Nuno Pedro de Jesus Silva

An Empirical Approach to Improve the Quality and Dependability of Critical Systems Engineering

PhD Thesis in Doctoral Program in Information Science and Technology,
supervised by Professor Marco Vieira
and presented to the *Department* of *Informatics Engineering*
of the Faculty of Sciences and Technology
of the University of Coimbra

August 2017

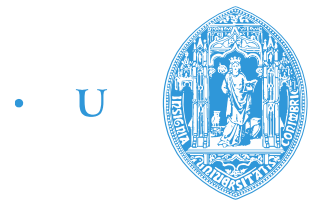
· U C ·

UNIVERSIDADE DE COIMBRA

An Empirical Approach to Improve the Quality and Dependability of Critical Systems Engineering

Nuno Pedro de Jesus Silva

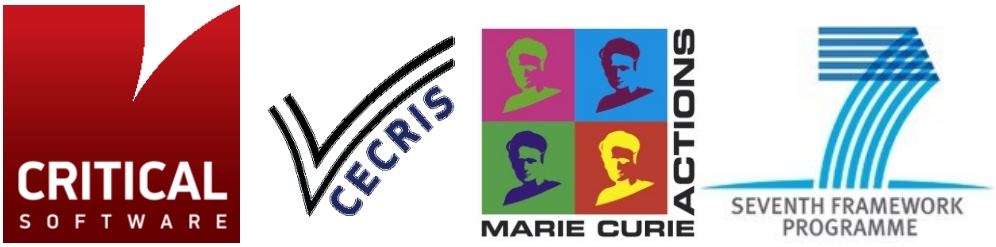
Thesis submitted to the University of Coimbra
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy
August 2017



Department of Informatics Engineering
Faculty of Sciences and Technology
University of Coimbra

This research has been developed as part of the requirements of the Doctoral Program in Information Science and Technology of the Faculty of Sciences and Technology of the University of Coimbra. This work is within the Dependable Systems specialization domain and was carried out in the Software and Systems Engineering Group of the Center for Informatics and Systems of the University of Coimbra (CISUC).

This work was supported financially by the Top-Knowledge program of CRITICAL Software, S.A. and carried out partially in the frame of the European Marie Curie Project FP7-2012-324334-CECRIS (Certification of CRItical Systems).



This work has been supervised by **Professor Marco Vieira**, Full Professor (*Professor Catedrático*), Department of Informatics Engineering, Faculty of Sciences and Technology, University of Coimbra.

*"**Safety** is not an option!"*
James H. Miller

Miller, James H. "2009 CEOs Who "Get It"" Interview. Safety and Health Magazine." February 1, 2009. Accessed October 13, 2016. http://www.safetyandhealthmagazine.com/articles/2009-ceos-who-get-it-16.

and

*"**Failure** is not an option"*
Bill Broyles, attributed to Apollo 13 crew of NASA

To my family,
to my friends

Abstract

Critical systems, such as space, railways and avionics systems, are developed under strict requirements envisaging high integrity in accordance to specific standards. For such software systems, generally an independent assessment is put into effect (as a safety assessment or in the form of Independent Software Verification and Validation - ISVV) after the regular development lifecycle and V&V activities, aiming at identifying and correcting residual faults and raising confidence in the software. These systems are very sensitive to failures (they might cause severe impacts), and even if they are today reaching very low failure rates, there is always a need to guarantee higher quality and dependability levels. However, it has been observed that there are still a significant number of defects remaining at the latest lifecycle phases, questioning the effectiveness of the previous engineering processes and V&V techniques.

This thesis proposes an empirical approach to identify the nature of defects (quality, dependability, safety gaps) and, based on that knowledge, to provide support to improve critical systems engineering. The work is based on knowledge about safety critical systems and how they are specified/developed/validated (standards, processes and techniques, resources, lifecycles, technologies, etc.). Improvements are obtained from an orthogonal classification and further analysis of issues collected from real systems at all lifecycle phases. Such historical data (issues) have been studied, classified and clustered according to different properties and taking into account the issue introduction phase, the involved techniques, the applicable standards, and particularly the root causes. The identified improvements shall be reflected in the development and V&V techniques, on resources training or preparation, and drive standards modifications or adoption.

The first and more encompassing contribution of this work is the **definition of a defects assessment process** that can be used and applied in industry in a simple way and independently from the industrial domain. The process makes use of a dataset collected from existing issues reflecting process deficiencies, and supports the analysis of these data towards identifying the root causes for those problems and defining appropriate measures to avoid them in future systems.

As part of the defect assessment process activities, we propose an **adaptation of the Orthogonal Defect Classification (ODC) for critical issues**. In practice, ODC was used as an initial classification and then it was tuned according to the gaps and difficulties found during the initial stages of our defects classification activities. The refinement was applied on the defect types, triggers and impacts. Improved taxonomies for these three parameters are proposed.

A subsequent contribution of our work is the **application and integration of a root cause analysis process** to show the connection of the defects (or issue groups) with the

engineering properties and environment. The engineering properties (e.g. human and technical resources properties, events, processes, methods, tools and standards) are, in fact, the principal input for the classes of root causes. A fishbone root cause analysis was proposed, integrated in the process and applied to the available dataset.

A practical contribution of the work comprises the **identification of a specific set of root causes and applicable measures to improve the quality of the engineered systems** (removal of those causes). These root causes and proposed measures allow the provision of quick and specific feedback to the industrial engineering teams as soon as the defects are analyzed. The list/database has been compiled from the dataset and includes the feedback and contributions from the experts that responded to a process/framework validation survey. The root causes and the associated measures represent a valuable body of knowledge to support future defects assessments.

The last key contribution of our work is the **promotion of a cultural change** to appropriately make use of real defects data (the main input of the process), which shall be appropriately documented and easily collected, cleaned and updated. The regular use of defects data with the application of the proposed defects assessment process will contribute to measure the quality evolutions and the progress of implementation of the corrective actions or improvement measures that are the essential output of the process.

Keywords:

Orthogonal defect classification, critical systems, defect, classification; root cause analysis, dependability, failure, safety.

Resumo

Os sistemas críticos, tais como os sistemas espaciais, ferroviários ou os sistemas de aviónica, são desenvolvidos sob requisitos estritos que visam atingir alta integridade ao abrigo de normas específicas. Para tais sistemas de software, é geralmente aplicada uma avaliação independente (como uma avaliação de *safety* ou na forma de uma Verificação e Validação de Software Independente - ISVV) após o ciclo de desenvolvimento e as respetivas atividades de V&V, visando identificar e corrigir falhas residuais e aumentar a confiança no software. Estes sistemas são muito sensíveis a falhas (pois estas podem causar impactos severos), e apesar de atualmente se conseguir atingir taxas de falhas muito baixas, há sempre a necessidade de garantir a maior qualidade dos sistemas e os maiores níveis de confiabilidade. No entanto, observa-se que ainda existe um número significativo de defeitos que permanecem nas últimas fases do ciclo de desenvolvimento, o que nos leva a questionar a eficácia dos processos de engenharia usados e as técnicas de V&V aplicadas.

Esta tese propõe uma abordagem empírica para identificar a natureza dos defeitos (de qualidade, confiabilidade, lacunas de *safety*) e com base nesse conhecimento proporcionar uma melhoria da engenharia de sistemas críticos. O trabalho é baseado em conhecimento sobre os sistemas críticos e na forma como estes são especificados / desenvolvidos / validados (normas, processos e técnicas, recursos, ciclo de vida, tecnologias, etc.). As recomendações de melhorias para os sistemas críticos são obtidas a partir de uma classificação ortogonal e posterior análise de dados de defeitos obtidos de sistemas reais cobrindo todas as fases do ciclo de vida. Estes dados históricos (defeitos) foram estudados, classificados e agrupados de acordo com diferentes propriedades, considerando a fase de introdução do defeito, as técnicas envolvidas, as normas aplicáveis e, em particular, as possíveis causas fundamentais (ou raiz). As melhorias identificadas deverão refletir-se nas técnicas de desenvolvimento / V&V, na formação ou preparação de recursos humanos e orientar alterações ou adoção de normas.

A primeira e mais abrangente das contribuições deste trabalho é a **definição de um processo de avaliação de defeitos** que pode ser usado e aplicado na indústria de forma simples e independente do domínio industrial. O processo proposto baseia-se na disponibilidade de um conjunto de dados de problemas que refletem deficiências de processo de desenvolvimento e suporta a análise desses dados para identificar as suas causas raiz e definir medidas apropriadas para evitá-los em sistemas futuros.

Como parte das atividades do processo de avaliação de defeitos, é proposta uma **adaptação da Classificação Ortogonal de Defeitos (ODC) para sistemas críticos**. Na prática, a ODC foi usada como uma classificação inicial e depois ajustada de acordo com as lacunas e dificuldades encontradas durante os estágios iniciais das atividades de classificação de defeitos. O refinamento foi aplicado aos tipos de defeito, aos

eventos que levaram a esses defeitos e aos seus impactos. Neste trabalho, são propostas versões melhoradas das taxonomias para esses três parâmetros.

Uma contribuição subsequente é a **aplicação e integração de um processo de análise de causas raiz** para relacionar os defeitos (ou grupos de problemas) com as propriedades e o ambiente de engenharia. As propriedades de engenharia (por exemplo, recursos humanos e técnicos, eventos, processos, métodos, ferramentas e normas) são, de facto, as principais fontes para a identificação das classes de causas raiz. A análise de causas de raiz proposta é baseada em diagramas *fishbone*, tendo sido integrada no processo e aplicada ao conjunto de dados disponíveis.

Uma contribuição prática do nosso trabalho é a **identificação de um conjunto específico de causas raiz e de medidas aplicáveis para melhorar a qualidade dos sistemas de engenharia** (eliminação dessas causas). As causas e as medidas propostas permitem um retorno rápido e específico logo que os defeitos são analisados. A lista / base de dados foi compilada a partir do conjunto de dados de defeitos e inclui os comentários e contribuições de especialistas que responderam a um formulário de validação do processo. As causas raiz e as medidas associadas representam um conjunto valioso de conhecimento que pode suportar futuras análises de defeitos.

A última contribuição chave do nosso trabalho é a **promoção de uma mudança cultural** para fazer uso apropriado de dados de defeitos reais (principal fonte do processo), os quais devem ser devidamente documentados e facilmente recolhidos, tratados e atualizados. O uso regular de dados sobre defeitos através da aplicação do processo de análise de defeitos proposto contribuirá para medir a evolução da qualidade e o progresso da implementação das ações corretivas ou medidas de melhoria que são o principal resultado do processo.

Palavras-chave:
Classificação ortogonal de defeitos, sistemas críticos, defeito, classificação, análise de causas, confiabilidade, falha, safety.

Acknowledgements

I will never be able to thank enough and everybody who somehow supported me during these years. These acknowledgements will be necessarily short and thus not complete. I would like to thank everybody who crossed my path during these last years and that gave me motivation to work during the different phases of the studies and the research. I am truly grateful and will never be able to pay back all the support received.

I would like to start by thanking Professor Marco Vieira for his guidance through the entire path that led me to this point. We had a very hard time in meeting since we both work full time, we travel a lot, so time was always short and precious, but Marco never ceased believing, supporting, motivating and providing very detailed reviews to improve and advance the research work.

I would also like to give two special thanks for professor Henrique Madeira and Professor João Carlos Cunha, the first for inspiring me and motivating me in entering this adventure, many years ago when we did a visit to NASA IVV center in West Virginia, and the second for his support and insightful contributions and reviews to the improvement of my work.

I would also like to give a special thanks to all the PhD professors and classes colleagues. To the researchers at DEI/CISUC, that have not seen much of me, but that were always there ready for anything.

An enormous gratitude goes to my employer, CRITICAL Software, for supporting my studies and giving me some time to achieve the objectives, particularly during the first year and over the periods just before the articles submissions. Thank You CRITICAL.

I would like to thank also my co-workers for their interested and their support to the research topics and for their patience over the last years. I have also a few ex-co-workers to thanks, especially Rui (UK) and Diogo (Germany) for their prompt and insightful help.

Another special thanks goes to all the CECRIS project partners. This European Commission FP7 research project allowed me to extend and develop my research and to meet very knowledgeable researchers. I cannot name all here, but special thanks need to go to Domenico, Stefano, Christian, Marcello, Roberto P., Roberto N., Domenico (DiLeo), Antonio (La Legenda), Fabio, Dario, Anna, Alma, Francesca, and more at CINI Napoli; then Dr. Pataricza, László, Ágnes, Ákos, Imre, Zoltán, Ábel and Gábor from BME in Budapest; also to Dr. Bondavalli, Andrea (Ceccarelli), Nicola, Francesco and others from CINI Firenze and Resiltech in Pontedera.

I also need to thank Ram Chillarege for his support during 2016 with all his knowledge about ODC and with all his contagious energy.

The most important thanks, however, needs to go to my wife, Daniela. For her unconditional love and support, for her indefatigable patience and understanding, for all the long moments where we could not be together because I was working or traveling, for being her and always supporting and motivating me, for always believing in me, especially in the difficult times occurred over the past years.

I need to also thank my parents, Fernando and Fernanda, who were far away most of the time, one ocean distant, but who recently became much closer. For the time that I could not dedicate to them over these years and for giving me the grounds to achieve this milestone in my life, I owe them a lot.

Likewise, I want to thank my sisters Cintia and Suzi, one in Portugal, the other in Canada, my brother-in-law Alcino, my nephews Laura and Afonso, particularly for their energy, for making me be a "crazy children" on weekends, for the smiles and the plays.

Another thanks need to go to my in-laws, Jaime and Maria Isabel, and all the family back in Angola (and not only) for their understanding of my absences in these busy years.

To the rest of my family, especially aunts, uncles and cousins, I would like to thank all of them for the great moments spent together over the past years, and I hope to have now the chance to be more present.

I also thank all the anonymous reviewers that helped me to improve this work with their comments, and the conference contacts that provided feedback and engaged in very fruitful conversations.

A particular "thank you" to my other friends, spread around this world. I need to give them a heads up: I will have more time to you from now on, so, all those in Brazil, Canada or Europe, beware, we will surely get together more often, particularly Pedro and Sandra (in Montreal), all my cousins and family in São Paulo (you are great), and Lubomir and his 3 "girls" (currently in Switzerland).

I would like to give a special thanks to Agnė, my co-worker, my gym partner, who became a good friend over time listening to my stories (too many), having a renewable patience and providing me courage and positive energy, support and many inspiring smiles to motivate me in the final phase of the process. Ačiū Agnė!

To all the people I met during these last 4 years in Coimbra, in Portugal, and around the world, hoping to see most of them again soon. I would like to thank all because without them I would not be me, and I would not have met this milestone.

I could not forget to mention one of my passions, *Associação Académica de Coimbra*, for all the emotions provided at the football matches, we went from Europa League down to Second League relegations, but the passion has only grown, and that proved that we do not explain love, we just love. Académica will continue being special and different. Força Briosa!

To the angels and those who left in the meantime (2013-2017), you will always be in my heart, thank you for being part of my life and part of my success.

Agradecimentos

Nunca seria possível agradecer o suficiente e a todos os que de alguma forma me apoiaram durante estes anos. Estes agradecimentos serão necessariamente curtos e, portanto, não completos. Gostaria de agradecer a todos os que atravessaram o meu caminho durante estes últimos anos e que me deram motivação adicional para continuar e trabalhar durante as diferentes fases dos estudos e da investigação. Estou realmente grato e nunca conseguirei retribuir todo o apoio recebido.

Gostaria de começar por agradecer ao Professor Marco Vieira pela sua orientação durante todo o período que me trouxe a este ponto. Foi complicado por vezes reunir, estado ambos sempre ocupados profissionalmente, ou em viagem, sendo que o tempo era sempre pouco e precioso, mas o Marco nunca deixou de acreditar, apoiar, motivar e fornecer revisões muito detalhadas para melhorar e avançar o trabalho de pesquisa.

Gostaria de agradecer, em particular, ao professor Henrique Madeira e ao professor João Carlos Cunha, o primeiro por me ter inspirado e motivado a abraçar esta aventura, há muitos anos, quando fizemos uma visita ao centro IVV da NASA, o segundo pelo seu apoio e contribuições técnicas e revisões para a melhoria do meu trabalho.

Gostaria também de agradecer especialmente a todos os professores e colegas do programa de doutoramento. Agradeço igualmente aos pesquisadores do DEI / CISUC, que não viram muito de mim, mas que estavam sempre prontos para qualquer eventualidade.

Tenho uma enorme dívida de gratidão para o meu empregador, a CRITICAL Software, por apoiar os meus estudos e dar-me algum tempo para atingir os objectivos, particularmente durante o primeiro ano e durante os períodos imediatamente antes das submissões de artigos. Obrigado CRITICAL.

Gostaria também de agradecer aos meus colegas de trabalho pelo interesse e apoio relativo aos temas de investigação e pela sua paciência nos últimos anos. Existem igualmente alguns ex-colegas de trabalho aos quais devo agradecer, especialmente o Rui (UK) e o Diogo (Alemanha) pela sua ajuda pronta e eficaz.

Outro agradecimento especial vai para todos os parceiros do projeto CECRIS. Este projecto de investigação da Comissão Europeia permitiu alargar e desenvolver a minha investigação. Não posso citar todos os nomes aqui, mas envio agradecimentos especiais para Domenico, Stefano, Christian, Marcello, Roberto P., Roberto N., Domenico (DiLeo), Antonio (La Legenda), Fabio, Dario, Anna, Alma, Francesca, e outros mais do CINI em Nápoles; o Dr. Pataricza, László, Ágnes, Ákos, Imre, Zoltán, Ábel e Gábor da BME em Budapeste; também para o Dr. Bondavalli, Andrea (Ceccarelli), Nicola, Francesco e outros do CINI em Florença e da Resiltech em Pontedera.

Também devo agradecer ao Ram Chillarege pelo seu apoio durante o ano de 2016 com todo seu conhecimento sobre ODC e com toda sua energia contagiosa.

Os agradecimentos mais importantes vão, no entanto, para a minha esposa, Daniela. Pelo seu amor e apoio incondicionais, pelas suas infatigáveis paciência e compreensão, por todos os longos momentos em que não pudemos estar juntos porque eu estava a trabalhar ou a viajar, por ser ela e sempre me apoiar e motivar, por sempre acreditar em mim, especialmente nos tempos difíceis dos últimos anos.

Preciso de agradecer aos meus pais, Fernando e Fernanda, que estavam longe a maior parte do tempo, a um oceano de distância, mas que recentemente se aproximaram, retornando a Portugal. Pelo tempo que não lhes pude dedicar ao longo destes anos e por me terem dado as bases para atingir este marco na minha vida, eu devo-lhes imenso.

Da mesma forma, quero agradecer às minhas irmãs Cintia e Suzi, uma em Portugal, a outra no Canadá, ao meu cunhado Alcino, aos meus sobrinhos Laura e Afonso, particularmente pela sua energia, por me fazerem ser uma "criança louca" nos fins de semana, pelos sorrisos e as brincadeiras.

Outro agradecimento vai para os meus sogros, Jaime e Maria Isabel, e toda a família em Angola (e não só) pela compreensão das minhas ausências nestes anos.

