maintenance' indicates the act of updating a safety case to keep it as a living document that always reflects the reality of the current status of the safe performance of the corresponding system.

Moreover, the maintainability definition by ISO/IEC 25010 [68] can be adapted to define the safety case maintainability so that it can be interpreted as *the degree of effectiveness and efficiency with which a safety case can be modified by the intended maintainers*. The modifications can include corrections in the safety case to respond to system and environment changes, and improvements to address new or higher quality evidence than exist.

Before changing (i.e., modifying) any part of a safety case, maintainers should assess which areas and elements of the safety case are impacted (or will be so) by the change. That is, any approach to maintain a safety case should employ a change management strategy which includes an impact analysis activity to make accurate estimates of the area of the safety case that is affected by a change. The change management strategy should also control the propagation of ripple effects. The accuracy of impact analysis results is dependent on:

1. If the recorded dependencies among the parts of a safety case truly reflect the actual dependencies among them.

2. If the recorded dependencies between the parts of a safety case and the system design truly reflect their actual dependencies.

3. The reliance upon correspondence between safety argument and safety case [5].

There are two techniques used to determine the dependencies in 1 and 2, namely, traceability analysis and dependency analysis.

3.2 Literature Review

This section presents a literature review of safety case maintenance. However, since change management and impact analysis are key principles to achieve maintenance, the review also considers their related contributions from the state-of-the-art and the state-of-the-practice. We divide this section into two subsection. The first subsection discusses the related contributions to safety case maintenance and the second discusses the related contributions to change management.

3.2.1 Safety Case Maintenance

Kelly et al., [5] emphasise that safety engineers have difficulties with safety case maintenance because they lack a systematic and methodical approach to examine the impact of change on safety argument. The authors, therefore, define and describe a tool-supported process to facilitate a systematic impact assessment of safety cases. The process is couched in terms of a safety argument recorded as a goal structure and it comprises two phases, namely, the *Damage* phase and the *Recovery* phase. In the damage phase, safety engineers should assess the impact of change on the safety argument of the safety case. This phase contains three steps as follows:

1. Recognise challenges to safety case

2. Expressing challenge in goal structure terms

3. Using the goal structure to identify impact of challenge

In the recovery phase, the engineers should identify a recovery action and check if the action has further impact in the argument (i.e., propagation). This phase contains two steps as follows:

4. Deciding upon action to recover damaged argument

5. Recover identified damaged argument

The two phases are correlated since a decided action may cause further damages in the safety case.

According to the authors, one limitation of the process is that its ability to express accurately and fully the impact of changes on the safety case depends on the degree to which the goal structured safety argument corresponds to the documented safety case. Hence, preforming an impact analysis in case of a poor match between what is contained in the safety case and what is presented by the safety argument will most likely mislead the engineers' attention. The second limitation of the process is that it considers the dependencies among the elements of the safety argument only and neglects any external dependencies.

Kelly in [70] suggests identifying preventative measures that can be taken when constructing the safety case to limit or reduce the propagation of changes through a safety case expressed in goal-structure terms. For instance, developers can use broad goals (goals that are expressed in terms of a safety margin) so that these goals might act as barriers to the propagation of change as they permit a range of possible solutions. A safety case therefore, interspersed with such goals at strategic positions in the goal structure could effectively contain "firewalls" to change. Some of these initial ideas concerning change and maintenance of safety cases have been presented in [71]. However, no work was provided to show how these thoughts can facilitate the maintenance of safety cases.

A consortium of researchers and industrial practitioners called the Industrial Avionics Working Group (IAWG) has proposed modular safety cases as a means of containing the cost of change. IAWG's Modular Software Safety Case (MSSC) process [72] requires the development of a modular safety case, where the safety case is composed of a set of independent modules within a safety case architecture to form a coherent safety case for the system. The MSSC process facilitates handling system changes as a series of relatively small increments rather than occasional major updates as shown in Figure 3.1. The process proposes to divide the system into a set of blocks [11]. Each block may correspond to one or more software components but it is associated to exactly one dedicated safety case module. Engineers attempt to scope

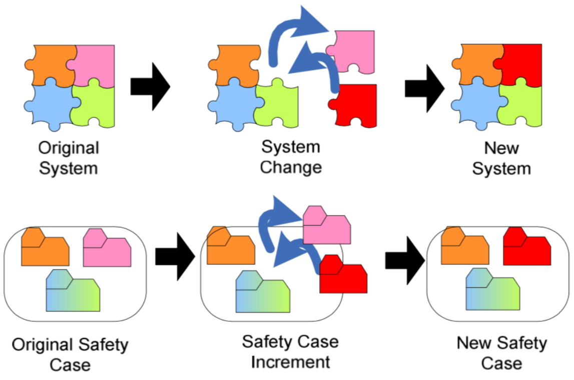


Figure 3.1: An incremental safety case updates [72]

blocks so that anticipated changes will be contained within argument module boundaries. The process establishes component traceability between system blocks and their safety argument modules using Dependency-Guarantee Relationships (DGRs) and Dependency-Guarantee Contracts (DGCs). DGRs and DGCs serve as a means to record the dependencies among the system blocks to allow efficient impact analysis.

Part of the MSSC process is to understand the impact of change so that this can be used as part of producing an appropriate argument. The MSSC process, however, does not give details of how to do this. Moreover, the MSSC process is dependent on a list of predicted change scenarios and it is almost useless for arbitrary changes. The lack of systematic ways to enable better changes prediction might lead to a big limitation to the process.

Nicholson et al., [73] propose a maintenance approach which is limited to Integrated Modular Systems (IMS). The underling logic of the approach is to contain the impact of changes, initiated as necessary during the operational life of the system, within a system' partition without requiring partition boundaries to be moved. More clearly, the presented approach assumes that IMS are built to be extended and modified after they are released. This means that there are safety margins (i.e., slack in the system resources) not used until a critical limit is reached. Hence, as long as the impact of a change to a partition in the system will not exceed the safety margin in the resources of this partition, the change can be accommodated without further changes to the system. However, evidence should be provided to prove that the safety margin is sufficient to cover the needs of the change. Also, the impact on the safety basis of the system can then be assessed via an incremental certification process. The authors identify eight potential categories of changes and explain that IMS can enable an incremental maintenance so that some, or all, of identified categories of change can be implemented without the need to re-certify the entire system.

Björnander et al., [74] use introduce the GSN and AADL (Architecture Analysis and Design Language (AADL) Graph Evaluation (GAGE) method which parses and maps safety argumentation structure against system architecture. The underlying logic behind the method is that bridging the gap between the safety case and the system architecture allows system developers to evaluate the safety assurance case against the properties of the components of the system. Björnander motivates the method as a means for tracing potential changes in the system onto the safety argumentation. The method comprises three main steps: 1) organise the system architecture as a directed acyclic graph, 2) organise the safety case as a directed acyclic graph, where each formal argument is connected to a Boolean expression, and finally 3) for each argument, identify the goals supporting the argument in the structure. One limitation of GAGE is that it does not consider the traceability analysis between the safety case and the model in order to detect the parts of the safety cases affected by a change in a component property, and to detect the components affected by a change in a safety case.

Kokaly et al., [75] propose a technique which uses a model-based approach to perform impact assessment on GSN-based compliance argument with ISO 26262 safety standard due to system changes. Kokaly defines and describes an impact assessment algorithm called GSN-IA (GSN Impact Assessment) and a supporting model transformations. The objective of GSN-IA is to highlight "mark" the safety case elements which are affected due to a change. GSN-IA depends on a model slicer used to determine how change impact propagates within a system model so that the slicer itself is a main input to GSN-IA. Kokaly considers two ways that a change to the system can impact the elements of a safety case: 1) *Revise* where the content of an element of a safety case may have to be revised because it referred to a system element that has changed and the semantics of the content may have changed. 2) *Recheck* where the state of the element must be rechecked because it may have changed. If an element is not impacted by a system change, it is marked as *reuse*. Hence, the output of GSN-IA is design model in which the elements are marked for revise, recheck or reuse.The technique addresses the effect of adding components in the system on the existing parts of the safety case. However, it does not consider how adding a component can require additions to the safety case. Also, it is unclear how the safety analyses are associated with the impact assessment process and how they can get impacted by a change.

3.2.2 Change Management and Impact Analysis

Generally, the change management process in systems engineering is the process of requesting, determining attainability, planning, implementing, and evaluating of changes to a system. Its main goals are to support the processing and traceability of changes to an interconnected set of factors [76]. In computer systems, a change to specific part of a system often impact other parts of the same system. Change impact analysis should be performed prior to the change implementation in oder to identify what might get impacted by the change and also gives an estimate of the adaptation costs.

Many researchers proposed change management approaches and techniques. For example, Dick [77] suggests a five steps change management process to conduct changes in a system. Dick believes that the investment in traceability is the key driver of the change management, therefore, his suggested change management process is dependent on building traceability matrices. According to Dick, traceability is the activity of documenting the relationships between layers of information — for instance between system requirements and software design. The required steps to perform Dick's change management process is as follows:

1. Determine which artefacts a change affects most directly
2. Calculate the potential impact tree by processing the traceability relationships
3. Prune and elaborate the impact tree using engineering judgements, where traceability rationale helps to determine the precise nature of change propagation
4. Define change by traversing the impact tree, working out the precise details of the changes at each point
5. Apply the change.

Other researchers propose investing in traceability to eventually enable similar change impact analyses like the one proposed by Dick (e.g., [78, 79, 80, 81]).

Almost all the existing impact analysis techniques in the literature focus on either identifying or enhancing the identification of the affected parts of the systems due to changes. The main downside of most of these techniques is that they neglect the amount of potential suspect parts due to changes. In fact, the inability of highlighting the impacted parts or highlighting a big set of impacted parts are two different drawbacks that are equally important. The unawareness of what might get impacted will push the developers to carry out a conservative and wider verification than strictly necessary, whereas detecting a big set of impacted parts will decrease the accuracy of the impact analysis and also increases the re-verification efforts [82].

Bohner [83], however, suggests to calculate the accuracy of detecting the affected parts of a system under a change within his proposed impact analysis approach. Bohner suggests different types of sets while performing the change impact analysis, as follows:

1. The Starting Impact Set (SIS): is an initial set of system elements that are thought to be impacted by a change. The SIS is usually thought of once the change specifications are available.

2. The Candidate Impact Set (CIS): is a set of system elements that are estimated to be affected. The CIS is produced while conducting the impact analysis.

3. The Actual Impact Set (AIS): is a set of system elements actually modified which is obviously produced after modifying system elements.

4. The Discovered Impact Set (DIS): is a set represents an under-estimate of impacts. The idea of producing this set is to identify other impacted system elements that were never thought of before implementing a change.

5. The False-Positive Impact Set (FPIS): is a set represents the over-estimate of impacts in the analysis.

The objective of the impact analysis, according to Bohner, is to conclude the CIS produced from tracing potential impacts as close to the AIS as possible by:

$$AIS = CIS + DIS - FBIS \tag{3.1}$$

Moreover, the accuracy is determined by:

$$Accuracy = \frac{DIS + FPIS}{CIS} \tag{3.2}$$

Bohner suggests to including more semantic information in the impact analysis process to reduce the number of false positives. Semantic information is actually the type of relationship between Software Life-Cycle Objects (SLOs), such as decomposition relationships among requirements. However, he does not provide any clues as how to use this information [82].

Oertel [82] proposes an impact analysis technique which provides a linear relation between the re-verification efforts and the size of the change by still guaranteeing safety. The proposed technique supports decisions about how to compensate a change by modifying a set of requirements instead of needing to change an implementation. The technique is based on a formal safety model using contracts to express fault containment properties and safety mechanisms. Although the author claims that the confidence in the safety of the system after the change is incorporated is identical or higher compared with the current practiced approach, there is no reflection of this claim on the safety case.

As for the state-of-the-practice, Nair et al., in [84] conducted a survey to determine practitioners' perspectives and practices on safety evidence management. The survey consists of six questions, where one of the questions is: How is evidence change managed? The aim of this question is to identify industrial practices for managing evidence evolution and performing evidence change impact analysis. A total of 52 practitioners from 15 countries and 11 application domains responded. The authors further analyse practices for safety evidence change management and give insights into the current challenges that practitioners face in terms of safety evidence provision. When asked about how they analyse the effect of the change of a piece of evidence on other pieces, 46% of the respondents noted manual checks according to some predefined process. Approximately the same percentage of respondents replied that the effect is checked manually without following any predefined process. The survey suggests that evidence change management is mainly performed manually and highlights the need for further analyses.

3.3 Problem Description

Safety assurance and certification are amongst the most expensive and time-consuming tasks in the development of safety-critical embedded systems [85]. Changes to those systems may require safety engineers to review and maintain the safe performance of their systems and repeat the certification process. Thereby, the cost of system changes including the cost of the activities that will follow them, such as regression testing, exacerbate the problems of cost and time for safety certification.

Safety case maintenance is more complicated task than it might first appear.

This is because change requests should be assessed before decision makers decide whether or not to accept them. The assessment should reveal if the change can cause unreasonable risks, and the required cost to implement the change. Hence, system developers should understand the change and the potential risks that it might carry before they identify the impacted parts. For example, a change might turn some implicit assumptions about the context in which a system should operate to be wrong. Misunderstanding the change might lead to skip those parts of the system which are dependent on that assumptions. Also, the developers need to understand the dependencies between the system parts to identify the affected parts correctly. For example, the effect of a change can propagate to other parts of the system — creating a ripple effect — and cause unforeseen violations of the acceptable safety limits. If the impact of change is not clear, developers might be conservative and do wider analyses and verification (i.e., check more elements than strictly necessary), and this will exacerbate the cost problem of safety cases. It is also necessary for the developers to describe how the change affects the system parts — that are listed as affected — in order to correctly estimate the cost of the response to that change. Otherwise, the response to a change might generate unplanned further changes to which the system must again respond [86], and this requires more cost than originally calculated.

One of the biggest challenges that affects safety case maintenance is that safety cases feature highly dependent elements. That is, safety goals, evidence, argument, and assumptions about operating context are highly interdependent. Hence, seemingly minor changes may have a major impact on the contents and structure of the safety argument. The lack of documentation of dependencies among the contents of safety cases is deemed as a challenge for safety case maintenance.

Moreover, the lack of traceability between a system and its safety case is another challenge that can impede safety case maintenance. More specifically, system developers need both top-down and bottom-up impact analysis approaches to maintain safety cases. A top-down approach is dedicated for analysing the impacted artefacts from the system domain down to the safety argument. In contrast, a bottom-up approach is dedicated for analysing impacted elements from the argument to the corresponding artefacts such as a safety analysis report, test results or requirements specification, etc. The lack of systematic and methodical approaches to analysing impact of change is a key reason behind the maintenance difficulties. However, conducting any style of impact analysis requires a traceability mechanism between the system domain and its safety case.

Safety case maintenance is inevitable task for many safety critical systems. The result of this task can communicate false status of the actual safe performance of a system if it is done incorrectly (intentionally or unintentionally). This false status, in the worst case, might not be detected by safety assessors, which can lead to certifying unsafe system. Many safety engineers are experiencing difficulties with safety case maintenance [5]. One reason for that is because they lack a systematic and methodical approach to identify the impact of change on safety cases.

Furthermore, using safety standards is one of the ways to control the risks in safety critical systems. Nowadays, safety standards heavily guide the development of these systems and form the basis for their approval and certification [60]. Despite clear recommendations to adequately maintain the systems and their safety cases by safety standards, existing standards offer little or no advice on how such operations can be carried out [5].

Hence, there is an increasing need for globally acceptable systematic and methodical approaches to facilitate the safety case maintenance without incurring disproportionate cost compared to the size of the change.

3.4 Research Goal

In this section, we derive the research questions that should address the identified problems as described in Section 3.3. We first identify the goal of our research and further break it down to pertinent research questions.

Based on the lack of standardised methods and techniques to enable change accommodation in safety critical systems and safety cases, the overall research goal of this thesis is to:

Design new techniques to facilitate the maintainability of safety cases due to system changes.

We refer to *changes* as modifications in the system due to anomalies, removals, additions, enhancements of components or subsystems.

We also use the adapted definition in Section 3.1 (i.e., *the degree of effectiveness and efficiency with which a safety case can be modified by the intended maintainers*) to refer to *maintainability of safety cases*. The maintainability degree is said to be high whenever the following three activities are done efficiently and effectively:

1. Identifying the impacted elements and those that are not impacted.
2. Minimising the number of impacted safety case elements.
3. Reducing the work needed to make the impacted safety case elements valid again.

The work in this thesis does not aim to measure the efficiency of achieving the three activities, but rather it strives to enable them and improve on them.

We refine our broad research goal by breaking it into four research questions (RQs), as follows:

RQ1: ***How can safety contracts be used to enable maintainability of safety cases?***

A component contract is a pair of properties, called the assumption, which must be satisfied by the component environment, and the guarantee, which must be satisfied by the component implementation when the assumption holds [87]. It is claimed that the cost of maintaining, reusing and changing software components is lessened while using contracts as developers may rework software components with knowledge of the constraints placed upon them [14]. Contract-based maintenance (or contract-based change management) can be a promising approach for safety critical systems [11, 12, 13]. In the literature, however, there is no consensus on what role the contracts should play in the overall change management process, how and where they should be derived, what they should contain, and how they should be presented. Also, there are different suggestions regarding the association of the contracts with the different properties of safety critical systems.

RQ2: ***How can the number of the impacted safety case elements be minimised so that the re-verification effort needed to make the impacted elements valid again be reduced?***

Changes to a system or its environment often necessitate tremendous re-verification activities. To reduce the re-verification efforts, the change propagation (aka ripple effect [88] or snowball effect [89]) should be as low as practicably possible. Hence, it is important for any proposal aims at facilitating system changes to contain (i.e., localise) the impact of changes. More clearly, to alleviate the cost of updating both a system and its safety case due to a change, it is crucial to minimise the effects of that change and prevent these effects from propagating into other parts of the system as long as it is practically possible.

Fricke et al. [90] claim, according to their case studies, that accommodating changes can take 30% of the development efforts. If systems' developers do not understand the impact of change, then they have to be conservative and do wider verification (i.e., check more elements than strictly necessary). This wider verification can include massive re-engineering efforts, which will increase the maintenance cost. Any effective change management process should help systems' developers to understand how the change affects a particular element (i.e., component, claim, work product) so that they know what is needed to make it valid again.

RQ3: ***How can traceability between a system design and its safety case be established to highlight the impacted parts upon changes?***

Any approach intends to maintain safety cases due to system changes should investigate 1) how a given change impacts system components and 2) how the impacted system components are presented in the safety case. Hence, performing safety impact analysis is a significant step to understand the impact of changes, the underlying reason for change or root cause, and the proposed solution in terms of the existing system and its constraints and requirements.

We refer to the ability to relate safety argument fragments to system design components as component traceability (through a safety argument). We refer to evidence across a system's artefacts as evidence traceability. Enabling component and evidence traceability is very useful to analyse the impact of change on a safety argument, and eventually, facilitates the overall maintenance of the safety case. Current standards and analysis techniques assume a top-down development approach to system design. The structure of safety arguments does not necessarily follow the same structure of the system design. Hence, it is unreasonable to assume a structured mapping between the elements of a system design and the elements of the safety argument related to that design. GSN can assist system developers to mechanically propagate the change through the goal structure and scope areas affected by a particular change. Using GSN makes capturing the underlying rationale of the argument easier since it clearly demonstrates the argument elements and their relationships. However, GSN is not enough since it does not tell if the suspect elements of an argument are still valid and expert judgements are still needed.