Ao resto da minha família, especialmente tias, tios e primos, gostaria de agradecer a todos eles pelos grandes momentos que partilhámos nos últimos anos, e espero ter agora a oportunidade de estar mais presente.

Agradeço também a todos os revisores anónimos por terem contribuído para melhorar o meu trabalho com os seus comentários, e também aos contatos dos seminários internacionais que forneceram feedback e participaram em conversas muito frutíferas.

Um especial "obrigado" aos meus outros amigos, espalhados por este mundo fora. Aviso: vou ter mais tempo para vocês a partir de agora, portanto, todos os que estão no Brasil, Canadá ou na Europa, vamos certamente encontrar-nos mais vezes, particularmente o Pedro e a Sandra (Montreal), todos os meus primos e restante família em São Paulo (vocês são especiais), e o Lubomir e as suas 3 "meninas" (na Suíça).

Gostaria de agradecer especialmente à Agnė, minha colega de trabalho, minha parceira de ginásio, que se tornou uma boa amiga ao longo do tempo por ouvir as minhas histórias (demasiadas), ter uma paciência renovável e dar-me energia positiva e muitos sorrisos inspiradores para me motivar na fase final deste processo. Ačiū Agnė!

A todas as pessoas que conheci nestes últimos 4 anos em Coimbra, em Portugal e em todo o mundo. Gostaria de agradecer a todos porque sem eles eu não seria eu.

Não posso deixar de mencionar uma das minhas paixões, a Associação Académica de Coimbra, por todas as emoções proporcionadas nos jogos de futebol, fomos da Liga Europa até à descida de divisão para a Segunda Liga, mas a paixão só cresceu e isso provou que não se explica o amor, apenas se ama. A Académica continuará a ser especial e diferente. Força Briosa!

Para os anjos e aqueles que nos deixaram entretanto (2013-2017), vocês estarão sempre no meu coração, obrigado por fazerem parte da minha vida e parte do meu sucesso.

List of Publications

This thesis relies on the published scientific research presented in the following peer reviewed papers:

[1] Nuno Silva and Marco Vieira. “Towards Making Safety-Critical Systems Safer: Learning from Mistakes”, ISSRE2014, 3-6- November 2014, Naples, Italy.

[2] Nuno Silva, Marco Vieira, “Experience Report: Orthogonal Classification of Safety Critical Issues”, ISSRE2014, 3-6- November 2014, Naples, Italy.

[3] Nuno Silva, Marco Vieira, Dario Ricci, Domenico Cotroneo, “Consolidated View on Space Software Engineering Problems – An empirical study”, DASIA 2015, 19-21 May, 2015, Barcelona, Spain.

[4] Nuno Silva, Marco Vieira, Dario Ricci, and Domenico Cotroneo. "Assessment of Defect Type influence in Complex and Integrated Space Systems: Analysis Based on ODC and ISVV Issues." In Dependable Systems and Networks Workshops (DSN-W), 2015 IEEE International Conference on, pp. 63-68. IEEE, 2015.

[5] Nuno Silva and Marco Vieira, “Software for Embedded Systems: A Quality Assessment based on improved ODC taxonomy”, SAC ACM 2016, April 04-08, 2016, Pisa, Italy, DOI: http://dx.doi.org/10.1145/2851613.2851908.

[6] Nuno Silva, João Carlos Cunha, Marco Vieira, A field study on root cause analysis of defects in space software, Reliability Engineering & System Safety, Available online 24 August 2016, ISSN 0951-8320, http://dx.doi.org/10.1016/j.ress.2016.08.016.

[7] Nuno Silva and Marco Vieira. 2016. “Adapting the Orthogonal Defect Classification Taxonomy to the Space Domain.” In Computer Safety, Reliability, and Security: 35th International Conference, SAFECOMP 2016, Trondheim, Norway, September 21-23, 2016, Proceedings, edited by Amund Skavhaug, Jérémie Guiochet, and Friedemann Bitsch, 296–308. Cham: Springer International Publishing. http://dx.doi.org/10.1007/978-3-319-45477-1_23.

[8] Nuno Silva, Marco Vieira, João Cunha and Ram Chillarege, “Evaluating a Corpus of Root Causes and Measures to guide RCA processes in Critical Software”, 2017 IEEE 18th International Symposium on High Assurance Systems Engineering, HASE 2017, January 12th-14th, 2017, Singapore.

Other published works with authorship or contributions from the author during the period:

[9] Andrea Ceccarelli, Nuno Silva, “Qualitative comparison of aerospace standards: an objective approach”, WoSoCer 2013, ISSRE 2013, 4-7 November 2013, Pasadena, CA, USA

[10] Nuno Silva, Alexandre Esper, Ricardo Barbosa, Johan Zendin, Claudio Monteleone, “Reference Architecture for High Dependability On-Board Computers”, WoSoCer 2013, ISSRE 2013, 4-7 November 2013, Pasadena, CA, USA (Presented by myself)

[11] Nuno Silva, Marco Vieira, “Certification of Embedded Systems: Quantitative analysis and irrefutable evidences”, ISSRE 2013 – Fast Abstracts, 4-7 November 2013, Pasadena, CA, USA (Presented by myself)

[12] Sun, Linling, Nuno Silva, and Tim Kelly. 2014. “Rethinking of Strategy for Safety Argument Development.” In Computer Safety, Reliability, and Security: SAFECOMP 2014 Workshops: ASCoMS, DECSoS, DEVVARTS, ISSE, ReSA4CI, SASSUR. Florence, Italy, September 8-9, 2014. Proceedings, edited by Andrea Bondavalli, Andrea Ceccarelli, and Frank Ortmeier, 384–395. Cham: Springer International Publishing. http://dx.doi.org/10.1007/978-3-319-10557-4_42.

[13] Ceccarelli, Andrea, and Nuno Silva. 2015. “Analysis of Companies Gaps in the Application of Standards for Safety-Critical Software.” In Computer Safety, Reliability, and Security: SAFECOMP 2015 Workshops, ASSURE, DECSoS. ISSE, ReSA4CI, and SASSUR, Delft, The Netherlands, September 22, 2015, Proceedings, edited by Floor Koornneef and Coen van Gulijk, 303–313. Cham: Springer International Publishing. http://dx.doi.org/10.1007/978-3-319-24249-1_26.

Table of Contents

List of Figures

List of Tables

Chapter 1
Introduction

"If builders built buildings the way computer programmers write programs, the first woodpecker that came along would have destroyed all civilization" -- Gerald Weinberg

Software is becoming more and more ubiquitous and the importance and complexity of software systems in the safety critical domains are constantly increasing. A safety critical system is a system where a failure might result in the loss of human life, damage the environment or cause a severe incident/accident.

Safety critical systems, which strongly rely on software, are nowadays an essential component of all aerospace, automotive, railways, nuclear, defense and medical systems. However, in the past 30 years, there has been a significant number of software problems that caused accidents and failures with severe impact within safety-critical systems, e.g., Therac-25 [14], the Ariane 5 explosion [15], the Boeing 777-200 accident (registered 9M-MRG) [16] or the Boeing 787 Dreamliner integer overflow bug that could shut down the electrical power [17], and the Toyota Prius break problems [18] or Toyota's electronic throttle control system (ETCS) that had bugs that could cause sudden unintended acceleration [19], [20].

Safety and mission critical systems rely nowadays on more and more complex software and are difficult to control while guaranteeing the highest levels of quality and dependability. These systems must deal with the effects of faults and failures and, even with the maturity and advances of software engineering, it is not possible to create "perfect" systems nor software [21]. However, safety and mission critical industries have kept an impressive safety record (compared to the complexity and size growth) mostly due to the large effort spent on developing and validating their systems and to the application of mature international standards and strict guidelines [22] and heavy use of standard-based Verification and Validation (V&V) methodologies [23]. In fact, these systems are developed according to very strict rules and guidelines, mostly due to the need to be qualified and certified: for human safety it is not uncommon to require

the system to be designed to lose less than one life per billion (10^9) hours of operation, and consequently it needs to follow specific and strict development standards [22] that recommend or force techniques and processes, dedicated personnel training and extensive domain expertise.

When critical systems fail and accidents cannot be effectively avoided lives are in danger, extremely expensive systems are lost or damaged, and there are significant economic and negative company exposure impacts. In fact, data and lessons learned collected over several years of industrial experience have shown that safety critical systems engineering is not perfect, and relevant issues are still transferred from phase to phase [22], [24], [25]. Even in very strict engineering processes (such has the railway and aerospace domains) these issues have critical impacts, might mask other issues, and become costly to correct and maintain, while providing lower trust in the system. Ebert states that "*Applications that enter testing with an excessive volume of defects cannot exit the testing phase because they don't work*" [22].

Safety critical systems require stable requirements and have several inflexible requirements (constraints) to be fulfilled [26]. These systems are also known for the demanding integration and validation efforts (on average, 40% of the software engineering effort as per [27]), as they are not only generally embedded but also require evidences to guarantee high dependability levels.

The increased importance and complexity of safety-critical systems is imposing new objectives in terms of safety and dependability (quality), and at the same time revealing that the current engineering techniques and applicable standards are probably not enough to reach the safety levels required by society. In fact, these systems are still causing (too many) severe accidents and failures keep being propagated and introduced in all the lifecycle phases (for a concrete real example survey see [28]).

Informally, we can define **safety** as "*nothing bad will happen*". Leveson, in her book Safeware [14], defines safety as the "*freedom from accidents or losses*". Storey, in Safety-Critical Computer Systems [29], defines a safety-critical system as a system that "*will not endanger human life or the environment*". Despite all the possible similar and generic definitions of safety, in the frame of our work we consider safety as the "**freedom from the occurrence or risk of danger, injury or loss**". No system can be completely "safe", thus engineering needs to focus on making it safe enough, knowing that there are constraints like budget, time, and resources. Two strategies have been followed to achieve these goals in industry: *i)* focusing on eliminating end effects of accidents rather than risks [14]; and *ii)* focusing on removing hazards rather before the actual accidents [29].

This work addresses the problem of systematically studying the existing problems in safety critical projects, and analyzing them, by identifying the potential root causes and proposing solutions for avoiding their recurrence and, consequently, a negative impact on the system.

1.1 Engineering Safety Critical Systems

Processes for developing critical systems are usually based on the waterfall/V-model. The V-model defines a development process that is an extension of the waterfall model, but instead of having the phases moving down linearly, the V-model process steps start from the conceptual phase, moves to the requirements, architectural and implementation (coding) phases and then bends upwards to the testing phases and the operations and maintenance. In the V-model, there is not only traceability between subsequent phases, but also horizontal traceability along the V, between the testing phases and the development phases. This traditional model, which is normally connected to the applicable international standards, provides the basic phases of the engineering process, guides the structure of development and V&V, and is sequential (waterfall). The sequential nature of engineering of these systems is considered essential for managing communications, scale and complexity, integrating multidisciplinary teams and managing the integration by well-defined phases, and promoting traceability between the phase artefacts to facilitate certification (as required by the safety critical standards).

Safety critical systems are very sensitive to failures, and the ultimate goal is to avoid them at all costs, as "*failure is not an option*". Since empirical research helps in integrating research and practice, and empirical data and knowledge are essential to understand and respond adequately to the dynamics of engineering, to build upon what is already known and to act in order to adjust and correct situations that caused the issues, the importance of empirical data is undeniable.

Empirical data includes not only the knowledge of the development frameworks and processes, but also the actual data from failures, defects and identified issues, as well as the analysis of what led to these defects in the first place (be it a human, a process or a technology related root-cause). However, studying defects of safety critical systems seems to be different from non-critical systems, not only because the nature of the defects is different, but also due to the fact that the frequency of these defects is rather different, for example:

- **Industry average**: "about 15-50 errors per 1000 lines of delivered code" [14];
- **Microsoft applications**: "about 10-20 defects per 1000 lines of code during in-house testing, and 0.5 defect per KLOC in released product" [14];
- **Space**: as low as 3 defects per 1000 lines of code during in-house testing and 0.1 defect per 1000 lines of code in the released product [30];
- **Best code**: 0.5 to 1 defect per KLOC [31];
- **NASA**: down to 0.004 defects per KLOC, but a cost of 1000$/LOC compared to 25$/LOC for commercial code [31].

The general idea is that a fault density of 1 fault per thousand Lines of Code (KLOC), for safety critical systems is a world class value [32], and some studies do provide an insight on the real values. For example, the Space Shuttle software [33] reaches fault

densities lower than 0.1 per KLOC, but this is considered an exceptional result, and its development costs are higher than any other documented software development. In the aeronautics domain, the C130J software, developed according to DO178B [34], had a retrospective static analysis funded by the UK MoD (Ministry of Defense) that concluded the existence of about 1.4 safety critical faults per KLOC (the overall flaw density was about 23 per KLOC [35]). In commercial software, however, the fault density is commonly higher, usually up to 10 faults per KLOC, while pre-release fault densities can go up to 30 per KLOC [36], [37]. Other studies determine that the typical values for commercial software are up to 4 faults per KLOC, and confirm that the best fault densities possible for safety critical software are between 0.1-1.0 per KLOC (in-line with the values presented before) [38]. Lastly, in one of our studies [25], with data collected from 10 years of Independent Software Verification and Validation (ISVV), the fault density prior to ISVV for space systems is at least 0.97 defects per thousand lines of code. Although the fault density might not be stable when safety critical systems become large and complex, if we consider a stable rate of 1 per KLOC for a system with 100 KLOC, we are already talking about 100 safety critical faults.

In what concerns software failure rates, its estimation and collection is a bit harder than for software fault density. Ellims [39] has studied the failure rates in automotive industry, where most accidents are caused by driver action, and from those, the majority has mechanical causes. Based on 0.1% of the recalls due to software, Ellims has estimated that software issues (severe) cause a maximum of 5 deaths and 300 injuries annually in the UK. Taking into account the 5 million vehicles in the road and 300 hours of driving time per year, failure rate for software becomes 0.2×10^{-6} failures/hour (causing injury or death). Shooman [40] did a software fault analysis for the aviation industry where he reaches a value of 10^{-7} failures/hour, and McDermid [32] also calculated a value of 10^{-7} fatal accidents/hour for the aviation industry for software causes.

The safety critical industry relies on strict rules and application of **international standards**. An example of comparison of some of the most important standards has been performed with particular focus on Verification and Validation [23]: we have concluded that the basic contents and guidelines of these standards are common, and some industries provide only particular additions in what concerns the techniques and the way of presenting the evidences (with safety cases, or evidences format, for example). Several of these standards have, in fact, common roots, and reuse the lessons learned from the application of other standards in different domains.

The safety critical standards are generally mature, well established and most of the times updated regularly (although DO-178B [34] is from the 1990's, DO-178C [41] is already available and in use). However, they might be too generic and become outdated considering the new technologies, systems complexity and new software responsibilities (e.g. FPGA, ASIC, more intelligence, safety, reconfiguration, security, emergent behaviors, etc.).

For the safety critical industries, the application of standards is not optional, and all development must strictly follow them. Some industries have a "certification" body that needs to ensure that the full system is developed according to the requirements in

the standards. Other industries are less strict and do not require a formal certification by an authorized body, but still use the standards as mandatory guidelines for the development and acceptance of the systems. Examples of the first case are the aviation and the railway industries, which need to have all systems certified by the certification authorities before any market usage. In the second case, we can include the space industry where certification is not required but following the applicable standards is highly recommended and the systems are qualified before the acceptance phase by the International Space Agencies.

Examples of safety critical standards for different domains are (some of these have been used during this research, as we will see later): *Space Domain* - European Cooperation for Space Standardization (ECSS) series (e.g. [42] and [43]), and NASA Standards (e.g. [44]); *Airborne Domain* - DO-178B/C (software related) [34], [41], DO-254 (hardware related) [45], ARP-4761 [46], ARP-4754 [47]; *Automotive Domain* - ISO-26262 [48]; *Railway Domain* - CENELEC EN-50126 [49], EN-50128 [50], EN-50129 [51]; *Automation Domain* - IEC-61508 [52], IEC 61511 [53], [54], [55], IEC-62061 [56]; *Medical Domain* - IEC-62304 [57]; and *Nuclear Energy Domain* - IEC-60880 [58];

Some standards last much longer than technology (e.g. D0-178B, ARP), while others are not yet mature and widely accepted (ISO-26262). Practical experience and feedback from the ECSS working groups show how these standards evolve, and this evolution is not in a systematic and structured way (due to pressure from tool suppliers, influent companies forcing, technology preferences or experience influences/preferences, etc.). A recent case is DO-178C that has evolved the B version due to technology evolutions and trends (object oriented programming, use formal methods, improve the testing requirements, and industrial/commercial pressure).

In summary, engineering critical systems with a very low defect rates is a challenging objective. The existing technological domains follow different standards and processes, and are integrated in different development cultures, which leads to diverse defect rates and different types of problems. Engineers can learn from the most successful domains but it will certainly come with a cost. Alternatively, they can learn from the mistakes and problems in their own domain, and improve gradually on the base of real defects and real deficiencies. Within each domain, these issues can be related to the maturity or suitability of the standards, the culture of development, V&V or safety, the applied techniques, processes and tools, the engineers experience and training levels, and the managerial constraints (time, cost, customer). The study of these concrete problems (and the related solutions) for a specific domain is what is intended by this work.

1.2 Contributions of the Work

The goal of this research is to propose **an approach to identify quality gaps and consequently improve systems engineering based on the data available from engineering execution quality**. This approach is hereby called an assessment process or assessment framework. In practice, the three main pillars of the work are the use of

empirical data about defects and issues from critical systems, the root cause identification and relation of these issues and defects to the engineering processes, and the measurable improvement in reducing the frequency and severity of issues and defects in critical systems.

The **usage of empirical data about defects and issues from critical systems** enables the provision of factual background intelligence about the dependability of systems, provides concrete evidence of issues, defects and discrepancies and their nature, and allows to confirm or deny the idea that critical systems based on strict requirements and international standards achieve generally very high quality.

Root cause identification and relating the issues and defects to the engineering process components is key to pinpoint what effectively contributed to the existence of discrepancies, to determine the full (and sometimes, complex) chain of events, resources and processes that lead to the problems, and to provide specific solutions to avoid the issues and defects in the future.

The **measurable improvement in reducing the frequency and severity of issues and defects for critical systems** is a central goal to demonstrate quantitatively that the systems have improved, to clearly measure the impact of changing resources or processes (either positive impact or negative/no impact), and to provide concrete feedback to the framework and thus help in improving the whole systems engineering and validation processes.

In detail, the main contributions of this research can be summarized as follows:

- The definition of a defects **assessment process/framework** that can be used and applied in industry in a simple way and independently from the industrial domain. The framework makes use of a dataset collected from existing issues and process deficiencies, and supports the analysis of these data towards identifying the root-causes for those problems and defining appropriate measures to avoid them in future developments.
- The adaptation of the **Orthogonal Defect Classification** (ODC) [92] for critical issues (integrated in the process/framework). In practice, ODC (from IBM) was used as an initial classification and then it was refined according to the gaps and difficulties found during the initial stages of our defects classification. The refinement was applied on the defect types, defects triggers and defect impacts. Improved taxonomies for these three parameters are proposed.
- The integration of a **root cause analysis** process to relate the issues (or issue groups) with the engineering properties. The **engineering properties** (e.g. human and technical resources properties, events, processes, methods, tools and standards) are, in fact, the principal input for the classes of root causes. A fishbone root cause analysis is proposed, integrated in the process/framework and applied to the available dataset.
- The identification of a dynamic **set of root causes and applicable measures** to improve the quality of the engineered systems. These allow the provision of quick and **specific feedback to the industrial engineering teams** as soon as the root causes are identified. The list/database has been compiled from the

dataset and includes the feedback and contributions from the experts that responded to the process/framework validation survey.