RQ4: ***How to incorporate safety case maintenance and through-life assurance into development and operational processes?***

Safety cases can be more readily maintained when using process models that encourage evolutionary development of the safety case in parallel with system development, and through explicitly recording applied engineering judgment and experience [71]. To increase the degree of effectiveness and efficiency with which a safety case can be modified, system development processes should consider and support safety case maintenance during the lifecycle of safety critical systems. For instance, some software corrective changes (e.g., bug fixes) can change several items of evidence (i.e., work products) that support safety claims in the safety case. If the development process does not mandate adequate activities to establish and maintain the integrity of the impacted evidence and update the safety case, there will be a gap between the documented safety reasoning in the safety case and the actual safety of the system. However, safety case is provided not only during initial development and deployment, but also at runtime based on operational data [60]. Hence, the operational processes should also mandate activities to continuously assessing and evolving the safety case according to the collected operational data. Such activities can reinforce a safety case for through-life safety assurance.

3.5 Research Methodology

Research is an investigation to find solutions to scientific and social problems through objective and systematic analysis. Research methods are basically all the methods (e.g., theoretical procedures, experimental studies, numerical schemes, statistical approaches) that are used by a researcher during a research study [91]. In this section, we demonstrate the process of our research and the followed research method. Figure 3.2 describes the process step by step.

Our research work started with an initial literature review, where we looked into both the state-of-the-art and the state-of-the-practice regarding the maintainability of safety critical systems and safety cases under system changes. Our literature review revealed that the maintenance of safety cases is significant and required by many safety standards but it has received no or little support yet. This lead us to formulate the problem statement, which we used later to derive a generic research goal (i.e., maintain safety cases after introducing a system change). Since more clarity of ideas, assumptions and practices can be acquired through study of literature, we started digging deeper and chasing the challenges of safety cases maintenance. After many iterations of the

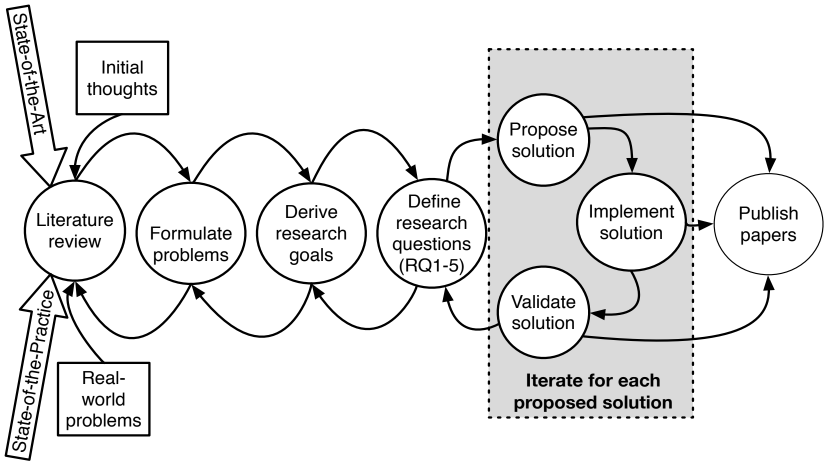


Figure 3.2: Overview of our research process

literature survey and problem formulation, we derived the main research goal, which is *"Design new change management techniques in order to facilitate the maintainability of safety cases due to system changes"*.

Since changes to safety critical systems are of different kinds and they can cause different effects, there is no one technique or methodology that can respond to all kinds of changes. However, we focused on the tools and mechanisms that can facilitate any change management process. To this end, we refined our overall research goal by defining five research questions, which investigated how safety contracts can be exploited to facilitate the maintenance of safety cases. To the best of our knowledge neither the state-of-the-art nor the state-of-the-practice contain supporting processes or methods that provide detailed steps of how to analyse the impact of change on safety cases using safety contracts and sensitivity analysis.

For our research questions, we used the iterative process presented in the grey dotted box in Figure 3.2. We started every iteration by proposing a solution that might answer one or multiple research questions. We implemented each solution by providing a motivation of the solution as well as a detailed description, which usually takes the form of a technique, method or algorithm. Each solution was assessed and validated by a case study, where we listed the limitations and the possible improvements. We evaluated the limitations, after each proposed solution, with respect to the research questions and proposed a new solution that should address these limitations. The new proposed solution is not designed to address the limitations of the previous solutions only, but it can also answer different research questions.

The direct result of our proposed approaches and techniques is a series of published research papers that communicate our contributions to the research community.

Chapter 4

Research Contributions

This chapter provides an overview of the contributions included in this thesis. First, an overview is provided to describe the different contributions by each paper (included ***Part II***). Second, we group the technical contributions presented in this thesis into five main contributions. Figure 4.1 shows how the main contributions answer the research questions (in Section 3.4), and ultimately, achieve the research goal.

4.1 Contributions of the Included Papers

- ***Paper A:*** *Using Sensitivity Analysis to Facilitate The Maintenance of Safety Cases*

  One contribution of the paper is to introduce a maintenance technique named Sensitivity ANalysis for Enabling Safety Argument Maintenance (SANESAM), which derives safety contracts from FTA using sensitivity analysis. SANESAM combines sensitivity analysis together with safety contracts to identify the sensitive parts of a system and highlight these parts to help the experts to make an educated decision as to whether or not apply changes. Also, since considering a complete list of anticipated changes is difficult, the paper shows how to determine the flexibility (or compliance) of each component to changes. This means that regardless of the type of changes, the change will be seen as a factor to increase or decrease a certain parameter value. Thus system developers can focus more on predicting those changes that might make the parameter value

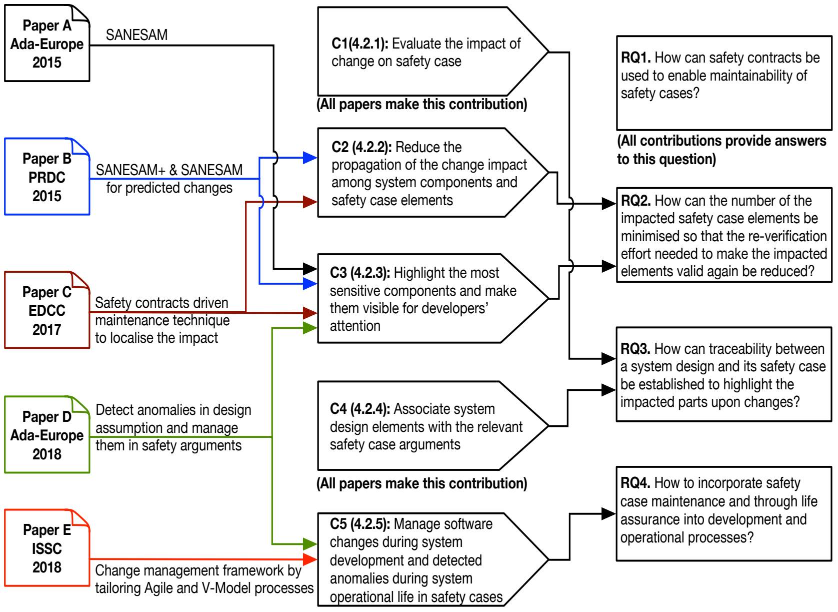

Paper A
Ada-Europe
2015
SANESAM
Paper B
PRDC
2015
SANESAM+ & SANESAM
for predicted changes
Paper C
EDCC
2017
Safety contracts driven
maintenance technique
to localise the impact
Paper D
Ada-Europe
2018
Detect anomalies in design
assumption and manage
them in safety arguments
Paper E
ISSC
2018
Change management framework by
tailoring Agile and V-Model processes
C1(4.2.1): Evaluate the impact of
change on safety case
(All papers make this contribution)
C2 (4.2.2): Reduce the
propagation of the change impact
among system components and
safety case elements
C3 (4.2.3): Highlight the most
sensitive components and make
them visible for developers'
attention
C4 (4.2.4): Associate system
design elements with the relevant
safety case arguments
(All papers make this contribution)
C5 (4.2.5): Manage software
changes during system
development and detected
anomalies during system
operational life in safety cases
RQ1. How can safety contracts be
used to enable maintainability of
safety cases?
(All contributions provide answers
to this question)
RQ2. How can the number of the
impacted safety case elements be
minimised so that the re-verification
effort needed to make the impacted
elements valid again be reduced?
RQ3. How can traceability between
a system design and its safety case
be established to highlight the
impacted parts upon changes?
RQ4. How to incorporate safety
case maintenance and through life
assurance into development and
operational processes?

Figure 4.1: The connections between the published papers, research questions and contributions

inadmissible. Another contribution by the paper is introducing a traceability mechanism between a structural design of a system and its safety case to improve the change impact analysis on the safety case. More clearly, the paper provides a set of steps which guides system developers to associate the derived contracts with particular elements of the safety case argument. The association includes a versioning management approach, which stores additional information (e.g., artefact version number) into the safety contracts that are associated with the safety argument.

- ***Paper B:*** *Deriving Hierarchical Safety Contracts*

  The main contribution of the paper is to identify some limitations to SANESAM technique (Paper A), and suggest two options as extensions to resolve these limitations. The first option is SANESAM+, which is useful in the case of arbitrary changes because it calculates the Failure Probability (FP) for all events in the FTA regardless of any change scenario. SANESAM+ assumes that all events in FTA may change at a time. Hence, the technique requires measuring the sensitivity of all events in the FTA, which means that the Maximum Allowed Failure Probability (MAFP) should be calculated for each event. The second option is SANESAM+ *For Predicted Changes*, this option excludes the events that are unlikely to change and it increases the FP for only the events that are associated to a predicted change. A derived safety contract by SANESAM+ by *Predicted Changes* can guarantee higher FP than the guaranteed FP (for the same event and using the same set of assumptions) in a derived safety contract by SANESAM+. Hence, the derived safety contracts by SANESAM+ *For Predicted Changes* are more tolerant and robust than those derived by SANESAM+ but it requires in-advance prediction of changes.