- The promotion of a cultural change to appropriately use **real defects data**, which are the main input of the process/framework, and that shall be easily collected, cleaned and updated. The regular use of defects data will contribute to measure the quality evolutions and the progress of implementation of the corrective actions or improvement measures that are the output of the process/framework.

These contributions are described in this thesis with concrete results and a specific case study application. In practice, the central contribution is the defects assessment process/framework (reusable across different domains or industries) that supports analysts and engineers in the classification of issues with a defined (adapted) orthogonal issues classification and allows the identification of the relevant root causes, that are mapped to the systems engineering elements in order to provide measures and improvement recommendations (these cover suggestions to improve the standards, development and V&V techniques, training to the human resources or modifications to the organization or project lifecycles and management). The assessment process/framework is flexible enough to be updated and receive feedback from the implementation of the measures or from judgement of experts. The classification schemes and the root cause analysis techniques applied are not necessarily attached to the process and other techniques can be applied if deemed necessary or if proved more efficient.

As a practical result of the work, a dataset including European space engineering issues has been collected and used through the full cycle of the process/framework. The feedback obtained can be provided to the community in order to improve the development and V&V processes (some of these results have already been provided to European space industries). The resulting root causes identified and measures proposed can be used by the space industry, but also by other safety critical industries, to adapt and improve existing standards (ESA ISVV Guide, ECSS, DO-178, ISO26262, CENELEC, etc.), in particular by identifying the missing V&V activities and promoting a regular/enforced application of the proposed measures.

1.3 Structure of the Thesis

This chapter introduced the context, the problem and the main contributions of the thesis.

Chapter 2 presents background concepts relevant for the current work as well as existing related works. It covers the dataset sources (namely ISVV activities), defects classification schemes, types of procedures for root cause analysis, and background studies on defects/failures analysis and engineering improvements studies.

Chapter 3 depicts and details the approach for defects assessment. The chapter overviews the proposed approach, then details the process/framework steps one by one,

and finally presents a generalization of the utilization and outcomes of the process/framework in order to make it usable in any industrial domain.

Chapter 4 explains the defects data collection and data preparation procedures. In practice, the chapter presents an overview of the data collection and preparation process, then details the data collection and the data preparation, and discusses some data quality, confidentiality and availability issues. Furthermore, the dataset used in this work is described.

Chapter 5 describes the defects data classification procedures that lead from the original ODC taxonomy into the Enhanced ODC proposed for critical systems in the frame of this work. The Enhanced ODC taxonomy is explained and justified in this chapter, which contains an overview of the adaptation procedure, a detailed description of the modifications proposed to the original ODC taxonomy, and the validation strategy applied to the new taxonomy.

Chapter 6 presents the detailed results of the application of the Enhanced ODC proposed in Chapter 5 and of the Process for Defects Assessment defined in Chapter 3. The results support the characterization of the problem types, triggers and impacts associated to the defects under analysis. Based on these results, this chapter also discusses the root cause analysis derived from the study of the defects while presenting hints on how the root cause analysis and the obtained results can be extended/applicable to different critical systems domains.

Chapter 7 presents the procedure used to validate the results of the defects assessment process and how the root causes resolution can be verified in the long term and applied to any critical systems domain. In practice, this chapter presents the results of a survey submitted to a significant number of experts where the procedure to analyze defects, derive root causes and identify solutions was tested and commented.

Finally, Chapter 8 concludes the thesis, summarizing the lessons learned, evidencing the potential of the proposed solutions, and presenting the weaknesses that we believe should be tackled as future work.

This document contains 4 annexes. Annex A contains the details of the survey provided to the experts for validation of the defects assessment process and the obtained results. Annex B provides the textual responses provided by the experts, including comments on the process and additional root causes and measures proposed. Annex C includes the results of the answers to the quantifiable questions of the survey. Annex D presents the description of the data collection elements as used for our dataset collection and preparation.

Chapter 2
Background and related work

In this chapter, we present relevant background concepts and related work. The first topic covered is related to the methodology used to collect and detect defects. In particular, in this work we used defects identified by a team independent from the development/design/validation ones, applying **Independent Software Verification and Validation** (ISVV) techniques. Any other defects detection and collection techniques and method can be applied as long as the defects contain enough detail to be analyzed.

The second topic described in this chapter covers the **defects classification methods**. This is also the second step of our process and is important to allow grouping of the defects types and triggers in order to better define generic corrections and improvements. During our work we have improved the selected classification method that is described later in this document.

The third topic covers **root cause analysis** methodologies. This is the process part where solutions are identified and improvements are proposed. We provide an overview of the existing background and related work and describe the applied root cause analysis in our process description later on this document.

The next topic in the chapter summarizes the **failure analysis** related studies that provided a background for this work. These studies, including academic and industrial studies, are an essential part of the work as they provided inspiration for the structure of our proposed overall approach.

The fifth covered topic is about **software engineering improvements** studies. The objective is to study what is being proposed and how it is being proposed so far in order to adapt also our process in the best way to have real improvements in software engineering.

The last topic described in this chapter is related to **empirical analysis for critical systems**. Empirical studies are quite important as they are the most realistic sources of corrections and improvements, though they are quite difficult to be made available publicly due to confidentiality issues.

This chapter is composed by a short background and motivation description, then we describe the ISVV activities, next we cover the defect classification schemes studied, followed by the root cause analysis techniques considered, the failure analysis and engineering improvements studies analyzed and some final remarks concerning these topics.

2.1 Background Concepts and Motivation

This section presents two general concepts applicable to our study and some motivation for the work performed in the rest of the report.

2.1.1 General Concepts

Most of the traditional systems are still developed according to the generic V-model (Figure 1) which is adapted on a case-by-case situation by the companies to their needs and according to their experience. Most of the concepts used in this work consider a similar development lifecycle, but can be applied to any other lifecycle as can be seen in our conclusions. This V-model encompasses the different phases that ranges from System Concept down to System Operations, and our defects analysis process can be applied to defects arising from any of these phases.

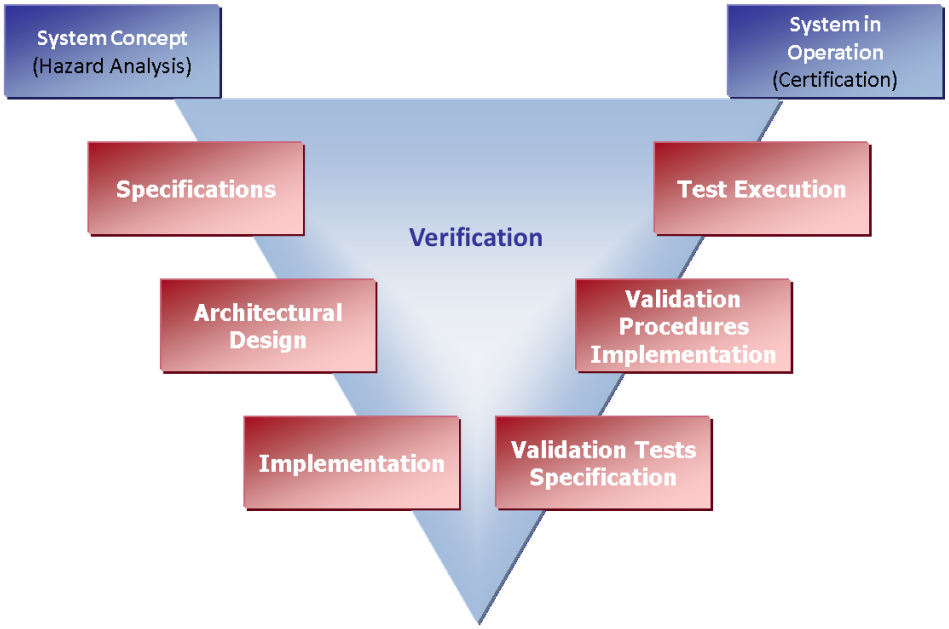


Figure 1: V-model example

Figure 2 depicts a set of important industry standards and shows some relations between them, for example, several standards have been based on the more "generic" IEC 61508 standard. As shown, there are several (overlapped and complementary) standards for each domain. On one side, IEC 61508 is a quite generic and high level

standard that was used to derive very specific and focused standards. From this list, the latest one the ISO-26262 that is still not contextualized in a certification process, as it does not exist for the automotive domain. The airborne set of standards are complementary (e.g. DO-178 for software, DO-254 for hardware, ARP-4762 for safety), and the same is true for the ECSS and NASA standards for space, as they are composed by a set of different standards and handbooks covering different areas of the engineering processes.

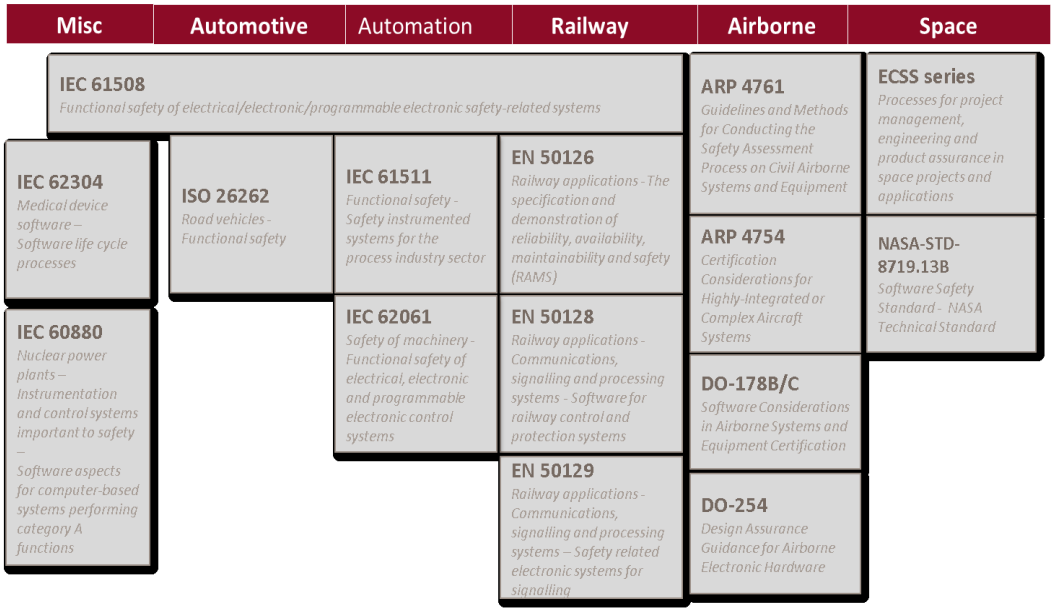


Figure 2: International standards for safety critical systems

2.1.2 Systems and Software Growth and Complexity

The best safety critical software fault densities (between 0.1-1.0 per KLOC) are still not enough since these systems cannot fail, nor contain faults that can cause incidents, accidents or loss of human life (whenever we less expect it). Typical values for commercial software are a few times higher (up to 30 faults per KLOC) [38]. With the growing size, complexity and percentage of software in safety-critical systems, the opportunities for software related problems also increase, thus the development and V&V techniques must also be improved.

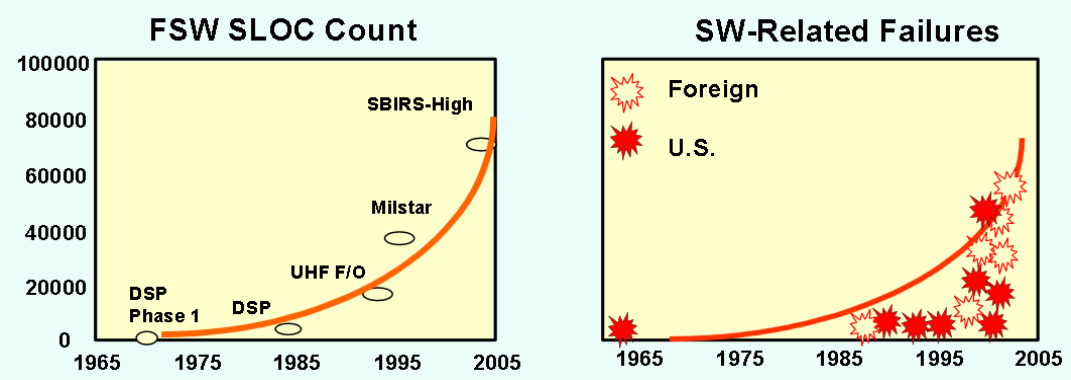


Figure 3: Software increases and software related failures in space systems

Figure 3 presented at the ESA Software Initiative [59], data originally collected by Cheng [60], states that over half of the last three shown years of failures involved software. FSW SLOC indicated the Flight Software lines of code. These graphs show exponential growth patterns both for the size and for the software-related issues occurrence. It gets even more complicated (or extremely costly) if we think that the recent Mars Rover contained about 3.8 million lines of code [61]. For the aircraft software the trend is similar. Figure 4, from [62], shows that by year 2000, about 80% of an US aircraft functions are already performed by software, 40 years before this dependence was only 10%.

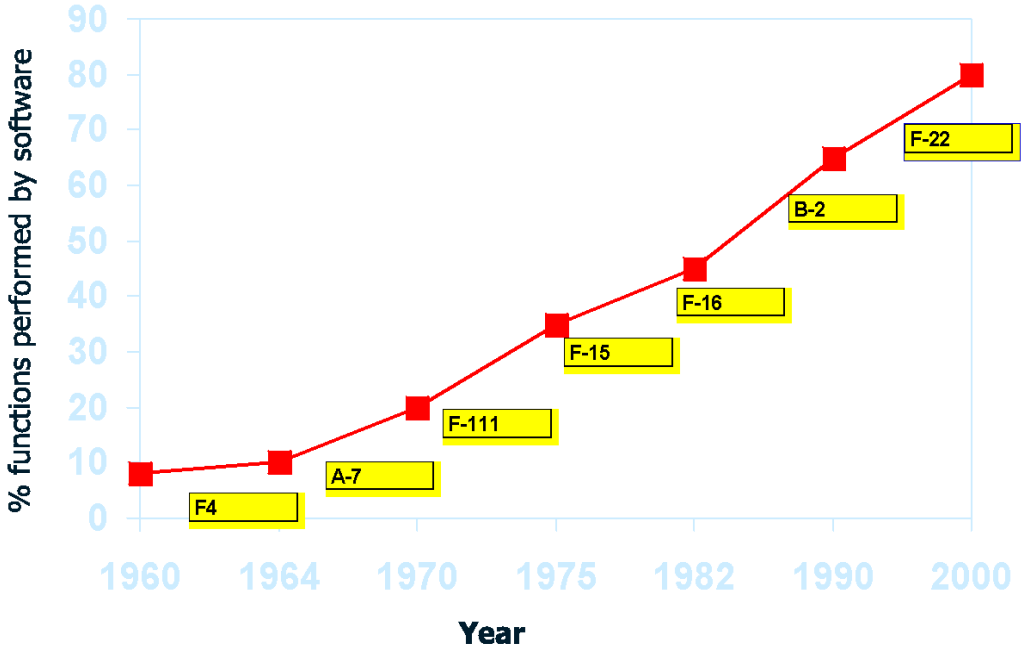


Figure 4: US Aircraft Software Dependence

NASA has also performed a detailed study about Flight Software Complexity [63], this study confirms the growth tendency, but also relates the size with the complexity (for example software interactions and urgency of development) as shown in Figure 5 from [63]. The more complex (and critical systems) are the safety critical ones, such as nuclear and chemical, aerospace and military.

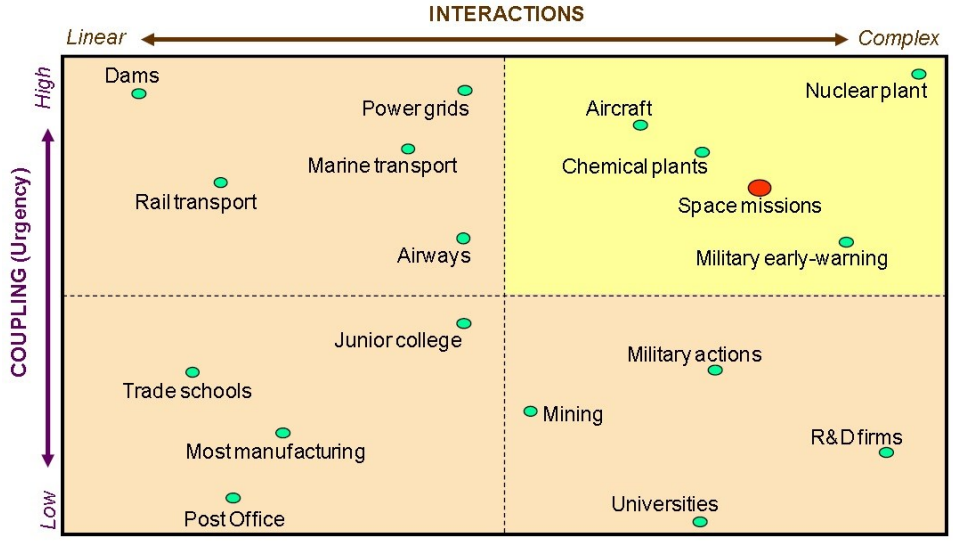


Figure 5: Risk Categorization of systems according to interactions and coupling

The safety critical code tends to be small (to be controllable, maintainable and simple) but that is not always possible. In fact, the airborne software size has grown from about 10 thousand lines of code in 1980 to over 10 million of lines of code nowadays [64] as shown in Figure 6.

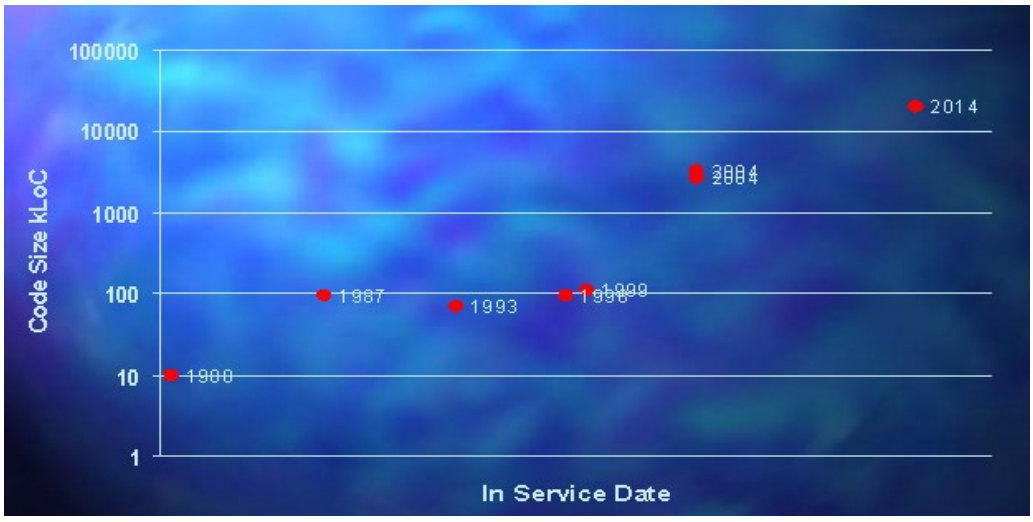


Figure 6: Growth of Airborne Software

2.2 Independent Software Verification and Validation (ISVV)

ISVV stands for Independent Software Verification and Validation, and it is composed by a set of activities performed by and independent team that assesses the software and the system artefacts in order to improve the quality of critical systems and detect defects. This section presents an introduction to ISVV and some details about the ISVV techniques and methods.