- ***Paper C:*** *Using Safety Contracts to Guide the Maintenance of Systems and Safety Cases*

  The main contribution of the paper is to introduce a new technique that can save huge efforts in re-verification or re-certification due to some design changes. The technique uses the key principle of SANESAM and SANESAM+ in Paper A and B, respectively, to contain (i.e., localise) the potential changes in the smallest (in terms of number of parts affected) possible segment in the system architecture and its safety case. More clearly, the new technique compares the calculated MAFP of the events with new estimated failure probability of those events due to a change. If a new estimate failure probability of an event is $\leq$ MAFP, then the change will not, necessarily, require a considerable system modification. However, if the estimate of the new failure probability is $>$ MAFP, then the technique investigates whether or not the deficits in the failure probability is containable by the budgeted failure rates of events in higher levels in the fault tree. The ripple effects of the change will stop at the event which will have enough margin in its failure probability to contain the deficit (i.e., change). The technique can serve as a first impact analysis layer that helps system's developers to estimate the required effort to accommodate a design change. It also guides the developers to avoid massive re-engineering efforts when it is not really needed.

- ***Paper D:*** *Using Safety Contracts to Verify Design Assumptions During Runtime*

  The main contribution of this paper is to introduce a novel runtime monitoring technique to detect the discrepancies between the failure rates of system's components during their operational life and their generic failure rates used for analysis and assurance during the design time. Since it is infeasible to monitor the failure rates of all components of a system, the technique utilises importance and sensitivity analyses to evaluate the criticality of the system components, and selects the most critical ones for monitoring. The technique derives safety contracts for the selected components and associate them with the relevant events in the FTA and the relevant parts in the safety case. If a discrepancy is detected between an observed failure rate (λ_O) and a generic failure rate (λ_G) of the same component, where $\lambda_O > \lambda_G$, then the relevant contract should be flagged and the referred parts of both the FTA and the safety case should be revisited. Detecting more precise measure of failure rates than the predicted ones will 1) improve the efficacy of the system design to reduce the risk (mitigate by design), 2) define stronger evidence (e.g., refine or rectify the test results) and 3) highlight the required preventive, corrective, perfective or adaptive maintenance for safer operation

The second contribution of the paper is to provide GSN argument patterns to help system's developers to argue more compelling over the failure rates of the monitored components in the light of the derived evidence from the operational phase (confidence from use). The suggested patterns take into consideration the association of the derived contracts from the FTA and recommend to associate these contracts with the included ACPs.

As a support for the concept of "Through-Life Safety Assurance", the paper shows how to utilise safety contracts to provide prescriptive data for what should be monitored, and what parts of the safety argument should be revisited to maintain system safety whenever a divergence between the design assumptions of system and its actual operation, is detected.

- ***Paper E:*** *A Safety-Centric Change Management Framework by Tailoring Agile and V-Model Processes*

  Following the V-model to accommodate system changes might be very strict, which might be justifiable for structural system changes since many parts get impacted and no accurate estimation for the size of work needed to maintain the system. For software non-structural changes, this might not be justifiable. The main contribution of this paper is to propose XP-Kan-Safe as a novel maintenance framework to facilitate the accommodation process of software changes in safety critical systems by utilising the strengths of agile methods and the V-model. More clearly, we reconcile the known effective validation and verification process of the V-model to the known effective practices and the TDD process of agile methods. XP-Kan-Safe comprises two main processes. The first process is called the preliminary process and its main objective is to derive safety contracts and enrich them with additional information to enhance the traceability between the safety requirements (i.e., guarantees) and different related artefacts (i.e., test suites). The preliminary process is applicable to any development process that decomposes safety requirements from the very high level the lowest possible level. The second process is called the change management and it contains ten activities to guide system developers to accommodate software changes by using the derived safety contracts during the preliminary process.

4.2 Main Contributions

The contributions presented in this thesis can be grouped into five main contributions.

4.2.1 Evaluate the impact of change on safety case

Changes to safety critical systems can require a tremendous effort to accommodate. The size of a change's effect varies based on the nature of the change. Ideally, the amount of work needed to accommodate a change should commensurate with the size of this change. Impact analysts shall determine the robustness of their system against a give change request. They should provide the decision makers sufficient information about the size of impact in order to decide wether or not accept the change request.

We propose different techniques which serve as a first impact analysis layer that helps system's developers to estimate the size of effort needed to accommodate a design change. In Paper A, we introduce a Sensitivity ANalysis for Enabling Safety Argument Maintenance technique (*SANESAM*) that supports system engineers to accommodate changes to the reliability figures of system components. We also propose *SANESAM+* and *SANESAM+ for Predicted Changes* in paper B as a modified version of *SANESAM* that covers wider variety of changes. The main objective of the techniques is to help the experts to make an educated decision as to whether or not apply changes. This decision is in light of beforehand knowledge of the impact of these changes on the system and its safety case. Using the techniques helps to bring to developers' attention the most sensitive parts of a system for a particular change.

The key principle of *SANESAM*, *SANESAM+* and *SANESAM+ for Predicted Changes* is to determine the flexibility (or robustness) of a system to changes using sensitivity analysis. The techniques determine, for each component, the allowed range (i.e., thresholds) for a certain parameter within which a component may change before it compromises a certain system property (e.g., safety, reliability, etc.). Sensitivity analysis is utilised as a means to determine the range of failure probability parameter for each component. Hence, the techniques assume the existence of a probabilistic FTA where each event in the tree is specified by an actual (i.e., current) failure probability $FP_{Actual|event(x)}$. In addition, the techniques assume the existence of the required failure probability for the top event $FP_{Required(Topevent)}$, where the FTA is considered unacceptable if: $FP_{Actual(Topevent)} > FP_{Required(Topevent)}$.

As part of *SANESAM*, *SANESAM+* and *SANESAM+ for Predicted Changes*, safety contracts are derived to 1) highlight the sensitive events to make them visible up front for developers attention and 2) to record the dependencies between the sensitive events and the other events in the FTA. Hence, if any contracted event has received a change that necessitates increasing its failure probability where the increment is still within the defined threshold in the contract, then it can be said that the contract(s) in question still holds (intact) and the change is containable with no further maintenance.

One difference between the three techniques is based on the number of changes addressed by each one. For example, *SANESAM* calculates the MAFP for all events in FTA but it does not allow multiple changes at a time, *SANESAM+* calculates the MAFP for all events in FTA but it allows multiple changes at a time. Finally, *SANESAM+ for Predicted Changes* calculates the MAFP for particular events based on a predicted change and it allows multiple changes at a time but only to those events.

In paper C, we propose another technique which relies on *SANESAM* and *SANESAM+* but the objective of the technique is not limited to determine the robustness of a system against the potential changes (see Section 4.2.2). Moreover, the proposed technique in paper D also utilises *SANESAM* to determine the robustness of system models against changes (see Section 4.2.5).

In paper E, we propose a new technique, different from the sensitivity-based techniques discussed above, to evaluate the impact of potential software changes in a system. The main objective of this technique is to enable tri-directional impact analysis process between safety requirements (i.e., guarantees), system design elements and the related test suites. The technique derives safety contracts to capture the dependencies among the safety requirements, and enriches the derived contracts with additional traceability information to map the guaranteed requirements with the related system design elements, test cases and safety case elements. Using this additional information helps impact analysts to evaluate the impact of changes on the system design once they know the entry points of the changes in safety requirements or test cases and vice versa.

4.2.2 Reduce the propagation of the change impact among system components and safety case elements

If system developers underestimate the actual impact of a change, there will be a considerable probability to overlook arising failures which can compromise system safety. If system's developers, on the other hand, overestimate the impact of change, there maybe no impact on safety but the cost will be higher than required.

SANESAM, *SANESAM+* and *SANESAM+ for Predicted Changes* can identify the propagation of the change impact through the guaranteed thresholds in the derived safety contracts. However, if a contract is broken due to a greater impact than what is actually assumed by this contract, these techniques do not investigate how to brace this contract to make it more robust.

After approving a change request, a big amount of time is spent to ensure that the system is still safe and all potential work products, i.e., items of evidence which are impacted, are maintained. Minimising the impact of change to the smallest possible part within a system will reduce the work required to verify the system and speed up the change management process. In paper C, we propose another technique to contain (i.e., localise) the potential changes in the smallest possible part of a system. This technique utilises the same rules by which *SANESAM+* and *SANESAM+* calculate the sensitivities and associate them with a safety argument via safety contracts. However, the technique adds additional steps to enable effective usage of the safety margins in a probabilistic FTA. More clearly, the technique compares the calculated MAFP of the events with new estimated FP after introducing a change. If the change's effect (i.e., difference between the new estimated FP and the MAFP) is not containable in the safety contract of the impacted event, then the safety contract of the ancestor event should be investigated as whether or not it can contain it. If the change's effect still cannot be contained by the ancestor, safety contracts in one more level up should be investigated and so on and so forth until

Figure 4.2: Safety contract notation

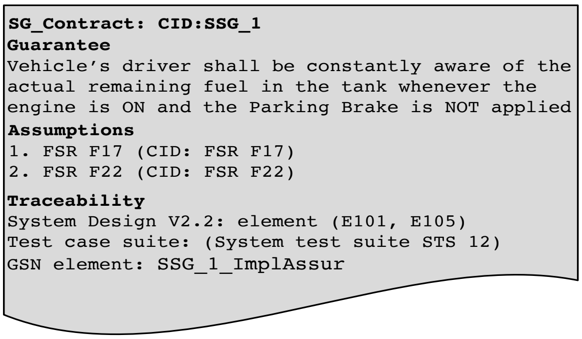
a safety contract contains it. Once the contract which contains the change's effect is identified, all associated claims with this contract together with their supporting arguments and evidence should be highlighted as suspect.