2.2.1 ISVV Introduction

Any techniques that allow the detection of defects can be applied in software engineering in order to improve its quality. We might think about the traditional verification and validation (V&V) activities that are essential to any software engineering lifecycle. These are, commonly, tasks that identify issues and allow their "immediate" correction, most of the time not improving the software engineering process nor the organization, but the product under development. We hereby have used defects detected during independent assessment activities, after the regular V&V activities have been performed and the found issues corrected.

ISVV is particularly targeted at critical software systems and intends to be an additional tool to increase the quality of software products, thereby reducing risks and costs through the operational life of the software-based systems. ISVV supports engineers by providing assurance that software performs according to the defined requirements, to the specified level of confidence and safety, and within its designed and intended parameters.

ISVV activities are performed by independent engineering teams, not involved in the software development process, to assess the engineering processes and the resulting software products. The ISVV team independency shall be financial, managerial and technical.

ISVV intends to go far beyond "traditional" V&V techniques, applied by project engineering teams. While the latter aim mainly to ensure that the software performs well against the nominal requirements, ISVV is especially focused on non-functional requirements such as safety, performance, robustness and reliability, and on conditions that can lead the software or system failures. ISVV results and findings are fed back to the development teams for correction and improvement and these modifications are later confirmed by the ISVV teams (acting similarly to independent safety assessors).

ISVV is a set of structured engineering activities supported by tools that allow independent analysts to evaluate the quality of the software engineering artefacts produced at each phase of the engineering lifecycle. ISVV is commonly performed on mature artefacts, which follow strict engineering standards (due to the nature of the domains where ISVV is applied), and that have been previously verified and validated as part of the regular engineering processes. It provides an additional layer of confidence and is not expected to find a large number of severe defects. ISVV produces

evidences that support measuring the quality of the engineering processes, of the software and of the human and organizational resources involved in the software engineering processes. ISVV is referenced in several international standards: a) ISVV guide from the European Space Agency (ESA) [65]; b) ISO Software Lifecycle Processes (ISO/IEC 12207) [66]; c) IEEE Software V&V (IEEE 1012) [67], NASA IV&V Quality Manual [68], and mentioned in DO-178B [34].

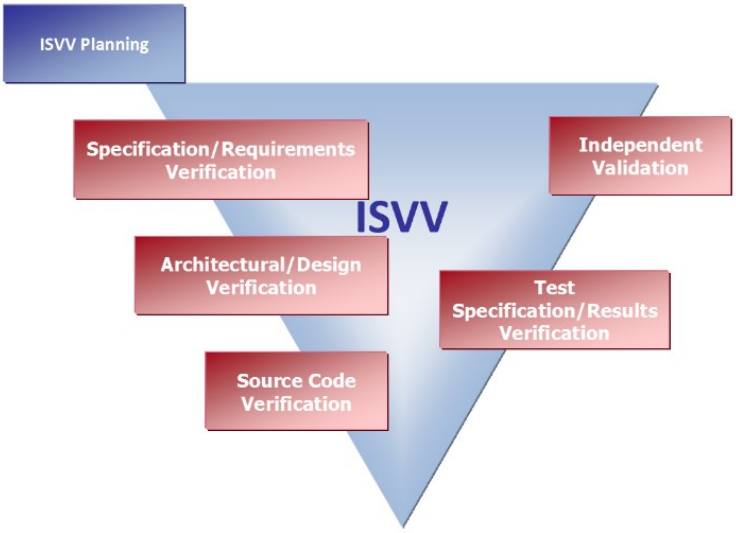


Figure 7: ISVV phases

ISVV includes six basic phases that can be executed sequentially or selected/adapted as the result of a tailoring process based on a criticality analysis [65]. These phases are (see Figure 7):

- **ISVV Planning**: planning of the activities to be performed 8based on the size and complexity estimations), definition of the ISVV level that will impact the set of techniques to be applied, System Criticality Analysis (through a set of Reliability, Availability, Maintainability and Safety – RAMS – activities), and selection of the appropriate methods and tools to be applied.
- **Specification/Requirements Verification**: verification activities for completeness, correctness, consistency, testability, etc.
- **Architectural/Design Verification**: verification of design adequacy and conformance to software requirements and interfaces, internal and external consistency checks and verification of feasibility and maintenance.
- **Source Code Verification**: verification of the code for completeness, correctness, consistency and traceability through code inspections, code metrics analysis, coding standards compliance verification and static code analysis.
- **Test Specification/Results Verification**: verification of the test artefacts, which might include test specifications, procedures, results and reports, as well as traceability verifications and completion of test areas.

- **Independent Validation**: validation activities based on the identification of unstable components/functionalities and missing testing areas to promote validation focused on Error-Handling.

According to the ESA ISVV Guide [65], ISVV engineers classify defects considering three severity levels as described in Table 1.

Table 1: ISVV Severity Levels

Severity	Description
Comment	The discrepancy found does not present any threat to the system. The issue was raised as a recommendation that aims at improving the quality of the affected item.
Minor	The discrepancy found is a minor issue. Although it does not present a major threat to the system, its correction should be done.
Major	The discrepancy found refers to the lack of pertinent information or presents a threat to the system. The correction and/or clarification of the discrepancy are pertinent.

Each ISVV defect is also classified according to an ISVV defect type:

- **External consistency**: differences between implementation of artefacts between phases or with other applicable or reference artefacts (e.g. inconsistent documentation);
- **Internal consistency**: inconsistency against another part of the same artefact (e.g. different code for similar purpose, differences within the same document or architectural components);
- **Correctness**: item incorrectly implemented or with technical issues (e.g. erroneous implementation, wrong documentation description, bad architectural definition);
- **Technical feasibility**: item not technically feasible with the actual constraints (e.g. unattainable or impossible requirement, architecture nor viable);
- **Readability and Maintainability**: item hard to understand and/or maintain (e.g. lack of comments or no description, requirements too complex or too generic);
- **Completeness**: item not completely defined or insufficient details provided (e.g. missing details, missing architectural components, insufficient requirements, not all requirements coded);
- **Superfluous**: item that is a repetition or brings no added value to the artefact (e.g. repeated requirements, copy-pasted code doing the same actions);
- **Improvement**: suggestion to improve any property of the artefact usually not related to a single of the other classification types (e.g. efficiency, simplicity, readability);

- **Accuracy**: the item does not describe with precision or follows the applicable standard (e.g. measurement precision, calculation precision, exact implementation).

Some previous studies, based on ISVV results and data collection, have analyzed metrics, efficiency and efficacy of the ISVV results and techniques used within ISVV to identify the defects in critical projects [69], [70], [71], [72] and [73]. The essential conclusions from these studies are that ISVV still finds a significant amount of issues even after the regular V&V activities have been carried out as seen in Table 2 from [71] (0.69 issues per requirement, 0.97 issues per 1000 lines of code, 0.32 issues per test case), the identified issues are accepted by the customer with over 80% of acceptance rate, and all ISVV phases contribute effectively to find defects from specification analysis down to the very important independent test analysis.

However, none of these studies considered their observations and results to classify or group the defects and improve the development processes, techniques, tools, or standards at large, they have been used to correct each individual issue one by one.

Table 2: ISVV Issues results

Metric	Requirements	Design	Code	Test	TOTAL
RIDs per Requirement	0.25	0.2	0.14	0.1	0.69
RIDs per 1000 SLOCs	N/A	N/A	0.24	0.73	0.97
RIDs per Test	N/A	N/A	N/A	0.32	0.32
RIDs per hour	0.15	0.22	0.15	0.21	-
% Major Issues	21%	15%	21%	21%	-

These previous studies have shown that existing standards and good engineering practices are not enough to guarantee the required levels of safety and dependability of Critical Systems. Independence of V&V avoids author bias and is often more effective at finding defects and failures. It can be managerial, financial and technical, it brings separation of concerns, complementarity, second/alternative opinions, and it also has the merit of pushing development and in-house V&V teams to focus on the quality of their work. The role of independence at early development phases is highlighted in [74] and clearly stated in the requirements of several standards such as CENELEC [75] (depending on the SIL level), and DO-178 [34] (where for example, for the most critical level -A-, 33 out of the 71 objectives/requirements of the standard must be satisfied with full independence).

2.2.2 ISVV Technologies, Techniques and Methods

This section presents a summary list of commonly used techniques for ISVV of critical systems, focused in aerospace (aeronautics and space) standards. These techniques are

referred in the safety critical standards presented in Table 3, which are one of the targets for the assessment process suggested improvements, because this list and the current practice are not complete, some techniques are not properly applied or not applied at all in some domains, and some standards only suggest the use of some techniques in a very generic way.

Table 3 illustrates the main applicable V&V techniques that have been identified in the standards, organized in 12 groups. ECSS-E-ST-10-06C [76] is reported although it does not describe V&V processes, but it does mention some V&V techniques. ECSS-E-ST-10-03C [77] and ECSS-Q-ST-20-10C [78] are not reported in table as they provide little or no relevance to specific techniques. We note that all standards have elements from group 1 and group 11, that is, analysis, reviews, traceability and testing are common keywords of all V&V processes and related aerospace standards surveyed. From this classification, the DO-254 [45] is the standard which mentions the highest number of techniques, followed by ARP and ECSS-Q-ST-30C [79]. A marginal note is that FAA HDBK006A [80] mentions Reliability Modelling, but specifying that "Reliability modelling requirements [..] should be limited to simple combinatorial availability models […]; Complex models intended to predict the reliability of undeveloped software […] generate a false sense of complacency." This sentence merges reliability and availability, but also addresses the difficulties in predicting software reliability, currently another open research problem. Finally, we note that little emphasis is devoted to the schedulability analysis technique, even for safety-critical systems.

Table 3: Techniques referred in standards

		ARP4754-A + ARP4761	DO178-B + FAA 8110.49	DO254	Do-178C + supplements	ED-153	FAA HDBK 006°	ECSS-E-ST-10C	ECSS-E-ST-10-02C	ECSS-E-ST-10-06C	ECSS-E-ST-40C	ECSS-Q-ST-30C	ECSS-Q-ST-40C	ECSS-Q-ST-80C	TOTAL
1	reviews, inspections (Fagan, walk-through, …), analysis (traceability, static code analysis, HW/SW interaction, …)	x	x	x	x	x	x	x	x	x	x	x	x	x	13
2	FMEA, FMECA, FMES	x		x			x	x			x	x	x	x	8
3	Hazard Assessment	x	x	x		x							x		5
4	modeling (SW reliability models, Finite state machine, Petri Nets, Finite State Machines, Markov models, …)	x	x	x	x	x	x	x			x	x		x	10
5	Fault Trees, Dependence diagrams	x		x			x	x			x	x	x	x	8
6	prediction methods			x			x					x			3
7	Common Cause Analysis (CCA), Common Mode Analysis	x		x		x						x	x	x	6
8	Functional Failure Path Analysis (FFPA)			x											1
9	Formal methods, model checking		x	x	x	x					x				5
10	Schedulability analysis										x				1
11	Testing	x	x	x	x	x	x	x	x	x	x	x	x	x	13
12	Similarity, service experience, failure statistics	x	x	x	x	x			x			x	x	x	9
	TOTAL	8	6	11	5	7	6	5	3	2	7	8	7	7	

Table 4 also presents the main testing techniques identified in the aerospace standards. Some standards as the ARP-4754A [47] where attention to identify punctual testing techniques is minimal are not reported. DO-254 [45] is not reported, as its testing techniques are hardware specific (built-in, system bench, aircraft testing). We note the

following: FAA HDBK006A [80] mentions reliability growth testing for software systems (not reported in Table 2), which by its description seems simply meaning that fault removal process should be applied on the developed software. ECSS-Q-ST-30C [79] discusses reliability, availability and maintainability testing, but no indications are reported on the specific techniques to perform these tests (probably these are robustness and fault injection, but still not clarified). As it could be expected, the testing techniques that are most mentioned are interface and functional testing, and integration testing, followed by input-based testing, unit testing and stress testing. Timing testing is mentioned explicitly only once, in the ECSS-E-ST-40C [42] (which also mentions the schedulability analysis).

Apart the information in the table, we also observe that ECSS-E-ST-10-03C [77] explicitly mentions test accuracy, tolerance, margin in tests results and inaccuracies, and that ECSS-E-ST-40C [42] discusses intrusiveness of the environment specifying "*testing that the software product can perform successfully in a representative operational and non-intrusive environment*".

Table 4: Main testing techniques referred in aerospace standards

		Avionics				Space						
	Test name	**DO178-B + FAA 8110.49**	**DO-178C + supplements**	**ED-153**	**FAA HDBK 006A**	**ECSS-E-ST-10-06C**	**ECSS-E-ST-10-03C**	**ECSS-E-ST-40C**	**ECSS-Q-ST-30C**	**ECSS-Q-ST-40C**	**ECSS-Q-ST-80C**	**Total**
1	requirements-based	x	x	x								**3**
2	(hw/sw, sw/sw) integration	x	x	x				x			x	**5**
3	input-based (extensive inputs, normal-range, boundary, ...)	x	x					x			x	**4**
4	interface and functional			x	x		x	x			x	**5**
5	unit			x				x			x	**3**
6	white box											**0**
7	black-box			x								**1**
8	grey-box			x								**1**
9	fault tolerance diagnostic				x							**1**
10	fault injection --, failure --							x		x		**2**
11	robustness	x	x									**2**
12	stress							x	x		x	**3**
13	performance						x				x	**2**
14	structure			x			x					**2**
15	low-level	x	x									**2**
16	implementation			x								**1**
17	isolation							x				**1**
18	closed loop											**0**
19	periodic --- during storage,						x					**1**
20	destructive tests (e.g., Burst test)						x					**1**
21	reliability, availability, maintainability								x			**1**
22	usability										x	**1**
23	mechanical, thermal, electrical						x					**1**
24	timing/schedulability							x				**1**
	TOTAL	**5**	**5**	**8**	**2**	**0**	**6**	**8**	**2**	**1**	**7**	

Another relevant source of V&V techniques for aerospace application is the ESA ISVV Guide [65]. During the time of production of this guide we have surveyed also other domains and we have collected a list of relevant, non-exhaustive, techniques that could be useful for independent verification and validation. The surveyed domains included space, aeronautics and nuclear. This guide contains all independent verification and

validation activities detailed per lifecycle phase and according to component criticality, a suggested list of applicable methods and suggested checklists.

The main methods/techniques described in the guide are:

- Formal Methods
- Inspection
- Modelling
- Data Flow Analysis
- Control Flow Analysis
- Real-Time Properties Verification
- Reverse Engineering
- Simulation (Design execution)
- Software Failure Modes, Effects and Criticality Analysis (SFMECA)
- Static Code Analysis
- Traceability Analysis

The impact of methods and techniques in the quality and dependability of software-based systems is undeniable, and we could go on in listing more V&V and development techniques that influence somehow the engineering process.

2.3 Defects Classification Schemes

One of the first areas that were considered important for the research was the issues (or defects) classification. This is an essential part of the defined process since it directs all the subsequent process phases and the results are dependent on a proper classification. The main topics that were researched concerning this topic include: defect classification, orthogonal defect classification, empirical data analysis and classification taxonomies. This section presents these studies and then focuses on the orthogonal defects classification topic.

2.3.1 Defects Classifications Studies Background

From the different possible defects classification taxonomies (e.g. as described in [81], [82] and [83]), the one who has been adopted and consistently used by industry is ODC [84]. The list of taxonomies is extensive, sometimes connected to a specific engineering phase, and commonly questioned by the practitioners that tend to propose some adjustments. These taxonomies are often complex (Kaner, Falk and Nguyen's Taxonomy [85] contains about 400 types of defects). Some examples of considered defect classification taxonomies are:

1) Beizer's Taxonomy [86],

2) Kaner, Falk and Nguyen's Taxonomy [85],

3) Robert Binder's Taxonomy [87],

4) Whittaker's "How to Break Software" Taxonomy [81], [82],

5) Vijayaraghavan's eCommerce Taxonomy [81], [82],

6) Hewlett Packard Taxonomy [88],

7) IEEE Standard Classification for Software Anomalies [89], [90],

8) Orthogonal Defect Classification [84], [91] and [92].

Some researchers have compared different taxonomies, such as in [83] where the authors presented a framework for comparing six of the previous defect taxonomies, the results of the evaluation and concluded that all of them presented deficiencies. The Freimut report [93] presents the aspects of a defect that have been measured in the literature and possible structures of a defect classification scheme with examples of frequently used defect classification schemes. It also presents general methods to analyze defect classification as reported in the literature as well as concrete analyses for a variety of purposes.

The Vallespir [83] analysis has shown that only two of the six analyzed taxonomies (from the list above: 1.2, 3, 6, 7 and 8) guarantee orthogonality:

- The IEEE Standard Classification for Software Anomalies, and
- The Orthogonal Defect Classification (ODC).

The IEEE Standard Classification for Software Anomalies [90] provides a uniform approach to the classification of software anomalies, regardless of when they originate or when they are encountered within the project, product, or system life cycle. Data thus classified may be used for a variety of purposes, including defect causal analysis, project management, and software process improvement. However, this IEEE taxonomy might lead to quite extensive taxonomies to be easily applicable in an industrial and recurrent context. Yet, this taxonomy has not been extensively used in industrial defects assessment.

ODC [84], [91] and [92] is one of the more adopted defect classification approaches, originally proposed by IBM. In ODC a defect is classified across several dimensions: (1) type, (2) source, (3) impact, (4) trigger, (5) phase found, and (6) severity. There are only eight options for the defect type making it easy and still covering the defect type space. Defect triggers represent a limited list of detection techniques for finding the defects, connecting defect types and triggers. ODC is quite generic and applicable to different domains but mostly oriented to design, implementation and testing originated defects.

From the multitude of studies and reports where defects classification has been applied and studied we have selected some of the most important and relevant ones that are presented and summarized hereafter.

Orthogonal defect classification using defect data to improve software development [94]: This paper describes ODC and illustrates how ODC can be used to measure development progress with respect to product quality and identify process problems. The paper presents the results of a feasibility study conducted by the Motorola Corporate Software Center, Software Solutions Lab and the Cellular Infrastructure Group, GSM Products Division's Base Station Systems software development group using ODC.

The repeatability of code defect classifications [95]: This paper evaluates an adaptation of ODC with the Kappa statistic. Defect data from code inspections conducted during a development project was used. The results indicate classification repeatability, in general. Improvements are suggested to improve classes of defects categorization. The author notes that defect classifications are subjective and this is why it is necessary to ensure that the classifications are repeatable (classification not dependent on the individual).