4.2.3 Highlight the most sensitive components and make them visible for developers' attention

Changes are often only performed years after the initial design of the system making it hard for system's developers performing the changes to know which parts of the system or the safety case are affected. We propose using safety contract to document (i.e., highlight) the most critical and sensitive components to changes to make them visible up front for developers' attention. In other words, we use safety contracts to highlight prescriptive data for what is needed to be revisited and verified to maintain system safety when changes happen. In papers A, B and C we propose to derive safety contracts from FTA to highlight the most sensitive FTA events to changes. To this end, we introduce a contract notation to annotate the contracted events in FTA, where every derived contract should have a unique identifier as shown in Figure 4.2.

In paper D, we propose to derive safety contracts from FTA to highlight the most important components to make them visible up front for developers' attention. Furthermore, the derived contracts is also used to record the thresholds of $\lambda_D(i)$ to continuously compare them with the monitoring results (λ_{D_O}). Hence, if λ_{D_O} of component i exceeds the guaranteed $\lambda_{D\_Max}(i)$ in the contract of that component, then we can infer that the contract in question is broken and the related FTA should be re-assessed in the light of the λ_{D_O}.

in Paper E, we propose to derive safety contracts from FTA and FMEA to highlight the generated safety requirements. These safety contracts should also highlight additional traceability information and make visible up front for developers' attention. Figure 4.3 presents an example of a derived safety contract and the suggested traceability information.



```
SG_Contract: CID:SSG_1
Guarantee
Vehicle's driver shall be constantly aware of the
actual remaining fuel in the tank whenever the
engine is ON and the Parking Brake is NOT applied
Assumptions
1. FSR F17 (CID: FSR F17)
2. FSR F22 (CID: FSR F22)
Traceability
System Design V2.2: element (E101, E105)
Test case suite: (System test suite STS 12)
GSN element: SSG_1_ImplAssur
```

Figure 4.3: Traceability information in a safety contract [92]

4.2.4 Associate system design elements with the relevant safety case arguments

A traceability between the system domain and its safety case should be established to highlight the impacted parts in one side whenever the other side changes. To this end, we use the same contract notation, which we suggested to use in FTAs (in Figures 4.2 and 4.3), to associate the system elements with the relevant elements in the safety case arguments. Figure 4.4 show an example of how the related FTA events to a specific system components are associated with the relevant safety case argument. The mapping between safety contracts from the system domain (e.g., safety contracts in FTAs) with the related safety contracts in the safety case enables a bi-directional safety impact analysis. This means that once the entry point of a change is pinpointed in the safety case, the logic of the safety case argument can be used to evaluate the impact of this change in the system model and vice versa. It is worth noting that the safety contract notation shall not affect the way GSN is being produced but it brings additional information for developers' attention.

4.2.5 Manage software changes during system development and detected anomalies during system operational life in safety cases

Observing anomalies in design assumptions during testing or runtime is deemed as a change that might need to be considered based on its criticality.

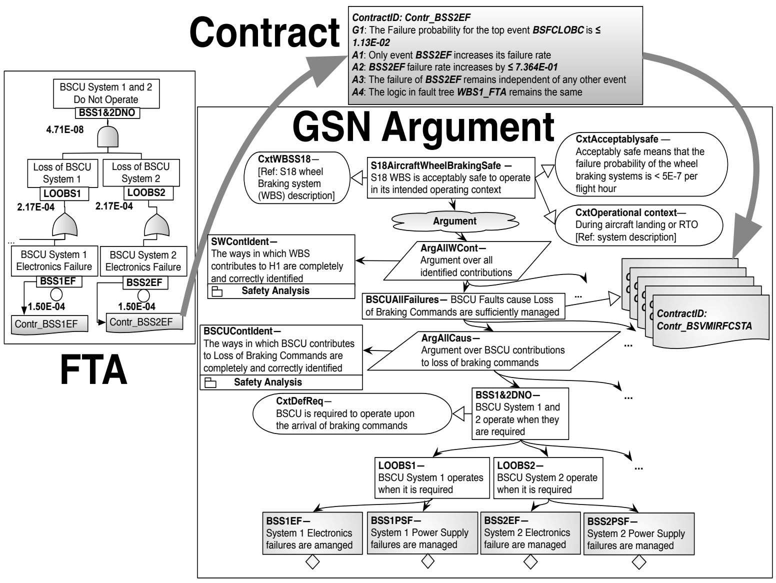

Contract
ContractID: Contr_BSS2EF
G1: The Failure probability for the top event BSFCLOBC is ≤ 1.13E-02
A1: Only event BSS2EF increases its failure rate
A2: BSS2EF failure rate increases by ≤ 7.364E-01
A3: The failure of BSS2EF remains independent of any other event
A4: The logic in fault tree WBS1_FTA remains the same
BSCU System 1 and 2 Do Not Operate
BSS1&2DNO
4.71E-08
Loss of BSCU System 1
LOOBS1
Loss of BSCU System 2
LOOBS2
2.17E-04
2.17E-04
BSCU System 1 Electronics Failure
BSS1EF
BSCU System 2 Electronics Failure
BSS2EF
1.50E-04
1.50E-04
Contr_BSS1EF
Contr_BSS2EF
FTA
GSN Argument
CxtWBSS18— [Ref: S18 wheel Braking system (WBS) description]
S18AircraftWheelBrakingSafe — S18 WBS is acceptably safe to operate in its intended operating context
CxtAcceptablysafe — Acceptably safe means that the failure probability of the wheel braking systems is < 5E-7 per flight hour
CxtOperational context— During aircraft landing or RTO [Ref: system description]
Argument
SWContIdent— The ways in which WBS contributes to H1 are completely and correctly identified
Safety Analysis
ArgAllWCont— Argument over all identified contributions
BSCUAllFailures— BSCU Faults cause Loss of Braking Commands are sufficiently managed
ContractID: Contr_BSVMIRFCSTA
BSCUContIdent— The ways in which BSCU contributes to Loss of Braking Commands are completely and correctly identified
Safety Analysis
ArgAllCaus— Argument over BSCU contributions to loss of braking commands
CxtDefReq— BSCU is required to operate upon the arrival of braking commands
BSS1&2DNO— BSCU System 1 and 2 operate when they are required
LOOBS1— BSCU System 1 operates when it is required
LOOBS2— BSCU System 2 operate when it is required
BSS1EF— System 1 Electronics failures are amanged
BSS1PSF— System 1 Power Supply failures are managed
BSS2EF— System 2 Electronics failure are managed
BSS2PSF— System 2 Power Supply failures are managed

Figure 4.4: An example of an association of an FTA event with a safety case argument

As the system evolves after deployment, there could be a mismatch between our communicated understanding of the system safety by the safety case and the safety performance of the system in actual operation, which might invalidate many of the prior assumptions made, undermine the collected items of evidence and thus defeat safety claims [60]. In paper E, we propose a novel technique to detect the discrepancies between the failure rates of system's components during their operational life and their generic failure rates used for analysis and assurance during the design time. Since it is infeasible to monitor the failure rates of all components of a system, the technique utilises probabilistic FTA to evaluate the criticality of the system components, and selects the most critical ones for monitoring. The technique derives safety contracts for the selected components and associate them with the relevant events in the FTA and the relevant parts in the safety case. If a discrepancy is detected between an observed failure rate (λ_O) and a generic failure rate (λ_G) of the same component, where $\lambda_O > \lambda_G$, then the relevant contract should be flagged and the referred parts of both the FTA and the safety case should be revisited. The technique also checks wether the changes to the assumed failure rates during runtime is containable without the need to make further changes.

In paper E, we propose and describe XP-Kan-Safe as a safety-centric change management framework. The main idea of XP-Kan-Safe is to incorporate an agile based change management in the development process of safety critical systems to systematically assess and implement software changes. XP-Kan-Safe utilises safety contracts as (1) stitches that connect the V-model, Extreme Programming (XP) and Kanban into our tailored process, and (2) means to enable a tri-directional impact analysis process. More specifically, the framework derives safety contracts from safety analyses (FTA and FMEA) to capture different levels of safety requirements decompositions. A guaranteed requirement at one level (e.g., functional safety requirement) is realised by assumptions that represent lower level requirements (e.g., technical safety requirements). The derived safety contracts record the dependencies among safety requirements. Additionally, the contracts comprise traceability related information which associates the safety requirements with the relevant design elements, safety argument elements and test suites. Hence, the safety contracts can efficiently support the impact analysis upon changes to safety requirements, safety argument or test suites (i.e., code level).

Chapter 5

Conclusions and Future Work

5.1 Conclusions

The main objectives of safety cases are to justify and communicate that a system is acceptably safe to operate in a specific context. The validity of a safety case relies on assumptions about the safe performance of a system as well as the assumptions made about the context in which that system should operate. Changes to these assumptions will lead to a mismatch between the real safe performance of a system and the communicated safe performance in its safety case. Thus, safety case should be maintained as a living document that should always justify and communicate the real safe performance of the system. Moreover, changes to safety critical systems lead to a re-certification process in which system developers review an existing safety case and maintain it by identifying its impacted parts, checking the validity of the safety arguments and generating a new set of evidence. If system developers are unable to identify the actual impacted parts in the safety case, they might re-execute more verification and validation activities than strictly necessary.

Safety case maintenance is not an easy task since changes are often performed years after the initial design of the system making it hard for system developers to know which parts of the safety case are affected. Also, system developers need to understand the dependencies between the different elements of safety case and identify the affected parts correctly. Hence, there is a pressing need for effective methods and techniques to enable efficient safety case maintenance without incurring disproportionate cost compared to the size of the change.