Using defect patterns to uncover opportunities for improvement [96]: This paper presents the application of Bellcore tool Efficient Defect Analyser (EDA) that supports ODC to three case studies, identifies the main pitfalls of the classification (incomplete data, wrong classification, etc.) and provides some future directions.

Improving software testing via ODC: Three case studies [97]: This paper presents the results of applying ODC to three case studies in order to improve software testing. For the first case study, with a high maturity development process, the study has provided specific testing strategies to reduce field defects. For second one, a middleware project, it identified areas of system test that needed to be improved. For the third, a small project, small team and inadequate testing strategy, the study made the team acknowledge the project risks, schedule delays and proposed necessary missing testing scenarios. The authors claim that ODC helps the identification of actions to increase the efficiency and effectiveness of development and test.

Classification and evaluation of defects in a project retrospective [98]: This study consists of three investigations: a root-cause defect analysis (RCA) study, a process metric study, and a code complexity investigation on an optical network project. The authors made a correlation between the classification and the root-cause analysis and the adherence to the applicable development process.

Empirical analysis of safety-critical anomalies during operations [99]: This paper presents some results from applying ODC to analyze about 200 hundred operational anomalies from seven different spacecraft systems. They found interesting (unexpected) classification patterns and lead to identification of proposed improvements to the software, the development process and the operational procedures.

Defect categorization: making use of a decade of widely varying historical data [100]: This paper describes the results of an aggregation of historical datasets containing inspection defect data (with different categorization schemes). By using historical data and ODC-based classification the authors intended to create models to guide future development projects. A very interesting set of recommendations for classification of defects is provided in the paper.

Defect Classifications and Defect Types Revisited [101]: This short position paper summarizes the work of defect classification as applied in academia and industry with similar classification schemes but none widely accepted. The paper identifies a set of challenges and proposes generic directions in order to get a more widely accepted and common classification.

A systematic literature review to identify and classify software requirement errors [102]: This paper presents a systematic literature review to develop taxonomy of errors (i.e., the sources of faults) that may occur during the requirements phase of software lifecycle. Improvement of the requirements engineering and the overall software quality are the goals of this new taxonomy. The study identified 149 papers from different domains related to requirements faults. The authors provided a categorization of the sources of faults into a formal taxonomy that provides a starting point for future research into error-based approaches to improving software quality.

Using orthogonal defect classification in a Norwegian software company [103]: This report presents the work of a defect analysis and orthogonal classification made at a Norwegian company (not disclosed) by applying statistical methods. The authors found issues with the completeness of the defect report data, and analyzed the injection phase and time to fix of the defects. They have suggested some improvements but mostly to the defect reporting process.

Software Defect Analysis - An Empirical Study of Causes and Costs in the Information Technology Industry [104]:This master thesis report, by collecting defect reports from three different types of projects, represents the results of a quantitative and qualitative analysis (root-cause analysis). The authors concluded that there are differences among project types with regard to root causes for defects, and differences similar between different levels of effort required to correct defects. It was not possible in this study to measure how the differences influenced the root causes.

Quality Evaluation and Improvement Framework for Database Schemas - Using Defect Taxonomies [105]: This paper proposes a defective patterns taxonomy for database schemas. The authors identify four main classes of defects, namely complex constructs, redundant constructs, foreign constructs and irregular constructs. They develop some representative examples and discuss ways of improvement against three quality criteria: simplicity, expressiveness and evolvability. The proposed taxonomy and framework is applicable to quality assessment and improvement.

AutoODC: Automated generation of Orthogonal Defect Classifications [106]: This paper presents an approach and tool for automating ODC classification by casting it as a supervised text classification problem. The authors seek to acquire a better ODC classification system by integrating experts' ODC experience and domain knowledge into the learning process via proposing a novel Relevance Annotation Framework. The case study was from the social network domain and allowed reduction of manual classification with an accuracy of about 80%.

Classification of defect types in requirements specifications: Literature review, proposal and assessment [107]: The authors made a literature analysis about requirements defects classification taxonomy, they do mention ODC but conclude that it is more indicated to classify code defects, they proposed a modified classification for

requirements defect types. The new classification was found to be essential to the analysis of the root-causes of the defects and to their resolution. They have also concluded that his new classification might still not be consensual, i.e. might have different interpretations.

Classification of Software Defect Detected by Black-Box Testing: An Empirical Study [108], is a study and an ODC adaptation for black box testing activities only. Li et al effectively made a detailed analysis for black box types of tests, but the defects detection methods and techniques can be much larger as the ones in our used datasets.

All the papers and reports introduced above are somehow related to the research work we developed. They use defect classification schemes and taxonomies and some of them clearly point out that these schemes are not perfect, while suggesting improvements. Some papers go a bit beyond and identify root-causes and propose improvements either to the classifications, to the defect reporting processes or to the development/validation processes. None of these studies clearly studies safety-critical defects and map them up to the characteristics of these systems, namely the influence of the standards and certification processes. We feel, however, that all these works are a valuable input for our work.

2.3.2 Orthogonal Defects Classification (ODC)

The Orthogonal Defect Classification (ODC), originally proposed by IBM (Chillarege et al. [92]), is one of the most used defects classification approaches. It is intended to be generic and applicable to different technology domains, but it is mostly oriented to design, code and testing defects. ODC defines eight attributes for defects classification, divided into two main groups: a) opener, and b) closer. Three attributes (Activity, Trigger and Impact) classify the defect when it has been discovered and so they are part of the opener group. The other five attributes (Target, Type, Qualifier, Age and Source) are used when the defect is resolved, being thus part of the closer group. The full taxonomies for each attribute can be obtained from the ODC v5.2 specification [92]. A description of ODC attributes is summarized in Table 5.

Table 5: ODC attributes description

ODC Attribute	Description
Activity	The actual activity that was being performed at the time the defect was discovered. The main activities applicable to this work are: Requirements verification, design verification, code verification, test verification and test execution.
Trigger	A trigger represents the environment or condition that had to exist for the defect to surface.
Impact	The impact is the effect that the team who is classifying the defect thinks it would have on the system if not corrected.
Target	Represents the high level identity of the entity that was fixed.
Type	The defect type is defined according to the fix that is necessary to remove it from the system. For that reason, it is best classified by a team/person who applied the fix to the defect.
Qualifier	Captures the element of a non-existent, wrong or irrelevant implementation.
Age	Categorizes the age of the defect, whether if it is new or surfaced from a previous defect.
Source	Describes the source of the defect in terms of its developmental history.

As for several of the previous works about defects classification taxonomies, we had to tailor the taxonomy for the attributes Trigger, Impact and Target, as described later in this document and presented in [1], [2] and [7], that allow ODC to better comply with the needs of space critical software systems. In those works, we have analyzed the original ODC attributes and, with simplification in mind, as well as usefulness for root cause analysis, we have picked those three as the essential attributes for our process.

2.4 Root Cause Analysis

Once the defects properly are classified, we can perform a root cause analysis (RCA) in order to identify what were the sources and events that lead to the defects occurrence or detection. RCA supports the identification of why an issue occurred contrary to only identifying or reporting the issue itself, it also allows the identification of the underlying cause(s) of the issues and helps in preventing additional rework and proactively address future recurrences of the issues.

Some examples of root cause analysis techniques are [109]:

Five Whys: The "5 Why's" can show how causes connect; and it really simplifies the cause and effect relationship into a linear progression and typically focuses on the Action causes.

Failure mode and effects analysis (FMEA): FMEA is a systematic procedure and tool that helps identify every possible failure mode of a process or product, to determine its effect locally or globally. The FMEA also ranks and prioritizes the possible causes of failures of a process or product and can determine the frequency and impact of the

failure as well as suggest and implement preventative actions and compensating provisions.

SIPOC (Suppliers, inputs, processes, outputs, customers) diagram: SIPOC is a high level tool that simplifies the variables of any given process into five segments: S for suppliers, I for inputs, P for process, O for output and C for customers. During brainstorming sessions, team members determine the variables that are relevant to a given process under analysis.

Flowcharting of the process flow, system flow, and data flow: Performed by flow charting the process, system and data flow, and assembling a group of experts to analyze the situation, while drafting a new version of the flow chart with the information related to the events, facts and justifications.

Fishbone diagrams: The Fishbone method is a simple tool to identify the sources of cause: Man, Machine, Method, Material, and Environment (there are variations of the categories used on the Fishbone, another is: People, Procedure, Hardware and Nature). The Fishbone diagram is not intended to show how all these causes interact with each other, unless the analyst has experience in interaction analysis.

Critical to quality metrics: Critical to quality metrics are relevant measures of attributes of a part, product, or process that is critical to quality or that has a direct and significant impact on the actual or perceived quality.

Pareto chart: By studying and understanding data in the format of bar graphs that categorizes the frequency of a certain type of event.

Statistical Correlation: Identify relationships (correlations) between variables where they exist and discount them where they don't by using regression analysis and taking appropriate decisions.

Design of Experiments (DoE): DoE helps improving the capability of a process by identifying the process and product variables that effect the mean and the variance of the quality characteristics of a product.

The fact is that we do need to control the causes of problems [110] as a main objective, and problems cannot be solved without solving their causes [111]. To do this project analysis, called retrospectives are used, they are step-by-step processes [112]:

- Problem identification;
- Problem causes identification (using RCA) – the why [111];
- Cause-and-effect relationships are also identified;
- Root causes detection [111];
- Improvement suggestions for the identified root causes.

RCA is performed more or less frequently for every domain, for software engineering, besides some of the articles already mentioned in section 2.3 and that are related to the defect classification schemes we can highlight the survey made by Lehtinen et al [113] where they propose a new tool for RCA but they surveyed another 35 existing tools to

perform on-line RCA. They separate their analysis in Software project retrospectives, RCA and distributed retrospectives and the actual comparison of the 35 + 1 tools, but their goal was to tackle the on-line and distributed properties of software projects teams.

For the purpose of this work we have surveyed some relevant research works. Besides the already mentioned papers, the following papers and books present relevant inputs for the part of our process where root cause analysis is applied and applicable.

Review of Root Causes of Accidents due to Design [28]: A Safbuild project report produced by Eurocontrol providing the results of a review study from different industry databases of the proportion of accidents that have their root causes in design. It includes accidents from aviation, railway and nuclear industries and concludes that about half of them are root caused to the design phase by providing the more frequent root causes identified.

Root Cause Defect Classification (RCDC) for Documentation Defects [114]: Rao has made an industry study about root cause defect classification for documentation defects, analyzing a few dozen defects on a monthly basis.

Defect Analysis and Prevention for Software Process Quality Improvement [115]: Kumaresh et al conducted a study with data from a few hundreds of collected defects, where these defects have been classified and the corresponding root causes have been proposed to the learning of the projects as preventive ideas.

A case study in root cause defect analysis [116]: This paper presents a retrospective root cause defect analysis based on defects identified during different phases of a transmission network product. The authors present an RCA approach and classification and the results obtained as well as lessons learned. This work is related to [98].

Quantitative Analysis of Faults and Failures in a Complex Software System [117]: This paper presents the results of quantitative study of faults and failures of two releases of a commercial system. They studied correlations and fault prediction metrics and identified or denied some evidences and connections between components complexity and size and fault density, for example.

Using defect analysis feedback for improving quality and productivity in iterative software development [118]: This paper deals with defect analysis as a feedback mechanism to improve the quality and productivity in a software project developed iteratively. The authors discuss how defects found in one iteration can provide feedback for defect prevention in later iterations.

Root Cause Analysis: Simplified Tools and Techniques [119]: This book, first edited in 1999, describes the basic techniques and tools for root cause analysis.

Root Cause Analysis: Improving Performance for Bottom-Line Results [120]: This book describes RCA as a structured investigation of a problem to detect which underlying causes need to be solved.

Root Cause Analysis of Product Service Failure Using Computer Experimentation Technique [121]: In this paper, the authors propose a methodology of performing RCA and corrective actions in design by linking warranty failures with

product design parameters. An analytical approach based on computer experimentation technique performs RCA of product failures (linking warranty failure modes with design parameters) and identifies the analytical relationship between them. They perform the identification of root cause(s) to address in tolerance product design faults. The case study used was an automotive ignition switch.

Applying Root Cause Analysis to Software Defects [122]: In this short article Kaushal highlights the importance of RCA that it needs to get commitment from the institution champions and managers and that is an investment that must focus on finding solutions to improve the overall processes.

The analyzed articles and books present RCA and Retrospective analysis as the way to identify the root causes of faults or problems and address them instead of treating the symptoms or effects. In safety-critical systems we do need to take into account both, we can never ignore the symptoms and effects and we care about reducing possible causes of faults at a minimum (e.g. RAMS analysis for safety-critical systems). RCA has been a process and a set of tools that grew out of accident/incident investigations and became a standard feature of modern engineering, still no so frequently applied in industry. If something is failing, instead of just fixing it at the point of discovery, RCA allows investigation and supports fixing the underlying causes at the point of origin.

2.4.1 Fishbone diagrams

Some of the previous studies seem to conclude that there is not perfect or preferred root cause analysis techniques, instead a set of tools together will produce better results. We hereby describe the commonly used Fishbone (also called Ishikawa) diagram analysis that can be complemented by any of the other analysis to support correlation analysis between the different attributes.

The fishbone diagram, also called Ishikawa or cause and effect diagram helps, through brainstorming, to identify lists of causes of a problem and in grouping the causes into relevant categories. A fishbone diagram is a visual tool to look at causes and effects. It is a structured approach for brainstorming causes of a problem (more structured than e.g., the Five Whys tool). The problem or effect is displayed at the head or mouth of the fish, then possible contributing causes are listed on the smaller “bones” under various pre-defined cause categories. A fishbone diagram is helpful in identifying possible causes for a problem that might not otherwise be considered by directing the analysts to look at all the pre-defined categories and think of alternative causes. The analysts must be experienced in the problem domain ad be also aware of the process and other systems involved in the event to be investigated.

The main steps to conduct and appropriate fishbone diagram analysis can be summarized as follows (see Figure 8 for a simple example):

Problem statement: Clearly define the problem or the event/effect that is being analyzed. Position this problem (or effect) at the head or mouth of the “fish.” Make sure the problem is not a solution and is clear for all the involved experts.

Categories definition: The lists of categories for the causes of the problems is a key step. Common categories often include: equipment or supply factors, environmental factors, rules/policy/procedure factors, and people/staff factors. Common examples of main categories are: Man, Machine, Method, Material, and Environment, or People, Procedure, Hardware and Nature.

Brainstorm: Brainstorm the possible causes of the problem (leading to the effect). Ask "Why does this happen?" Note down the causes proposed by the participants in the brainstorming as a separate branch of the main category (causes can relate to more than one main category and so must be written under several categories).

Keep asking Why: Similar to the five Whys keep asking "Why does this happen?" for all the identified causes. Note down the proposed sub-causes related to each cause.

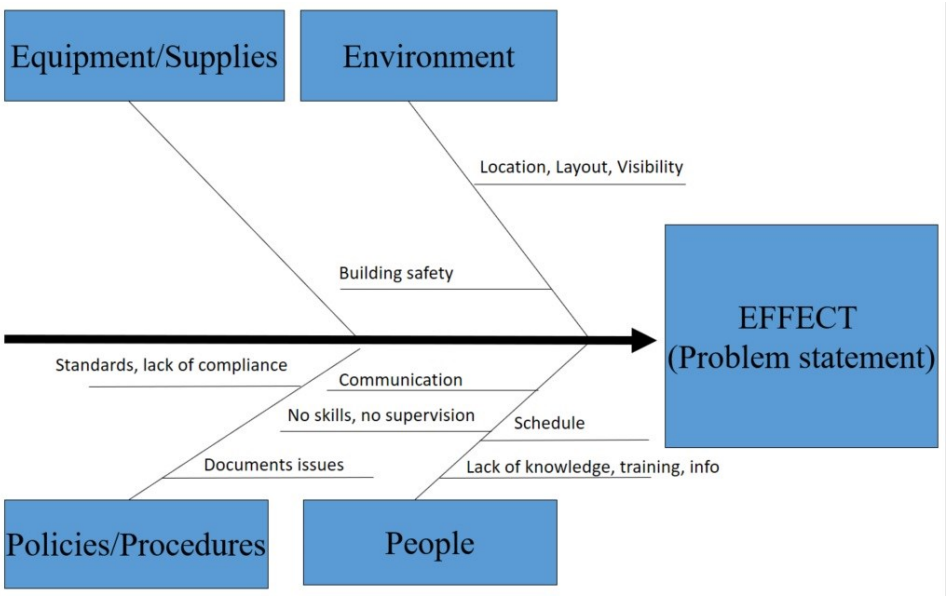


Figure 8: Fishbone diagram analysis example

2.4.2 Five Whys

The 5 Whys is a technique used in the Analyze phase of the Six Sigma DMAIC (Define, Measure, Analyze, Improve, Control) methodology. .It is applied by repeatedly asking and refining the question "Why" (five is simply a good rule of thumb), and this way the different levels of symptoms are identified, eventually leading to the root cause of a problem.

The 5 Whys can be used individually or as a part of the fishbone (also known as the cause and effect or Ishikawa) diagram. Thus supporting in the exploration all causes that result in a single defect or failure. The 5 Whys technique can be applied to drill down to the root causes once all inputs are established on the fishbone.

An example of 5 drilled down questions attributed to the creator of the 5 whys techniques, Talichi Ohno, is:

1) "Why did the robot stop?" - The circuit has overloaded, causing a fuse to blow.
2) "Why is the circuit overloaded?" - There was insufficient lubrication on the bearings, so they locked up.
3) "Why was there insufficient lubrication on the bearings?" - The oil pump on the robot is not circulating sufficient oil.
4) "Why is the pump not circulating sufficient oil?" - The pump intake is clogged with metal shavings.
5) "Why is the intake clogged with metal shavings?" - Because there is no filter on the pump.

2.4.3 Failure Mode and Effects Analysis

The Failure mode and effects analysis (FMEA) process is a proactive process used to systematically analyze specific or vulnerable areas of a system or process. It is commonly used for system assessment and development and not like the other root cause analysis techniques that are applied once the problem occurs.

The FMEA supports in preventing defects and problems by identifying early in the lifecycle phases the causes that might be hazardous and might cause these problems. The FMEA also provide detection, mitigation and elimination measures that can be implemented before any of the analyzed failures actually occur.

The FMEA is usually managed in a tabular format, where each row represents one specific failure mode (a failure situation) and the columns contain, for example, the contents specified in Table 6.

Table 6: Example of simple FMEA headers

Column Heading	Description
Item No.	A unique identifier for each row in the analysis
Name	The name of the component
Function	Brief explanation of component functionality
Failure Mode	Description of how the component could fail
Local Effects	Description of how the component will react if the failure occurs
System Effects	Description of how system will react if the failure occurs
Fault Detection	How the failure will be recognized as having occurred
Failure Management	Methods (design or process) to manage the failure mode

2.4.4 Fishbone (cause and effects, Ishikawa) diagrams

The fishbone diagram, also called Ishikawa (named after its originator Kaoru Ishikawa) or cause and effect diagram helps, through brainstorming, to identify lists of causes of a problem and in grouping the causes into relevant categories. A fishbone diagram is a visual tool to look at causes and effects. It is a structured approach for brainstorming causes of a problem (more structured than e.g., the Five Whys tool). The problem or effect is displayed at the head or mouth of the fish, then possible contributing causes are listed on the smaller "bones" under various pre-defined cause categories (see Table 7 for a few examples). A fishbone diagram is helpful in identifying possible causes for a problem that might not otherwise be considered by directing the analysts to look at all the pre-defined categories and think of alternative causes. The analysts must be experienced in the problem domain ad be also aware of the process and other systems involved in the event to be investigated.

The tasks involved in constructing a Fishbone diagram can be summarized in three main groups:

- **1. Define the problem**
    - **Problem statement**: Clearly define the problem or the event/effect that is being analyzed. Position this problem (or effect) at the head or mouth of the "fish." Make sure the problem is not a solution and is clear for all the involved experts.
    - **Categories definition**: The lists of categories for the causes of the problems is a key step. Common categories often include: equipment or supply factors, environmental factors, rules/policy/procedure factors, and people/staff factors. Common examples of main categories are: Man, Machine, Method, Material, and Environment, or People, Procedure, Hardware and Nature.
- **2. Brainstorm**
    - **Brainstorm**: Brainstorm the possible causes of the problem (leading to the effect). Ask "Why does this happen?" Note down the causes proposed by the participants in the brainstorming as a separate branch of the main category (causes can relate to more than one main category and so must be written under several categories).
- **3. Identify causes**
    - **Keep asking Why**: Similar to the five Whys keep asking "Why does this happen?" for all the identified causes. Note down the proposed sub-causes related to each cause.

In particular, the Categories definition is an important step to support the brainstorming and to direct the causes identification. Table 7 presents a few commonly used sets of

categories and also the applied set of categories for this work, that comes from a merge of the other categories (in column 4 of Table 7).

Table 7: Fishbone Common/Proposed Categories

Services Industry (4 Ps)	Manufacturing Industry (6 Ms)	Business Industry (6 Ms)	Proposed set of Categories for Software Systems
Policies Procedures People Plant/Technology	Machines Methods Materials Measurements Mother Nature (Environment) Manpower (People)	Method Man Management Measurement Material Machine	Method Man/People Management Measurement Material Machine Policies/Procedures Technology Environment

2.4.5 SIPOC

SIPOC (suppliers, inputs, process, outputs, customers) is a visual way for documenting a process from beginning to end. SIPOC diagrams are also referred to as high level process maps because they do not contain much detail.

SIPOC diagrams are useful for focusing a discussion and helping team members agree upon a common language and understanding of a process for supporting continuous improvement. In Six Sigma, for example, SIPOC can be used during the "Define" phase of the DMAIC improvement steps.

To create the SIPOC table (see example of Table 8), the following steps are required:

1) Name the process. It is suggested to use a Verb and a Noun (e.g. Read Memory Location);
2) Define the process Outputs. The outputs are the results created by the process (e.g. 100 bytes, a report);
3) Define the process Customers. The customers are the consumers of the outputs, all outputs must have a customer;
4) Define the process Inputs. The inputs are the actions or triggers to the process (e.g. a timer, a customer request)
5) Define the process Suppliers. The suppliers provide the process inputs, suppliers can also be costumers.
6) Define the sub-processes composing the process. The sub-processes use the inputs to create the outputs.

Table 8: Template of SIPOC Diagram

Process Name				
Supplier	**Input**	**Process**	**Output**	**Customer**
Entity providing each input	Trigger to the process	Sub-process	Result of the process	Received of the output

2.5 Failure Analysis, Engineering Improvements and Empirical Studies

A literature review has been performed during the duration of the research activities in order to find what the status of failure analysis research was. This review is presented in detail in the following sections we can conclude that some of these papers, books and reports have commonalities with our research work, although none of them fully covers the cycle from empirical data (historical data containing critical issues) to improvements up to the engineering, organization and standards level. Some activities have been performed for a much smaller scope and quite rarely to safety critical systems covering all the system lifecycle phases. Some researchers have focused on specific bounded problems, some have covered defects data from a particular lifecycle phase (e.g. requirements, source code, testing). However, some research work must be mentioned in this section especially due to the applicability to our research and commonalities that have been explored.

Analyzing Software Requirements Errors in Safety-Critical, Embedded Systems [123]: This paper from Dr. Robyn Lutz, analyses the root causes of safety-related software errors in safety-critical, embedded systems. She concluded that software errors identified as potentially hazardous to the system tend to be produced by different error mechanisms (non-safety-related software errors). They arise from discrepancies between the documented requirements specifications and the requirements needed for correct functioning of the system and misunderstandings of the software's interface with the rest of the system. The paper contains the identification of some methods by which requirements errors can be prevented. The objective is also to reduce safety-related software errors and to improve the safety of complex, embedded systems.

In-process improvement through defect data interpretation [124]: This paper presents an approach for the interpretation of defect data by the project teams to help correcting the software engineering process during development in order to improve quality and productivity. The authors use examples of corrections to evaluate and evolve the approach, and to inform and train those who will use the approach in software development.

Software System Defect Content Prediction from Development Process and Product Characteristics [125]: The PhD dissertation from Dr. Nikora had the objective of ensuring reliability for the ever growing, in size and complexity, software space systems by developing new techniques to measure and predict system's reliability and thus influence the development process and change the system's

structure. He has developed a model for predicting the rate at which defects are inserted into a system, using measured changes in a system's structure and development process as predictors, and show how to estimate the number of residual defects in any module at any time and determine whether additional resources should be allocated to finding and repairing defects in a module.

Defect Analysis and Prevention for Software Process Quality Improvement [115]: This paper represents the results of a quantitative analysis of defects that occurred in the software development process for five similar projects and from the classification of various defects using first level of ODC, then finding these defects root causes and use the learning of the projects as preventive suggestions. The paper also shows the improvement in terms of reduction of defects once the preventive suggestions are implemented in other projects.

Defect data analysis as input for software process improvement [126]: this paper describes the results of an analysis of 11879 software defects that have been classified and analyzed in order to determine the defect distributions and what are the most common defect types (defect from 3 companies). The authors noted that functional defects are, by far, the most common (65.5%), they concluded that unclear requirements or documentation only contribute to a residual percentage of defects (under 0.5%). The results of this study can be used to support the engineering process improvement.

Using defect analysis feedback for improving quality and productivity in iterative software development [118]: This paper deals with defect analysis as a feedback mechanism to improve the quality and productivity in a software project developed iteratively. The authors discuss how defects found in one iteration can provide feedback for defect prevention in later iterations.

Using Defect Analysis as an Approach to Software Process Improvement [127]: This presentation (and several other similar papers and presentations from the same author) demonstrate a way to classify bugs (problem reports), according to Beizer's taxonomy [86]. This taxonomy contains nine main categories, which are further detailed in up to four levels. The author claims that categorizing each bug takes about five minutes. The improvements inspired by this defect analysis lead to better products (less defects after release, higher customer satisfaction).

Digital Engineering Institute: Lessons Learned - klabs.org [128]: This webpage presents a scientific study of the problems of digital engineering for space flight systems, with a view to their practical solution. It contains links to the NASA Lessons Learned Information System. The website contains lessons learned from design, analysis, verification, and test of digital systems. It presents several reports with flight problems and the relevant solutions, it might be a good input for the improvement suggestions.

The Top Ten Things that have been Proven to Affect Software Reliability [129]: The work performed by Ann Marie Neufelder includes analysis of field failures and correlation to the engineering development characteristics (a total of 679 characteristics). The data comer from 75 complete datasets and 54 incomplete datasets. Another study (Rome Laboratory model) contained 50 datasets and 220 parameters,

only. She uses correlation analysis and does sensitivity analysis and can rank the development characteristics according to their contribution to field defects and use the most influential characteristics to run a predictive model.

A Literature Survey of the Quality Economics of Defect-Detection Techniques [130] and A Model and Sensitivity Analysis of the Quality Economics of Defect-Detection Techniques [131] study and propose an analytical model for quality economics with a focus on defect-detection techniques. These apply to mainly system tests, defects over document types, the removal of costs in the field, but the authors also admit that these have not been extensively empirically analyzed.

Requirements discovery during the testing of safety-critical software, Software Engineering [132], Analyzing Software Requirements Errors in Safety-Critical, Embedded Systems [133], and Operational Anomalies as a Cause of Safety-Critical Requirements Evolution [134] represent the results of the analysis of failures related to the requirements and the requirements phase as introduction phase.

Software Engineering: Are we getting better at it? [135], is a survey and study where M. Jones provides an interesting analysis about space failures in the frame of the European Space Agency missions, but simply concluded that the main cause for all the accidents was lack of testing. Although better testing could have detected some of the problems, their origin (or cause) could be traced to other engineering deficiencies.

An Analysis of Causation in Aerospace Accidents [136] presents a new model to evaluate the causal factors in a mission interruption of the SOHO (SOlar Heliospheric Observatory) spacecraft. The authors conclude that the causes of that specific accident are quite similar to causes found in other software- related aerospace accidents.

2.6 Final Remarks

This chapter presented the topics related to our research as relevant background and related work. Four main areas of research have been surveyed and used for our work. Firstly, we covered the **Independent Software Verification and Validation**, as one methods that provides techniques to detect defects. Any defect detection method, technique or tool can be applied as long as the defects are properly defined and documented. Secondly, we studied the **Defects Classification Schemes**, to support on the classification of the defects and enable the analysis of the causes. For the purpose of this work Orthogonal Defects Classification was considered the most applicable and mature scheme. Thirdly, we covered the **Root Cause Analysis**, where the defect data (types, triggers) are traced back to the causes and enable definition and application of solutions both in the form of better or earlier detection of elimination of the problems themselves. Lastly, we analyzed **Failure Analysis** techniques, Engineering Improvements and Empirical Studies, as a set of activities that take empirical quality data from defects or failures (from the engineering of critical systems), studies them by identifying under what conditions and what caused them, or what could detect them at an earlier phase of their engineering, and by defining solutions and improvements for the quality measured by the reduction of defects or accidents over time).

The extensive amount of literature about these subjects demonstrates the importance of the topics, and the fact that industry is still striving to keep up with the technology advancements required by the society while improving the quality of the dependability of the developed systems shows that defect avoidance is a key aspect of critical systems. In fact, the defects or failure rates achieved today are still high and still cause unacceptable failures. Thus, the need to merge the advantages of these topics (defect or failure detection methods, classification, root cause analysis), and to build up on previous research work and lessons learned to endow industry with appropriate processes to learn from past mistakes and continuously improve the engineering processes, and consequently the engineering products in what concerns dependability, safety, security and so on.

Chapter 3
Process for Defects Assessment

There are several stages for the definition of an appropriate and widely acceptable process for defects assessment. Not only the process must be able to help in solving the existing problems, but it must also be based on proven methodologies and technologies and shall be able to integrate with the existing engineering processes. To define our defects assessment process, we started by drafting and applying a workflow (based on empirical data) that, iteratively, led to the process detailed definition. This workflow is depicted in Figure 9, which shows not only the relation between the different research topics covered (defects classification, root cause analysis, issues correlation, process feedback, engineering lifecycle feedback), but also the importance of the integration of the defects data and the engineering applied processes, as described in the following paragraphs.

From left to right of Figure 9, we find the existing engineering practices (in red, depicted as *Processes* for each safety critical domain, since every domain applies specific processes, standards and tools), the objective, which is to improve these processes quality and dependability, and the existing knowledge (the actual data from defects and engineering processes performance).

For a specific *Process* (way of doing engineering for a particular domain) we need to collect, sanitize and study data (mostly, defects classification to start). The classification and aggregation or clustering of the classification classes are performed to support the root cause analysis of groups of issues (instead of an analysis per issue), leading to some issue pattern identification, which makes the resolution of the problems more efficient. Then, the identified root-causes are mapped to the engineering processes and all that influences the engineering (*General Engineering Process Framework* in Figure 9), and an analysis is done on how to avoid or eliminate the root causes for future applications.

The process summarized in Figure 9 has been adapted and adjusted according to the available data and has evolved to the more detailed process described in Section 3.1. Section 3.2 details the data collection and preparation. The defects classification activities are described in Section 3.3. The relevant root cause analysis tasks are

detailed in Section 3.4. Section 3.5 presents the tasks related to improvements and validation of both the process and the defects data. Finally, Section 3.6 concludes the chapter with some key remarks.

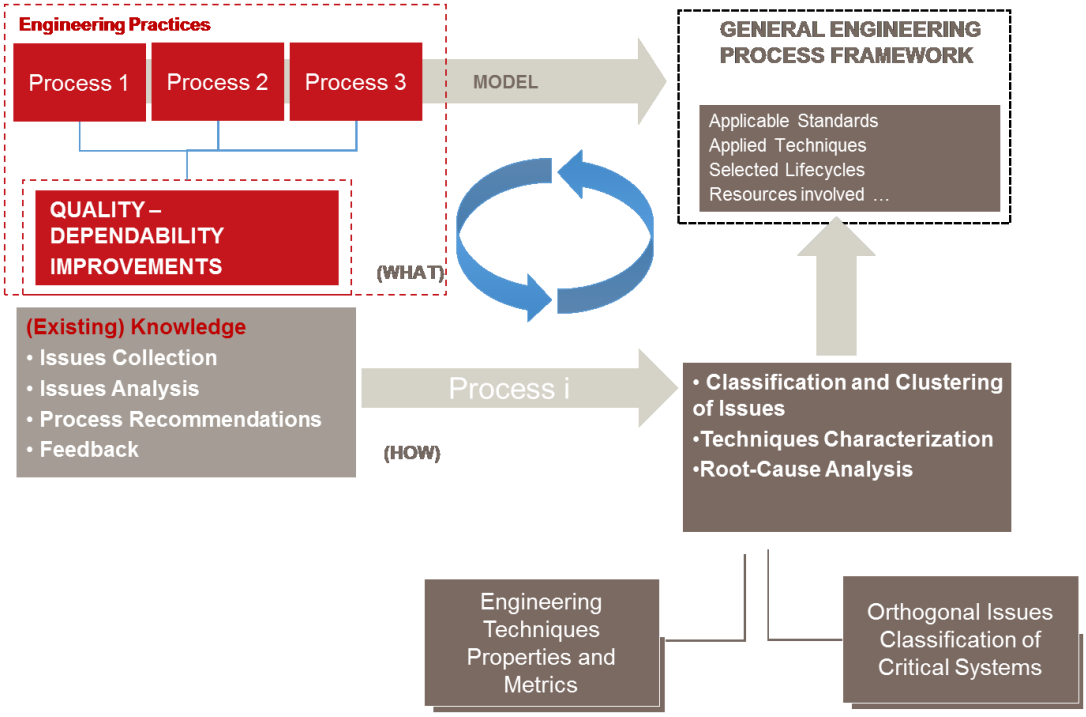


Figure 9: Overview of the proposed process

3.1 Overview of the Process

The defects assessment process needs to cover the data collection and preparation, data analysis and relevant feedback identification. Our proposal can be divided in four main phases (refer to Sections 3.2 to 3.5 and Figure 10 for details):

1) **Data Collection and Preparation**: defects data collection and preparation, aggregation of other data if necessary, such as complexity metrics, lifecycle data, etc. In practice, the issues (defects data) and the phase of issue introduction, the phase of correction, the type of project, etc., represent the main process inputs.
2) **Defects Classification**: classification of individual defects according to an orthogonal defects classification schema/taxonomy to identify the defect types, triggers and impacts. Note that the classification taxonomy can be adapted for specific domains and technology purposes.
3) **Defects Root Cause Analysis**: based on three perspectives (defect type, trigger and impact), identification of the root causes of the defect groups (e.g. per type, per trigger). Several root cause analysis techniques can be used (alone or

complementary) and the analysis can be applied only to the data on defect types, to the defect triggers, to the introduction versus detection phase information, or to any other combination.

4) **Improvements and Validation**: act upon the identified root causes, at a process, organizational or resources (human and techniques/tools) level, and measure the effects of the implemented actions. It is recommended to propose improvements to the systems under analysis (both environment/organization and processes) and also to the defects classification process in order to make it evolve and more applicable to the domain under analysis.

Based on the empirical analysis conducted and the lessons learned and feedback collected during the course of our research, we have refined the approach for root cause analysis of critical software, enabling the continuous improvement of engineering implementation and V&V at all levels (processes, techniques, tools, personnel, application of standards, organization, and so on). Although our dataset (details on the dataset are presented in Section 4.4) and our experience is mainly from space software, our approach can be used to support the evaluation and root cause analysis of any critical system, independently from the domain. Figure 10 depicts the general approach, which includes the data collection and preparation, the defects classification, the root cause analysis and a continuous improvement procedure.

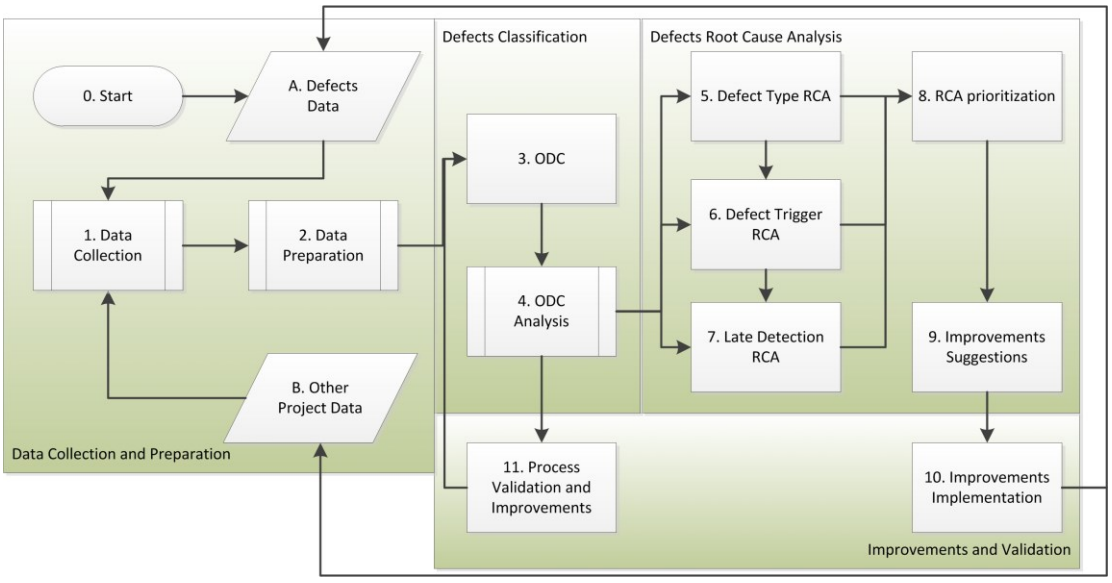


Figure 10: General Process Definition

Out of the 4 phases in our process, we highlight the particular importance of the work performed during phase 2 (*Defects Classification*) and phase 3 (*Defects Root Cause Analysis*). The following paragraphs discuss these two phases from a broad perspective, while sections 3.2 to 3.5 provide further details about the four phases of the process.