In this thesis we have designed new techniques to facilitate the maintainability of safety cases due to system changes. These techniques utilise safety contracts to provide feasible solutions to safety case maintenance such as helping to understand the indirect impact of a change on the safety case and providing a means to sufficiently record the dependencies between safety case contents and

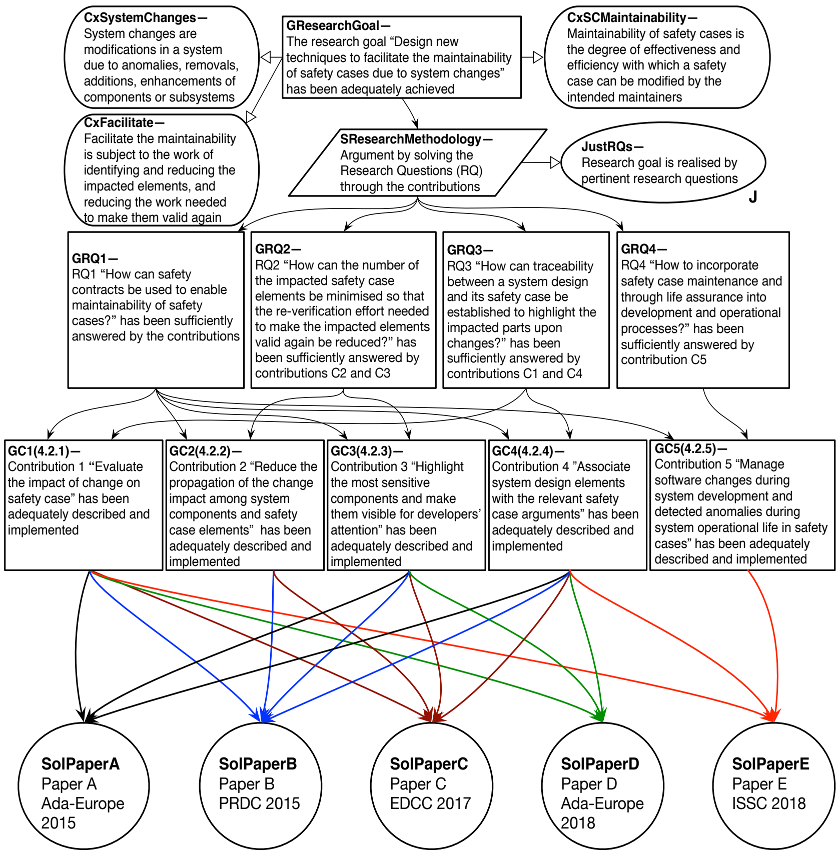

CxSystemChanges— System changes are modifications in a system due to anomalies, removals, additions, enhancements of components or subsystems
GResearchGoal— The research goal "Design new techniques to facilitate the maintainability of safety cases due to system changes" has been adequately achieved
CxSCMaintainability— Maintainability of safety cases is the degree of effectiveness and efficiency with which a safety case can be modified by the intended maintainers
CxFacilitate— Facilitate the maintainability is subject to the work of identifying and reducing the impacted elements, and reducing the work needed to make them valid again
SResearchMethodology— Argument by solving the Research Questions (RQ) through the contributions
JustRQs— Research goal is realised by pertinent research questions
J
GRQ1— RQ1 "How can safety contracts be used to enable maintainability of safety cases?" has been sufficiently answered by the contributions
GRQ2— RQ2 "How can the number of the impacted safety case elements be minimised so that the re-verification effort needed to make the impacted elements valid again be reduced?" has been sufficiently answered by contributions C2 and C3
GRQ3— RQ3 "How can traceability between a system design and its safety case be established to highlight the impacted parts upon changes?" has been sufficiently answered by contributions C1 and C4
GRQ4— RQ4 "How to incorporate safety case maintenance and through life assurance into development and operational processes?" has been sufficiently answered by contribution C5
GC1(4.2.1)— Contribution 1 "Evaluate the impact of change on safety case" has been adequately described and implemented
GC2(4.2.2)— Contribution 2 "Reduce the propagation of the change impact among system components and safety case elements" has been adequately described and implemented
GC3(4.2.3)— Contribution 3 "Highlight the most sensitive components and make them visible for developers' attention" has been adequately described and implemented
GC4(4.2.4)— Contribution 4 "Associate system design elements with the relevant safety case arguments" has been adequately described and implemented
GC5(4.2.5)— Contribution 5 "Manage software changes during system development and detected anomalies during system operational life in safety cases" has been adequately described and implemented
SolPaperA Paper A Ada-Europe 2015
SolPaperB Paper B PRDC 2015
SolPaperC Paper C EDCC 2017
SolPaperD Paper D Ada-Europe 2018
SolPaperE Paper E ISSC 2018

Figure 5.1: The logic of achieving the research goal

system elements. Additionally, the techniques offer new concepts of detecting change propagation, establish bidirectional traceability between a system and its safety case, and highlight the most sensitive system parts to changes. Figure 5.1 presents a GSN argument which explains how the research goal of the thesis is achieved.

SANESAM was designed to determine, for each component, the allowed range for a certain parameter within which a component may change before it compromises a certain system property (e.g., safety, reliability, etc.). The allowed ranges for different components result in safety contracts and if a contract is not violated by a change introduced to the system, the change can be easily contained. The contracts are mapped to a safety case argument and used in justification of the safe performance of the system. SANESAM+ and SANESAM for predicted changes are extensions to SANESAM and they provide more freedom by considering multiple events at a time and more robustness against predicted changes, respectively. We also designed a safety contracts driven maintenance technique to contain change impact in the smallest possible set of components to ultimately prevent (or minimise) the ripple of these effects from propagation. A Wheel Braking System (WBS) was used as a case study to evaluate the feasibility of the application of the described techniques so far. The techniques under discussion contributed to evaluate the impact of change on safety case, reduce the propagation of the change impact among system components and safety case elements, and highlight the most sensitive components and make them visible for developers‘ attention.

We designed another technique which also utilises safety contracts to continuously reassess the failure rates of the most critical components to safety and use the results to suggest system changes or maintenance. The main objective of the technique is to monitor the runtime of a system and detect the divergence between the failure rates (which were used in the safety analyses) and the observed failure rates in the operational life. The technique uses sensitivity analysis to define the maximum allowed deviation in the failure rate of a critical component before it compromises the safety requirements. Subsequently, a safety contract is derived to guarantee the maximum allowed deviation in the failure rate, and a monitoring algorithm continuously checks if the failure rate exceeds its acceptable threshold (i.e., the guaranteed failure rate). Any violated safety contract will indicate the affected part in the safety case. This technique enables through-life safety assurance by providing prescriptive data for what should be monitored, and what parts of the safety case arguments should be updated to maintain system safety when a divergence (i.e., anomaly) is detected in safety cases and during system operational life. We used an Automated

Guided Vehicle (AGV) system as a case study to evaluate the feasibility of the technique.

Finally, we designed a safety-centric change management framework called *XP-Kan-Safe* by tailoring agile and V-model processes. *XP-Kan-Safe* is a novel maintenance framework to facilitate the software change management process in safety critical systems. More clearly, we reconciled the known effective validation & verification process of the V-model to the known effective practices and the TDD process of agile methods. We used safety contracts as 1) stitches that connect the V-model, Extreme Programming (XP) and Kanban into our tailored process, and 2) means to enable a tri-directional impact analysis process (i.e., traceability between safety requirements, test suites and safety case). XP-Kan-Safe incorporates safety case maintenance into the software development process to manage software changes during system development in safety cases.

As a summary, the work in this thesis showed how safety contracts can be utilised to support the maintainability of safety cases. The work showed how the contracts can be derived, how they can be presented, what they might contain, and where to place them.

5.2 Future Research Directions

The designed techniques in this thesis provided different solutions to facilitate the accommodation of changes to system reliability as well as changes to the software of safety critical systems. However, several research directions remain for the future work. We envision three categories of research directions in the future, namely, validation, automation and extension:

- **Validation**: One direction for the future work is to carry out an industrial validation for the designed techniques. Such a validation should focus on measuring the feasibility and efficacy of using safety contracts for safety case maintenance as described in SANESAM and SANESAM+. Also, since failure rates are not constant across different applications, where conservative developers might assume less reliable failure rates than what is really reported, the future work should also validate the efficacy of our monitoring technique for reassessing more precise failure rates.

  An industrial validation of XP-Kan-Safe is also a direction for the future work. The designed framework can be validated in the context of AUTOSAR (AUTomotive Open System ARchitecture) based systems [93].

One of the purposes of AUTOSAR is to satisfy the need for modularity of the architectural components and their implementations, which facilitates the exchange of these components between different parties. Choosing to develop automotive systems based on the AUTOSAR standard requires continuous adoption of new AUTOSAR releases in the development projects in order to enable new innovative solutions in cars [94]. This means that there is a need for performing change impact analysis on AUTOSAR systems after each release of AUTOSAR standard and maintain safety cases to consider the new changes. XP-Kan-Safe can help to document the dependencies among AUTOSAR components in different layers (i.e., Basic Software, Runtime Environment and Software Applications), establish bidirectional traceability between safety requirements, system model, validation and verification artefacts, and the safety case arguments.

- **Automation**: Another direction for the future work is to automate the application of the techniques. The techniques can be implemented within a software tool to automatically identify the sensitive components of a system, derive the safety contracts and create traceable links between system elements, safety analyses and safety case arguments. Obviously, formalising the safety contracts is not only useful to support effective automation of the techniques but it also makes the safety contracts more checkable for completeness, consistency and correctness.

- **Extension**: Another direction for the future work is to extend the techniques to other properties. To this point, only failure rates are considered by SANESAM and SANESAM+. This can be less useful while dealing with software changes as it is recognised as being difficult to quantify the reliability (e.g., failure probabilities) of software components. Another future work can be stemmed from this limitation, which is extending the techniques to cover wider variety of changes. In other words, the underlying logic of the techniques can be utilised to consider software properties to ultimately facilitate the maintainability of safety cases upon the changes to these properties. For instance, SANESAM and SANESAM+ can address the problem of the Worst Case Execution Time (WCET) as a property where sensitivity is judged in terms of its impact on the ability to meet the system's timing requirements and the required level of confidence. For example, sensitivity analysis can be used to derive safety contracts that guarantee the reliability R of Task T to meet its deadline assuming that the WCET of T is within τ (time unit). Such contracts can be associated with the parts of the safety case arguments to support the impact analysis in case if changes to the timing behaviour are either planned or detected.

Likewise, our designed monitoring technique can be extended to monitor other critical properties during the runtime to continuously maintain the documented confidence in safety cases [95]. Software temporal properties are good candidates for the technique (e.g., WCET), however some communication properties are also interesting to consider for monitoring. In [96], for example, we propose a methodology for assuring wireless cooperative functions of vehicular systems, where we suggest to identify and monitor all system parameters that may increase the communication failures (i.e., packet loss rate). This suggestion might be impracticable since it is costly and infeasible to monitor almost all parameters in cooperative functions. Applying importance and sensitivity analyses helps to identify a reasonable set of parameters for monitoring. Moreover, almost all proposed techniques for monitoring communication failures in the literature do not show how safety cases should be maintained according to the newly detected behaviours. Extending our monitoring technique and its algorithm to 1) monitor packet loss rate during runtime, 2) derive level of confidence, and 3) support a through-life safety assurance, is a reasonable direction for the future work.

Bibliography

[1] J.C. Knight. Safety critical systems: Challenges and directions. In *Proceedings of the 24rd International Conference on Software Engineering (ICSE).*, pages 547–550, May 2002.

[2] O. Jaradat. Enhancing the maintainability of safety cases using safety contracts, Mälardalen University, Västerås, Sweden. http://www.es.mdh.se/publications/4082-, November 2015.

[3] O. Jaradat, P. Graydon and I. Bate. An approach to maintaining safety case evidence after a system change. In *Proceedings of the 10th European Dependable Computing Conference (EDCC)*, UK, 2014.

[4] P. J. Graydon and C. M. Holloway. An investigation of proposed techniques for quantifying confidence in assurance arguments. *Safety Science*, 92(Supplement C):53 – 65, 2017.

[5] T. Kelly and J. McDermid. A systematic approach to safety case maintenance. In *Proceedings of the Computer Safety, Reliability and Security*, volume 1698 of *Lecture Notes in Computer Science*, pages 13–26. Springer Berlin Heidelberg, 1999.

[6] R. Maguire. *Safety Cases and Safety Reports: Meaning, Motivation and Management*. Ashgate Publishing, Ltd., 2012.

[7] U.K. Ministry of Defence, "JSP 430 - Ship Safety Management System Handbook", Ministry of Defence January 1996.

[8] T. Kelly. *Arguing Safety – A Systematic Approach to Managing Safety Cases*. PhD thesis, Department of Computer Science, University of York, 1998.

[9] Health and Safety Executive (HSE). *Railway Safety Cases - Railway (Safety Case) Regulations - Guidance on Regulations*, 1994.

[10] L. Benvenuti, A. Ferrari, E. Mazzi, and A. L. Vincentelli. Contract-based design for computation and verification of a closed-loop hybrid system. In *Proceedings of the 11th International Workshop on Hybrid Systems: Computation and Control*, HSCC '08, pages 58–71, Berlin, Heidelberg, 2008. Springer-Verlag.

[11] J. L. Fenn, R. Hawkins, P. J. Williams, T. Kelly, M. G. Banner, Y. Oakshott. The who, where, how, why and when of modular and incremental certification. In *Proceedings of the 2nd IET International Conference on System Safety*, pages 135–140. IET, 2007.

[12] P. Conmy, J. Carlson, R. Land, S. Björnander, O. Bridal, I. Bate. Extension of techniques for modular safety arguments. Deliverable d2.3.1, technical report, Safety certification of software-intensive systems with reusable components (SafeCer), 2012.

[13] P. Graydon and I. Bate. The nature and content of safety contracts: Challenges and suggestions for a way forward. In *Proceedings of the 20th IEEE Pacific Rim International Symposium on Dependable Computing (PRDC)*, November 2014.

[14] S. Bates, I. Bate, R. Hawkins, T. Kelly, J. McDermid, and R. Fletcher. Safety case architectures to complement a contract-based approach to designing safe systems. In *Proceedings of the 21st International System Safety Conference (ISSC)*, 2003.

[15] O. Jaradat, I. Bate, and S. Punnekkat. Using sensitivity analysis to facilitate the maintenance of safety cases. In *Proceedings of the 20th International Conference on Reliable Software Technologies (Ada-Europe)*, pages 162–176, June 2015.

[16] *Oxford Dictionary of English (3 ed.)*. Oxford University Press, 2010.

[17] Industrial Avionics Working Group (IAWG). Modular Software Safety Case (MSSC): Glossary, November 2012.

[18] J. Knight. *Fundamentals of Dependable Computing for Software Engineers*. Chapman & Hall/CRC, 1st edition, 2012.

[19] M. Dorfman and C. Anderson. *Aerospace software engineering: a collection of concepts*. American Institute of Aeronautics and Astronautics, Washington, DC, 1991.

[20] I. Sommerville. *Software Engineering*. Addison-Wesley, Harlow, England, 9 edition, 2010.

[21] ISO 26262:2011. Road Vehicles — Functional Safety, Part 1-9. International Organization for Standardization, Nov 2011.

[22] A. Avižienis, J.-C. Laprie, B. Randell, and C. Landwehr. Basic concepts and taxonomy of dependable and secure computing. *IEEE Transactions on Dependable and Secure Computing*, 1(1):11–33, Jan 2004.

[23] N. Leveson. Software system safety, Lecture Notes, Massachusetts Institute of Technology (MIT), Aero/Astro Department, 2002.

[24] M. Rausand. *Reliability of Safety-Critical Systems: Theory and Applications*. John Wiley and Sons, Inc., Hoboken, New Jersey, 2013.

[25] R.J. Mikulak, R. McDermott, and M. Beauregard. *The Basics of FMEA, 2nd Edition*. CRC Press, 2008.

[26] M. Rausand and A. Høyland. *System Reliability Theory: Models, Statistical Methods and Applications*. Wiley-Interscience, NJ, 2004.

[27] SAE ARP4761 Guidelines and Methods for Conducting the Safety Assessment Process on Civil Airborne Systems and Equipment, December 1996.

[28] M. Stamatelatos, W. Vesely, J. Dugan, J. Fragola, J Minarick, and J. Railsback. Fault Tree Handbook with Aerospace Applications. Handbook, National Aeronautics and Space Administration, 2002.

[29] H. E. Lambert. Use of fault tree analysis for automotive reliability and safety analysis. SAE technical paper, Lawrence Livermore National Lab., 2004.

[30] US Department of Defense. *Military Handbook: Electronic Reliability Design Handbook (MIL-HDBK-338B)*, October 1998.

[31] IEC 61508:2010. Functional Safety of Electrical/Electronic/Programmable Electronic Safety-Related Systems, Part 1-7, International Electrotechnical Commission, 2010.

[32] *Component Reliability Data for Use in Probabilistic Safety Assessment (IAEA-TECDOC-478).* International Atomic Energy Agency, Vienna, 1988.

[33] M. Generowicz and A. Hertel. Reassessing failure rates. Technical report, I&E Systems Pty Ltd, 2017.

[34] International Society of Automation. *ISA-TR84.02-2002: Safety Instrumented Functions — Safety Integrity Level Evaluation Techniques*, North Carolina, 2002.

[35] M. Rausand. *Reliability of safety-critical systems: theory and applications.* John Wiley & Sons, 2014.

[36] A. Saltelli. *Global sensitivity analysis: the primer.* John Wiley, 2008.

[37] L. Breierova and M. Choudhari. An introduction to sensitivity analysis. Technical report, Massachusetts Institute of Technology (MIT), 1996.

[38] A.C. Cullen and H.C. Frey. *Probabilistic techniques in Exposure assessment.* Plenum Press, New York, 1999.

[39] D. J. Pannell. Sensitivity analysis of normative economic models: theoretical framework and practical strategies. *Agricultural Economics*, 16(2):139 – 152, 1997.

[40] R. D. Hawkins and T. P. Kelly. Software safety assurance - what is sufficient? In *4th IET International Conference on Systems Safety 2009. Incorporating the SaRS Annual Conference*, pages 1–6, Oct 2009.

[41] *Nuclear Installations Act 1965, section 14.* Her Majesty's Stationary Office, London, UK, 1965 (reprinted 1993).

[42] K. Cassidy. CIMAH Safety Cases — the HSE Approach. IChemE Symposium series no. 110, 1988.

[43] Anthony Hidden (QC). *Investigation into the Clapham Junction Railway Accident.* HMSO, 1989.

[44] R B. Whittingham. *Preventing corporate accidents : An Ethical Approach.* Elsevier Ltd, 2008.

[45] Goal Structuring Notation working group. GSN Community Standard Version 1. Origin Consulting Limited, York, UK, November 2011.

[46] Object Management Group (OMG). *Structured Assurance Case Metamodel (SACM)*, Technical report, Version 1.0, 2013. [Online]. Available: `http://www.omg.org/spec/SACM/1.0/PDF/`.

[47] P. Graydon. (personal communication), August 2015.

[48] Jacobs Sverdrup Australia Pty, Ltd. The development of safety cases for complex safety critical systems. lecture notes. April 2005. [online]. avaialble: `https://msquair.files.wordpress.com/2012/06/md12_safety_cases_r5.pdf`.

[49] T. Kelly. Introduction to safety cases, lecture notes, 2007. [online]. available: `http://www.omg.org/news/meetings/workshops/SWA_2007_Presentations/00-T3_Kelly.pdf`.