The **defects classification** is based on an orthogonal classification taxonomy, and can start once the defects data are collected and prepared. In practice, the ODC classification is performed on the organized dataset, taking the defects one by one. Enhancements and adaptations to the ODC taxonomy can be useful depending on the nature of the defects and the domain, however, these enhancements should be quite precise. The results of the classification will give the engineer a first view of the types of defects, of the main triggers that lead to the identification of these defects, and on the possible impacts distribution.

Root Cause Analysis (RCA) refers to determining how each defect was introduced and identifying the defect source. Identifying the defect source helps in preventing the root cause recurrence and finding process improvements. In practice, root cause analysis can be summarized as the process of finding the activity, process or action that caused the defects and supporting in eliminating/reducing the related effects by providing remedial measures. Two main principles drive the defect root cause analysis:

- **Analysis done/performed by experts**: internal resources with expertise to understand what went wrong must support the analysis of the processes prevalent in the organization, but also independent experts shall be involved. This way, all possibilities are reviewed, analyzed and the best possible actions are defined.
- **Reduction of the defects to improve the system quality**: the RCA must drive changes in processes, tools or human resources that improve the defect prevention at the earliest stage possible (defect origin) and ensure the early detection in case of recurrence.

The analysis of the enhanced ODC application (defects classification) to the dataset of defects is complemented by the identification of the root causes for the majority of the defects based on the classification results and the knowledge of the technological domain and environment, processes, methods and tools. The reasoning is that it may be quite expensive to identify the root cause for every single defect, thus we focus on the more frequent and more severe defect types – other strategies can also be applied.

One possible strategy could be to start by analyzing the more frequent and more severe defect types to determine their causes (origins). The set of causes obtained, once solved, would influence the severity and the recurrence of the remaining defects. There are several techniques and tools that can be used to facilitate the root cause analysis process, such as Fishbone (Ishikawa) diagrams, Pareto charts, change analysis or 5 Whys analysis.

Note that the root causes are not identified at the moment of resolution of the issues but at the moment of the analysis using the ODC or an enhanced ODC. In practice, they are the result of the analysis done on the defects by experts (in the present work, performed by the authors and complemented and reviewed by the industrial partners). Note also that several defects do not have a clear and unique root cause but a set of related root causes.

The root causes analysis shall include the root causes that originated the identified defect types (these are related with development issues) and also defect issues according to defect triggers, which represent the V&V weaknesses. In a subsequent step, the sets of root causes will lead to a dedicated list of measures to tackle both sets of root causes (development and V&V).

3.2 Data Collection and Preparation

The approach is based on defects data analysis and software engineering knowledge. Thus, it is important to fulfil some prerequisites prior to the efficient and correct application of the process, namely (refer to Figure 10 for each activity and to Chapter 4 for further details about the data collection and preparation tasks):

1. Data Collection: to successfully perform the analysis of the defects, the data collected (*A. Defects Data* and *B. Other Project Data*, in Figure 10) should contain the relevant and necessary information. This includes basic requirements, such as: *a)* detailed information about each defect and its fix; *b)* knowledge about defect environment conditions, such as tools, personnel, constraints; *c)* engineer's assessment of the defect causes; and *d)* phase when the defect was introduced and phase when the defect was detected.

Complementary prerequisites are also essential for a successful application of the process. The first one includes training on the involved techniques depicted in Figure 10, such as defects classification (e.g. ODC) and root cause analysis. The second includes rules and guidelines (such as a standards, templates) for defects description or defect data collection. For the proper application of the process it is essential that the collected defects contain a minimum of information, including: reference artefact, defect title and defect detailed description, phase where the defect was identified, phase where the defect was introduced, activity that detected the defect, defect author, and defect severity.

2. Data Preparation: once we have the necessary data it is important to organize it and perform some anonymization when required (when the process is applied internally in an organization this step is obviously not necessary). Data organization is essential for the next steps, since it is important to have the data in a searchable and manageable manner. It is also important to confirm the completeness of the data, from a defects description perspective, but also from all the complementary engineering information (life-cycle phase, techniques applied that lead to the defect detection, impact analysis, etc.).

In the case where some missing information that would affect the classification or the RCA is detected, this phase shall promote the completion and collection of that information using as source the defect author or the referenced artefacts (documents, tools, code, tests, etc.).

3.3 Defects Classification

To efficiently and concretely tackle the important problems of critical software engineering, the first set of dynamic analysis activities shall focus on an orthogonal classification of the sets of defects (see Chapter 5 for further details about the defects classification task):

3. ODC Classification: perform the ODC classification on the organized dataset (the completed set of defects). Enhancements and adaptations to the ODC taxonomy can be useful depending on the nature of the defects and the domain, however, these enhancements should be quite precise. Other classification taxonomies can be used if they are appropriate, well known by the user and provide relevant information (grouping, prioritization, clustering) to support the root cause analysis.

4. ODC Analysis: the goal is to analyze the classification results and provide a summary of the main findings, in particular in what concerns the distributions of defects types and triggers. This information gives the first hints about the quality of the dataset (defects frequencies, impacts, distributions), which can provide some quick feedback to the implementation (defect types results) and V&V teams (defects triggers results).

3.4 Defects Root Cause Analysis

The proposed root cause analysis is composed by several steps that include analysis of the defect types, the triggers allowing defect detection, the defects that could have been detected earlier, and later prioritization and consolidation of these root causes leading to concrete proposed improvements (see Chapter 6 for our results):

5. Defect Type RCA: the classification of the defect types will define which are the most common/frequent types of defects. If these defects are also mapped to high severity impacts, their prevention can efficiently reduce the impacts. With this data from the defects classification we can identify what caused the specific defects with the more common types, try to aggregate them, identify common root causes and common solutions.

6. Defect Trigger RCA: similarly, when the defects classification provides the most common defect triggers, it is possible to quickly conclude that those activities have not been efficient in detecting the defects earlier (in case they could be detected earlier), but also, the results provide the list of triggers that actually detect them and that can be applied from now on by the V&V teams to detect further defects as early as possible. These results support the identification of the causes and V&V techniques or triggers that allow the defects detection at the current phase.

7. Late Detection RCA: when relevant information is made available and it is possible to determine at what point in the lifecycle the defect was introduced (either generated or not detected) it is interesting to determine why certain defects have not been spotted and solved before, and why they have slipped through phases. The root cause analysis

of these specific slipped defects helps in identifying the causes of the failures in the V&V and ISVV techniques that allowed the defects to propagate until a later stage in the lifecycle.

8. RCA Consolidation: defects prioritization can be done to simplify the RCA and to make it more efficient. It can be done to tackle the defects with high impact on the system, or simply to analyze the defects with more common types or triggers (due to the large amount of defects and respective causes). Note that, it may happen that defects of the same type do not have the same causes, but at least the RCA will provide a list of causes for the majority of the defects and thus provide a quick reduction of defects when those causes are fixed.

9. Improvements Suggestions: after the prioritization of the lists of root causes from steps 5, 6 and 7, a root causes consolidation is required. As the list can become very extensive, causes may be merged (if appropriate) and ordered according to the prioritization performed, or to another specific root causes prioritization. For the consolidated root causes, define changes, solutions or modifications to the processes, techniques, tools, training, resources, environment or application of standards. The definition of the improvements should come straight forward from the list of root causes.

3.5 Improvements and Validation

Some of the suggested improvements might be difficult to implement, and their efficacy may vary from team to team or from organization to organization. They shall contribute to improve the software quality and reduce the number of defects or prevent them, but different defects can then surface, and therefore a consistent process improvement should be in place and shall provide constant feedback:

10. Improvements Implementation: the engineering (development and V&V) teams must be informed about the required changes or adjustments, and the organization, management and quality planning shall decide on the improvements to implement for future projects. This can be provided in the form of process improvements, dedicated workshop or training sessions, lessons learned sessions, improved guidelines or standards, etc.

11. Process Validation and Improvements: at every step, it is possible to derive improvements to the process. Such improvements can be set to adjust to the organization culture, to the project environment, to the customer requirements, etc. However, it is essential to measure the effectiveness of the implementation of the results once the suggestions have been implemented and new defects (or no defects) have been collected. Note that improvement can and shall also be about the current process, the defects classification scheme and taxonomies, the root cause analysis techniques and so on. The proposed process is able to adapt and help in improving itself and its composing techniques. For the validation of the process see Chapter 7.

3.6 Final Remarks

The process for assessment of defects has been applied and adapted according to our empirical study (see Chapter 5 and Chapter 6 for the results). It was also exposed to a large set of worldwide experts (see Chapter 7) that provided their feedback, even if most of them were from distinct domains and with a different background. Generally, the process was accepted with a "recommendation" rate of 60% and "possible recommendation" rate of 33%, while only 7% would not recommend such a process.

The process described in this chapter contains some strengths and weaknesses that are summarized here. The strengths are: *i)* the process itself represents a structured approach; *ii)* includes explicitly root cause analysis; *iii)* is based on an orthogonal classification scheme; *iv)* allows the provision of improvements and feedback; and *v)* relies on high quality of data. On the weaknesses, we should mention: *i)* concerns about how to guarantee the quality of defect data; and *ii)* large number of steps in the process; and *iii)* difficulty in implementing/enforcing such a process and the obtained results.

These remarks are obviously very relevant. First, the process will only work if the defect data are appropriate, of good quality and complete. For this, we shall relate weakness *i)* with weakness *iii)*, as a cultural enforcement must be broader than just the application of the process, but also cover the defect data collection, the quality checks of the data, the changes necessary to a certain way of working, and so on. The large number of steps is required to have the process detailed with simple blocks. Furthermore, the process contains permanent self-feedback and also feedback to the development and V&V processes, as a result of the root cause analysis suggestions.

The list of the essential suggestions made by the experts and that would make the process generic enough to cover different domains and different types of technologies, include (see Chapter 7): data collection improvements (process, database, quality guarantee); classification/validation activities and data quality check between phases of the process; consideration of projects details/specifics and team dynamics (skills, experience, motivation); and assessment covering also management related issues. In practice, we can observe that we have suggestions on the environment and prerequisites, which make absolute sense (data quality, projects details) and also on the validation of the internal process activities, namely the quality of the classification that cannot be easily automated as per today's technologies.

Each group of process tasks (Data Collection and Preparation, Defects Classification, Defects Root Cause Analysis, and Improvements and Validation) are described in the subsequent chapters, providing more details and the outcomes of the application of the process to our case study. In practice, Chapter 4 details the data collection and preparation processes and activities (steps 1 and 2 of the process), Chapter 5 describes the defects classification process and taxonomy adaptations (steps 3 and 4), Chapter 6 covers the root cause analysis activities and results (steps 5 to 9), and Chapter 7 presents the strategy and results of the process validation and implementation (steps 10 and 11).

Chapter 4
Data Collection and Preparation

Organizations have a challenge related to defect data management. First of all, these data are usually confidential and sensitive to the organization or the organization suppliers, and thus not usually available. Secondly, industry tends to not properly and completely document defects (especially industries with a lower CMMI maturity level), as they are more concerned about deploying the systems in a short timeframe and fixing the issues as soon as possible, than on really focusing on measuring and process improvement. Third, several cultural barriers (as the ones identified by Jäntti et al [137]) do not ease the implementation and usage of a defects management system, including data collection, acceptance and communication of the issues, organizational-wide processes, etc. Therefore, defects data management (collection, preparation, recording, analysis) is not an easy task. Moreover, the results of the whole defects management process depend on the quality and availability of data, making data and the way it is collected the most important step for any defects analysis activity.

Data collection and preparation is a set of complex processes that require organizational sponsorship, and organization wide processes require data collection rules or training, as well as standard defects collection contents. Furthermore, the defects data must be stored, either in a tool/database or in a set of documents, this being also an organizational strategy that needs to cope with data access and confidentiality issues. Defects data must be made available and must be complete or provide means to be completed with additional information (resources, author, documents, references, etc.).

For this work we had access to real defects data that were produced to inform the stakeholders about the issues with an acceptable detail level. However, for some of the defects further information was needed, as for example to consult the original artifacts or documents to better understand the problem and its origin. The data was collected and prepared according to the format described in this chapter. Important actions had to be taken to preserve the confidentiality of the data, in particular the projects from

where they came from and the involved stakeholders, as these would publicly expose weaknesses of the engineering process of those organizations.

The chapter is organized as follows. Section 4.1 presents a global view of the data collection and preparation related activities. Section 4.2 provides the details of the data collection tasks, next we present the applied data preparation and clean-up activities (Section 4.3), we then provide information about the defects included in our used dataset (Section 4.4), and we conclude with some final remarks about the data related tasks (Section 4.5).

4.1 Overview of the Process

The data collection and preparation depends on the completeness and quality of the input data. For this purpose and in order to use the data for our defects assessment process we have taken a straight forward approach to collect and prepare those data. The procedure is depicted in Figure 11 and described next.

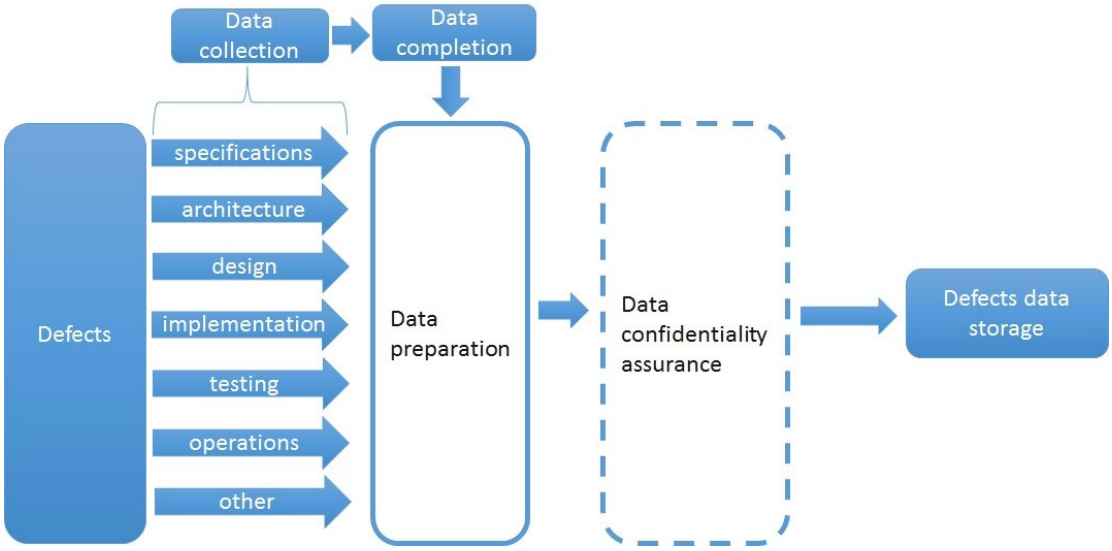


Figure 11: Data collection and preparation procedure

Defects can be originated and detected at any of the lifecycle phases, so the first important activity is to collect information (**data collection**) about the defects (from specifications, architecture, design, implementation, testing, operations, and other phases or V&V activities) in a structured way and with enough information to enable an orthogonal classification and later support the root cause analysis. This is exactly why there is also a **data preparation** phase, where the collected defects data are structured and completed or complemented with additional information sources (**data completion**).

The data collection and preparation phases can be based on a guideline or template to collect the minimum of defect data required for the next defects assessment activities

(see the used template in Annex D and the data collection and preparation details in sections 4.2 and 4.3). Other actions might be applied to the data to make them workable and available to the following phases of a defects analysis process. For example, we can apply some data clean-up, removal of unnecessary defects or details, and particularly data anonymization for **data confidentiality assurance**, during the data preparation activities (Section 4.3), if the data are to be made available to third parties. In the case where the data is being internally analyzed and there is no need to involve any third-party expert or to reveal the results to any external entity, this step is simplified and not required. Finally, the processed defects data need to be properly stored and managed (configuration management). For this, a tool or a dedicated database can be developed, for example by using Microsoft Access or Excel databases and manage their configurations with Git[1], CVS[2] or SVN[3] repositories.

4.2 Data Collection

Defects data can come from different sources, including V&V reports, Excel databases, emails, and bug tracking web based tools. All these sources contain different formats and different levels of details for the defect data. Furthermore, a selection should be performed on the data to be collected, in order to include defects from different types of systems and subsystems, but also to include defects from all the lifecycle phases (specifications, architecture, design, implementation, testing, and operations).

Our analysis is based on a set of real defects from ISVV activities in space projects. The projects include subsystems that compose satellite systems for three different domains: a) scientific exploration; b) earth observations; and c) telecommunications. These cover different types of software, such as start-up or boot software, on-board application software, command and control units, payload software, and attitude and orbit control units. The engineering processes used in the selected missions (and that drove the engineering lifecycles) were based on the ECSS standards, namely the space engineering standard E-ST-40 [42] and the quality standard Q-ST-80 [43], which have a comparable lifecycle and similar strict requirements imposed by the European Space Agency (ESA).

[1] http://www.github.com/
[2] http://savannah.nongnu.org/projects/cvs/
[3] https://subversion.apache.org/

Table 9: Generic caracterization of the subsystems contributing to the dataset

Subsystem	Domain	Software Types
SS01	Earth observation	On-Board Start Up / Boot Software
SS02	Scientific exploration	On-Board Application Software
SS03	Telecommunications	System Software
SS04	Earth observation	Payload boot Software
SS05	Earth observation	On-Board Application Software
SS06	Earth observation	Payload boot Software
SS07	Scientific exploration	Payload Software Payload boot Software
SS08	Scientific exploration	Payload Software Payload boot Software
SS09	Scientific exploration	Payload Software Payload boot Software
SS10	Scientific exploration	Payload Software Payload boot Software
SS11	Scientific exploration	Payload Software Payload boot Software
SS12	Scientific exploration	Payload Software Payload boot Software
SS13	Scientific exploration	Payload Software Payload boot Software
SS14	Scientific exploration	Payload Software Payload boot Software
SS15	Scientific exploration	Attitude and orbit control unit software
SS16	Scientific exploration	Command and control units Software

The subsystems (see Table 9) were developed according to functional and non-functional requirements mandated from ECSS and mission specifics (by ESA). They are characterized by the following needs/objectives, which are common to space critical systems, and that were collected from the ECCS standards [42], [43] and from the corresponding engineering interpretations of the specification documents from several missions:

- No crash or hang shall happen at any time;
- No dynamic memory allocation is allowed;
- Communications-Telemetry (TM)/Telecommands (TC) must always be possible between ground control and the satellite;
- The system must implement a Safe Mode (with basic communications, patch and dump functionalities);

- Most systems shall have a very simple and stable start-up software (also called boot software);
- There must be a watchdog (hardware and/or software) or an alive signal;
- Systems should be built with redundancy (at least hardware);
- Most systems must include FDIR (Fault Detection Isolation and Recovery) functionalities to account for the environment and external faults;
- The systems must have high autonomy and some self-correction procedures;
- Systems are categorized with a criticality level related to the impact or consequences of system failures (in this case, the ECSS defined levels are: Catastrophic, Critical, Major and Minor or Negligible).