[50] U.K. Ministry of Defence. *00-56 Defence Standard — Issue 6, 2015. Safety Management Requirements for Defence Systems — Part 1: Requirements and Guidance. (UK) Ministry of Defence*, June 2015.

[51] T. Kelly. A systematic approach to safety case management. In *Proceedings of SAE 2004 World Congress, Detroit*. The Society for Automotive Engineers, March 2004.

[52] O. Jaradat, I. Bate and S. Punnekkat. Facilitating the maintenance of safety cases. In *Proceedings of the 3rd International Conference on Reliability, Safety and Hazard - Advances in Reliability, Maintenance and Safety (ICRESH-ARMS)*, Luleå, Sweden, June 2015.

[53] O. Jaradat, P. Graydon and I. Bate. The role of architectural model checking in conducting preliminary safety assessment. In *Proceedings of the 31st International System Safety Conference (ISSC)*, Boston, USA, August 2013.

[54] P. J. Graydon and C. M. Holloway. "Evidence" under a magnifying glass: Thoughts on safety argument epistemology. In *10th IET System Safety and Cyber-Security Conference 2015*, pages 1–6, Oct 2015.

[55] J. Guiochet, Q. A. D. Hoang and M. Kaaniche. A model for safety case confidence assessment. In *Proceedings of the 34th International Conference on Computer Safety, Reliability, and Security - Volume 9337*, SAFECOMP 2015, pages 313–327. Springer-Verlag New York, Inc., 2015.

[56] B. Littlewood and D. Wright. The use of multilegged arguments to increase confidence in safety claims for software-based systems: A study based on a bbn analysis of an idealized example. *IEEE Transactions on Software Engineering*, 33(5):347–365, May 2007.

[57] X. Zhao, D. Zhang, M. Lu and F. Zeng. *A New Approach to Assessment of Confidence in Assurance Cases*, pages 79–91. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.

[58] E. Denney, G. Pai, and I. Habli. Towards measurement of confidence in safety cases. In *2011 International Symposium on Empirical Software Engineering and Measurement*, pages 380–383, Sept 2011.

[59] R. Hawkins, T. Kelly, J. Knight and P. J. Graydon. *A New Approach to creating Clear Safety Arguments*, pages 3–23. Springer London, UK, 2011.

[60] E. Denney, G. Pai, and I. Habli. Dynamic safety cases for through-life safety assurance. In *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering*, volume 2, pages 587–590, May 2015.

[61] B. Meyer. Design by contract. Technical Report TR-EI-12/CO, Interactive Software Engineering Inc., 1986.

[62] B. Meyer. *Object-Oriented Software Construction*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1st edition, 1988.

[63] C. A. R. Hoare. An axiomatic basis for computer programming. *Commun. ACM*, 12(10):576–580, October 1969.

[64] A. Benveniste, B. Caillaud, A. Ferrari, L. Mangeruca, R. Passerone, and C. Sofronis. Multiple viewpoint contract-based specification and design. In *Proceedings of the 6th International Symposium, FMCO*, pages 200–225, Amsterdam, The Netherlands, October 2007. Springer.

[65] W. Damm, H. Hungar, J. Bernhard, T. Peikenkamp, and I. Stierand. Using contract-based component specifications for virtual integration testing and architecture design. In *Proceedings of the Design, Automation & Test in Europe Conference & Exhibition*, pages 1–6, 2011.

[66] S. Bauer, A. David, R. Hennicker, K. G. Larsen, A. Legay, U. Nyman, and A. Wasowski. Moving from specifications to contracts in component-based design. In *Proceedings of the 15th International Conference on*

Fundamental Approaches to Software Engineering, FASE'12, pages 43–58, Berlin, Heidelberg, 2012. Springer-Verlag.

[67] ISO/IEC/IEEE International Standard for Software Engineering - Software Life Cycle Processes - Maintenance, Sept 2006.

[68] ISO/IEC 25010:2011 - Systems and software engineering - Systems and software Quality Requirements and Evaluation (SQuaRE) - System and software quality models, 2011.

[69] The Offshore Installations (Offshore Safety Directive)(Safety Case etc) Regulations. Health and Safety Executive (HSE), UK, 2015.

[70] T. Kelly. Literature survey for work on evolvable safety cases. Department of Computer Sceince, University of York, 1st Year Qualifying Dissertation, 1995.

[71] S. Wilson, T. Kelly, and J. McDermid. Safety case development: Current practice, future prospects. In *Proceedings of the 12th Annual CSR Workshop - Software Bases Systems*. Springer-Verlag, 1997.

[72] Industrial Avionics Working Group (IAWG). Modular Software Safety Case (MSSC): Process Overview, November 2012.

[73] M. Nicholson, I. Bate and J. Mcdermid. Generating and maintaining a safety argument for integrated modular systems. In *Preceedings of the 5th Australian Workshop on Safety Critical Systems and Software, Adelard for the Health and Safety Executive, HSE Books, ISBN 0-7176-2010-7, and Contract Research*, pages 0–7176. Institution of Engineers Australia, 2000.

[74] S. Björnander, R. Land, P. Graydon, K. Lundqvist, and P. Conmy. A method to formally evaluate safety case evidences against a system architecture model. In *Preceedings of the 23rd IEEE International Symposium on Software Reliability Engineering Workshops*, pages 337–342, Nov 2012.

[75] S. Kokaly, R. Salay, M. Chechik, M. Lawford and T. Maibaum. Safety case impact assessment in automotive software systems: An improved model-based approach. In *Preceedings of the 36th Computer Safety, Reliability, and Security (SAFECOMP)*, pages 69–85. Springer International Publishing, 2017.

[76] I. Crnkovic, U. Asklund and A P. Dahlqvist. *Implementing and Integrating Product Data Management and Software Configuration Management*. Artech House, Inc., Norwood, MA, USA, 2003.

[77] J. Dick. Design traceability. *IEEE Software*, 22(6):14–16, November 2005.

[78] I. Sommerville and P. Sawyer. *Requirements Engineering: A Good Practice Guide*. John Wiley & Sons, Inc., New York, NY, USA, 1st edition, 1997.

[79] S. Robertson and J. Robertson. *Mastering the Requirements Process (2Nd Edition)*. Addison-Wesley Professional, 2006.

[80] S A. Bohner and R S. Arnold. *Software change impact analysis*. Los Alamitos, Calif. : IEEE Computer Society Press, 1996. Includes bibliographical references (p. 361-374).

[81] A. De Lucia, F. Fasano, and R. Oliveto. Traceability management for impact analysis. In *Preceedings of the 24th IEEE International Conference on Software Maintenance*, pages 21–30, Beijing, China, 2008.

[82] M. Oertel. *A linear scaling change impact analysis based on a formal safety model for automotive embedded systems*. PhD thesis, University of Oldenburg, 2016.

[83] S. A. Bohner. Software change impacts: an evolving perspective. In *Preceedings of the 18th International Conference on Software Maintenance*, pages 263–272. IEEE Computer Society, Canada, 2002.

[84] S. Nair, J. Luis de la Vara, M. Sabetzadeh and D. Falessi. Evidence management for compliance of critical systems with safety standards: A survey on the state of practice. *Information and Software Technology*, 60:1 – 15, 2015.

[85] H. Espinoza, A. Ruiz, M. Sabetzadeh, and P. Panaroni. Challenges for an open and evolutionary approach to safety assurance and certification of safety-critical systems. In *Proceedings of the 1st International Workshop on Software Certification (WoSoCER)*, pages 1–6, Nov 2011.

[86] N. Tracey, A. Stephenson, J. Clark and J. McDermid. A safe change oriented process for safety-critical systems. In *Proceedings of the International Workshop on Software Change Evolution*, 1999.

[87] M. Bozzano, A. Cimatti, C. Mattarei and S. Tonetta. Formal safety assessment via contract-based design. In Franck Cassez and Jean-François Raskin, editors, *Automated Technology for Verification and Analysis*, pages 81–97. Springer International Publishing, 2014.

[88] W. P. Stevens, G. J. Myers, and L. L. Constantine. Structured design. *IBM Syst. J.*, 13(2):115–139, June 1974.

[89] C. Terwiesch and C H. Loch. Managing the process of engineering change orders: the case of the climate control system in automobile development. *Journal of Product Innovation Management*, 16(2):160 – 172, 1999.

[90] E. Fricke, B. Gebhard, H. Negele and E. Igenbergs. Coping with changes: Causes, findings, and strategies. *Systems Engineering*, 3(4):169–179, 2000.

[91] S. Rajasekar, P. Philominathan, V. Chinnathambi. Research methodology. October 2013. [Online]. Available: http://arxiv.org/pdf/physics/0601009v3.pdf [Lastchecked]: October 2015.

[92] A. Salameh and O. Jaradat. A safety-centric change management framework by tailoring agile and V-model processes. In *Proceedings of the 36th International System Safety Conference (ISSC), Arizona, USA*, August 2018.

[93] AUTomotive Open System ARchitecture (AUTOSAR). www.autosar.org/ [Last checked] November 2018.

[94] D. Durisic, M. Staron and M. Tichy. ARCA – Automated Analysis of AUTOSAR Meta-model Changes. In *Preceedings of the 7th International Workshop on Modeling in Software Engineering*, pages 30–35, May 2015.

[95] O. Jaradat, I. Sljivo, I. Habli and R. Hawkins. Challenges of safety assurance for industry 4.0. In *European Dependable Computing Conference (EDCC)*. IEEE Computer Society, September 2017.

[96] S. Girs, I. Sljivo, and O. Jaradat. Contract-based assurance for wireless cooperative functions of vehicular systems. In *Preceedings of the 43rd Annual Conference of the IEEE Industrial Electronics Society IECON*, pages 8391–8396, Oct 2017.