The projects could also be characterized by:

- Requirements written in natural language (structured), highly based on documentation and non-formal processes and languages;
- Documentation in UML/SysML and PDF files, with limited possibilities of automated verification and formal analysis;
- Programming languages such as C, Ada and Assembly, that are quite mature and low level languages;
- Unit tests performed using commercial tools (e.g. Cantata++, VectorCast, LDRA), commonly developed and adapted for the specific projects embedded systems and environments;
- Integration and system testing performed in a specific validation environment (Software Validation Facility - SVF) developed for this purpose on a case by case situation, with hardware emulation and hardware in-the-loop, simulated instruments, etc.
- A strong quality assurance process, based on the ECSS standards and monitored by the European Space Agency and complemented by and Independent Software Verification and Validation (ISVV) activities for the critical areas of the project;
- A well-defined and mature Software Development Process (SDP).

The defects collected for our study (also called issues or Review Item Discrepancies – RID) were identified by independent teams in the project development artefacts, after the development teams have performed their own required verification and validation activities, as defined in the ECSS SDP. The selection of systems and defects was based on several criteria, namely (further details about the dataset are presented in Section 4.4):

- The defects are all confirmed (i.e. no false positives);

- The systems were developed for different space missions (in this case, 4 different missions, including scientific exploration, earth observation and telecommunications);
- The systems were developed by diverse prime contractors (3 main European prime contractors) and multiple software development entities (a dozen entities);
- Different types and sizes of systems or sub-systems are included (in our case 16, including start-up and boot systems, central control units, attitude and orbit control systems, payloads control software);
- The defects were originated by different independent assessment activities, and detected after each of the SDP phases (a total of 1070 defects): 162 defects were detected after the conclusion of the specification phase, 112 after architecture, 378 after implementation, 398 after testing and 20 after deployment;
- The defects cover different severity categories, as defined in ECSS Q-ST-30 series. In our dataset, 14% of the defects were classified as *Major*, 66% as *Minor*, and 20% as *Improvements*.

4.3 Data Preparation

The collected defects have been integrated in an Excel database where specific information was added. Some information required was not possible to import from the defects data sources (defects reports) and had to be complemented by external sources and in most cases by consulting the defected artifact or original documentation. This data was either not existing or not documented in the collected defect reports. It includes, for example, the year where the issue was raised, the maturity of the engineering team that generated the issue, or the exact activity (V&V tasks) that allowed the detection of the issue. Furthermore, the introduction and correction phases (Phase Detected, Phase Applicable) were also not explicitly stated in the defect reports but could easily be complemented. The excel spreadsheet has the following structure:

- **Number**: a unique identifier for the defect, usually the original defect identifier;
- **Project**: the name of the project from where the defect originated;
- **Subsystem**: the subsystem or component to which the defect applies – a subsystem code was used in order to anonymize data for external experts;
- **Domain**: the technology domain where the defect applies to (e.g. space, aeronautics, automotive, defense, railway, etc.);
- **System Type**: a description of the system or component type, which can be different for every technology domain (e.g. for space systems it can be: on-board start-up software, payload software, ground control software, on-board

application software, on-board systems, on-board component, command and control system, etc.);

- **Issue Title**: a summarized title for the defect;
- **Description**: the defect detailed description. This description shall be as complete as possible to simplify the classification and the root cause analysis. We have taken the original defect description that would allow full understanding and resolution of the defect;
- **Classification**: the original severity classification of the defect, according to the organization severity classification scheme (e.g. for ISVV there is a classification scheme that was used and is shown in Section 4.4: *Minor*, *Major* and *Comment*);
- **Problem Type**: classification of the defect type according to the original classification scheme (e.g. for ISVV there is a classification scheme that was used and is shown in Section 4.4);
- **Phase Detected**: the lifecycle phase where the defect has effectively been detected and recorded. This column can be adapted to the applicable lifecycle phases.
- **Phase Applicable**: the lifecycle phase where the defect has effectively been introduced. This column can be adapted to the applicable lifecycle phases.
- **Defect Type**: the ODC (and later enhanced ODC) defect type classification for each defect. For details on this classification see Chapter 5;
- **Defect Trigger**: the ODC (and later enhanced ODC) defect trigger classification for each defect. For details on this classification see Chapter 5;
- **Defect Impact**: the ODC (and later enhanced ODC) defect impact classification for each defect. For details on this classification see Chapter 5;
- **Comment**: field used to store information about the classification doubts and suggestions of modifications to the ODC original classification taxonomy. This information was later extracted and used to propose an enhanced ODC taxonomy;
- **Notes**: notes related to the defect understanding or additional defect information;
- **Activity**: the review or V&V activity that led to the discovery or detection of the defect. This is usually the activity that was being performed when the defect was uncovered;
- **Keywords**: field reserved for keywords related to the defect in order to enable future automation of the defects analysis.

Filling this structure was not always straightforward due to the different sources of defects data (mostly from ISVV reports, defect reports and other excel spreadsheets),

thus the data preparation included some data harmonization, in particular for the fields: System Type, Classification, Problem Type, Phase Detected, Phase Applicable, and Activity. Additional information was also collected but later discarded such as the age of the defect (year of raising it), the organization responsible for the defect, and complexity metrics related to the defect subsystem or component. This additional and complementary information was not adding enough relevant information for the root cause analysis and was making the process too complex to be easily applied.

With the defects dataset built and filled we had to consider which amount of the data could go public and which should stay internal to the organization (**data confidentiality assurance** in Figure 11). It is natural that defects originated from a specific team, in particular those that are dealt with internally and during the engineering lifecycle phases, do not go public. No organization likes to have their shortcomings revealed publicly, as they deal with them internally, solve them and present a system or a product with an acceptable level of quality and dependability. Even when these organization are assessed by independent assessors, this information is not revealed and is used internally to correct the issues and improve the engineering methods.

For this purpose, we had to operate some modifications on the dataset in order to eliminate the possibility of identification of the involved parties. The main fields that have been hidden or modified are: Number, Project, Issue Title and Description. All these fields contained sensitive data that could lead to the identification of the organizations involved in the development and V&V, and thus either they have been hidden (the two first ones have been replaced by the Subsystem identifier) or reviewed and anonymized (the two latter ones) in case some sensitive information was included in the title or the description (e.g. the name of the component, the company, etc.).

4.4 Defects in the Dataset

Table 10 summarizes the 1070 defects included in the dataset, divided by severity (having a major or minor impact in the system, or just being comments to improve the engineering) and considering the ISVV activities in which they were found. The defects have been originated from the analysis of more than 10.000 software requirements, more than 1 million lines of code (mostly C, Ada95 and some Assembly), and over 3.000 tests[4] (some unit tests, some integration tests, some system tests). In practice, the objective of ISVV was to find issues in the project artefacts, report and classify them in a clear and consistent way for the customer to act upon immediately and avoid these issues to slip over subsequent phases.

For the particular set of selected defects, Table 10 shows the results of a typical ISVV analysis, i.e. issues identified during all the lifecycle phases (Requirements, Design, Implementation, Testing, and Operations), the large majority of the issues are Minor or

[4] The 3000 tests correspond to only part of the requirements and code referred, as not all ISVV activities cover the full set of artefacts, e.g. for some projects only source code analysis was performed, no tests related to that specific code have been assessed.

Comments, which is consistent with the strict and mature development and validation processes applied for the space domain software, and a significant amount of defects is identified at the implementation and testing phases. This comes from the fact that source code artefacts and testing specifications, procedures and results are the ultimate focus of ISVV activities and represent the large majority of items under assessment, thus generating also a large amount of defects.

Table 10: Dataset of ISVV defects

	ISVV activity					
Severity	**Req. Verification**	**Design Verif.**	**Code Verif.**	**Test Verif.**	**Operation Monit.**	**Total**
Major	27	14	43	62	2	148
Minor	98	84	185	294	18	679
Comment	37	14	150	42	0	243
Total	162	112	378	398	20	1070

The ISVV originally classified the defects with the following classification types (from [65]):

- **External consistency**: differences in the implementation of artefacts between phases or with other applicable or reference artefacts (e.g. inconsistent documentation);
- **Internal consistency**: inconsistency against another part of the same artefact (e.g. different code for similar purpose, differences within the same document or architectural components);
- **Correctness**: item incorrectly implemented or with technical issues (e.g. erroneous implementation, wrong documentation description, bad architectural definition);
- **Technical feasibility**: item not technically feasible with the actual constraints (e.g. unattainable or impossible requirement, architecture not viable);
- **Readability and Maintainability**: item hard to understand and/or maintain (e.g. lack of comments or no description, requirements too complex or too generic);
- **Completeness**: item not completely defined or insufficient details provided (e.g. missing details, missing architectural components, insufficient requirements, not all requirements coded);
- **Superfluous**: item that is a repetition or brings no added value to the artefact (e.g. repeated requirements, copy-pasted code doing the same actions);
- **Improvement**: suggestion to improve any property of the artefact usually not related to a single of the other classification types (e.g. efficiency, simplicity, readability);

- **Accuracy**: the item does not describe with precision or follows the applicable standard (e.g. measurement precision, calculation precision, exact implementation).

The resulting classification of the defects by the ISVV teams is shown in Table 11. The main types of defects are external consistency, completeness, and correctness. These three types account for 75% of the total of ISVV defects. Note that the data shown in Table 11 have been used in an industrial context to provide simple metrics and to help promoting the immediate correction of the issues (including defects). This classification has never been intended to determine the defect types (although it represents generic defect types categories), nor determine the triggers, and only the severity of the impact has been considered in those cases. Therefore, a classification that allows to orthogonally classify defect types, triggers and impacts, as well as to support the analysis of the introduction versus the detection phases, is needed to support the RCA activity.

Table 11: ISVV original defect types classification

Defect Type	Number of Defects	Percent
External Consistency	313	29%
Completeness	275	26%
Correctness	213	20%
Internal Consistency	132	12%
Technical Feasibility	3	0%
Readability & Maintainability	84	8%
Superfluous	14	1%
Improvement	34	3%
Accuracy	2	0%
Total	1070	100%

4.5 Final Remarks

The quality of the inputs is key to any process. In order to ensure relevant results from the application of a defects assessment process, we have collected defects data, prepared and harmonized the defects, and complemented them whenever required. Anonymization was also required due to confidentiality of the defects data. The defects data have been collected in an Excel spreadsheet, which was also used later on for the individual classification of each defect based on the selected orthogonal defect classification taxonomy.

We believe that the simple data collection and preparation approach, complemented with a short training on writing defects and filling the standard defects information can be an important step into the successful analysis of organizational weaknesses and reduction of the number of critical defects in the short/medium term.

Our case study dataset includes 1070 defects from space projects. Data were collected and harmonized from different sources (ISVV reports, excel spreadsheets with issues and operational defects), in order to obtain a coherent set of defects, covering different types of systems and all the lifecycle phases. This dataset is the source for all the activities and tasks of the defects assessment process (Chapter 3), namely the definition/adaptation/validation of the defects classification taxonomy, the root cause analysis based on the results of the defects classification (particularly the defect type, the defect trigger and the phases where the defect was detected versus the phase where it had been introduced) and the identification of correction, suggestions and improvements to both the process and the systems where the defects come from. These aspects will be addressed in the following chapters.

Chapter 5
Defects Classification

The defects dataset has been first classified with ODC v5.2 [92]. The outcome of this initial classification of our dataset was that we could not properly classify all the defects with the existing ODC defect types, defect triggers and defect impacts (see Section 5.1 for the justification of the selection of these attributes). The classification difficulties were annotated when a specific issue was being classified and there was no classification fit or agreement. In fact, we experienced that for 31.7% of the defects, the original ODC taxonomy had some limitations. The issues affected (i.e. the RIDs that could not properly be classified according the standard ODC taxonomies) have been set aside and dully noted in order to contribute to the ODC adaptation.

The standard ODC was thus considered not totally fit for this classification because it was not developed for the specific case of ISVV nor for critical systems, or even to cover the whole engineering lifecycle - from the 1070 defects classified, 136 defect *types*, 76 defects *triggers* and 201 defect *impacts* could not be properly mapped to the standard ODC taxonomy. A fitter ODC taxonomy was possible with some extended types and triggers that cover in a more efficient and concrete way the specific critical ISVV issues (e.g. traceability, verifiability, robustness/dependability and safety related properties).

For example, defects related to tests and requirements specification are not clearly mapped to an existing ODC defect type and they are quite common sources of the reported defects from the *Independent Test Verification* activity. In what concerns the impacts, we can point out the absence of *Testability and Verifiability* specific classifications (as important requirements for critical systems and related standards), and we have also identified a few defects that would fit into more than one ODC impact classification (this indicated some orthogonality issues or the need to break the defect into more than one defect from the ISVV team point of view).

To efficiently and concretely tackle the important problems of critical software engineering, defects classification includes two main tasks (see Chapter 3), the first consisting of applying ODC (or an enhanced version of ODC) to the dataset, and the

second focusing on the analyzes of the classification results to provide a summary of the main findings. This is precisely the goal of this chapter.

The outline of this chapter is the following. Section 5.1 presents an overview of the process to adapt the standard ODC classification and operate the classification itself. Section 5.2 presents the original ODC classification and the identification of needed adaptations. The next section describes the proposed adaptations to the original ODC taxonomy. Section 5.4 details the obtained results of the enhanced ODC classification applied to our defects dataset. Section 5.5 provides the results of the validation strategy of the ODC classification and acceptability of the results. Finally, Section 5.6 summarizes the ODC activities and provides the final remarks concerning the defects classification related activities.

5.1 Overview of the Process

The adaptation of ODC may lead to changes in the taxonomy of the different attributes to make them more applicable, more complete and more adjusted to critical systems defects from all lifecycle phases. Thus, some additions, reductions or merges might be needed over the original ODC taxonomy. Although the ODC general approach remains unchanged after adaptation, the attributes themselves were evaluated and adapted when necessary. In practice, the most relevant inputs for the taxonomy adaptation were the difficulties felt while applying the standard ODC to the defects in our dataset.

We have not considered all eight attributes provided by the ODC specification, leaving aside the following ones: *target*, *qualifier*, *age* and *source*. We did not find the need to use these to achieve orthogonality, to promote a simple and regularly usable classification (with only essential attributes) and to avoid very specific code oriented attributes (not necessary to enable root cause analysis). The selected attributes are the most important and the candidates for adaptation for critical systems: *activity* and *trigger* represent the defect detection method and activity and can thus influence the root cause from a V&V perspective, *type* represents the development defect category and drives the root cause for all development defects, and *impact* can be used to prioritize and cluster groups of defects based on the defects effect on the system and also on the type of effect – safety, robustness, maintainability, etc. A small justification for the attributes not used is provided next:

- **Target** – we claim that this attribute is not needed for orthogonality as the Activity/Trigger/Type is sufficient to derive what is the target of a defect. Target simply represents the high-level artifact that was fixed, for example code, design or requirements. Nonetheless it is additional information that may provide useful support in some specific situations;
- **Qualifier** – we opted not to use this qualifier as our source of defects have this already specified in the description of each defect. Qualifier is simply the type of code fix: addition of missing code, fixing of existing incorrect code, or the removal of extraneous code;

- **Age** – this attribute seems to be useful when describing defects that occur during development, but not when classifying defects of an already developed product such as our case of ISVV, hence we opted to not use it. Age can be useful to compare the evolution of the defects sets over different releases and to study the changes of defect types due to the evolution of technologies, design or programming, languages, etc. This is generally covered by the defects description;
- **Source** – this attribute captures the origin of the code that had the defect (developed in house, reused from a library, etc.) and was not classified because it captures very specific information that is not useful for root cause analysis of groups of defects (unless these groups include the software type, but this information is already include in the defect descriptions anyway).

The above considerations do not imply that these attributes are not useful, as they surely provide additional information that can be used, although some of that information is already included in the defects description. However, it is also essential that the classification process is simple and contains the most relevant attributes to be efficiently applied in industry and adopted by the engineering teams. Thus, for the approach to be used by industry in a regular way, it should be kept as simple as possible and use the essential attributes only.

Our **strategy to perform the ODC enhancement** started by applying the original classification to the dataset, then noting the classification difficulties, later aggregating and clustering these non-classified defects, harmonizing them, and finally defining new attributes or merging existing ones. A validation of the proposed enhanced taxonomy (for type, trigger and impact) was conducted to confirm its fit. The set of defects that were not originally classified or classified with attributes that changed were classified/reclassified by applying the new taxonomy.

The first step to be performed when an organization decides to implement ODC is, as specified in ODC v5.2, to map activities to triggers. Although triggers are given by the ODC specification, the ODC activities are meant to be customizable, defined by each organization according to their approach in defect detection and removal (e.g., workflow, processes and the applicable lifecycles). Table 12 presents a set of activities performed at industrial levels related to ISVV of critical systems and the mapping to the set of ODC triggers. The *Activity* attribute was extracted directly from the ISVV activities and helps in identifying the introduction phase of the defects. Then, for each activity a set of triggers is mapped, these triggers are related to the nature of the ISVV tasks performed for every activity (described in detail in [65]), and have been harmonized in Table 12 to include the ODC set of triggers. As shown, some activities are mapped mostly to documentation/inspection related triggers, as for *Requirements* and *Design*, and some others are related to testing or dynamic execution triggers, as is are the cases of *Test Verification* and *Test Execution*.

Table 12: Mapping between activities and triggers

Activity	Triggers
Requirements verification	Standards conformance Traceability/Compatibility Consistency/Completeness
Design verification	Design conformance Standards conformance Traceability/Compatibility Logic/Flow Concurrency Consistency/Completeness
Code verification	Standards conformance Traceability/Compatibility Logic/Flow Concurrency Consistency/Completeness
Test verification	Consistency/Completeness Logic/Flow White box path coverage Test coverage Test variation Test sequencing Test interaction Workload/Stress
Test execution	White box path coverage Test coverage Test variation Test sequencing Test interaction Workload/Stress Blocked test
Operation monitoring	Design Conformance Workload/Stress Start-up/Restart HW/SW Configuration

In practice, using ODC consists of applying it to the issues to support the analysis and feedback of defect data targeting quality issues in software design, code and documentation. Taking into account information on the issues, ODC identifies a defect type and the relevant trigger (assessment techniques, testing, analysis methods, etc.) for each defect identified. The results can be used for statistical quality control (e.g. measuring improvements), as well as for in-process monitoring and reliability assessment (required for critical systems). They are also frequently used to promote specific process and resources improvements by tackling the identified issues directly.

The analysis of the classification results supports the ODC adaptation: once the classification work has been performed, the obtained classifications are analyzed and

the results used to propose adaptations of the classification taxonomy, if necessary (type, trigger and impact) to be aligned with the domain. Classification recommendations from experts and data clustering analysis can be used to identify classification patterns.

The results of re-classifying the issues with the newly proposed taxonomy allows to validate the recommendations. The adaptations proposed for the classification taxonomy after applied and adjusted to the nature of the critical systems defects shall lead to more straight forward classifications of the defects and thus simplify the classification tasks. Further adaptation can be fed back into the classification taxonomy proposed and lead to a re-classification of the affected issues.

The ultimate goal of the process is to analyze the classification results and provide a summary of the main findings, in particular in what concerns the distributions of defects types and triggers. This information gives the first hints about the quality of the dataset (defects frequencies, impacts, distributions), which can provide some quick feedback to the implementation (defect types results) and V&V teams (defects triggers results).

5.2 ODC Classification Results

The ODC classification of the issues identified by the ISVV teams (meaning that the assessment was made in quite mature software artefacts by independent experts – so these are not regular software engineering lifecycle issues) and during operation, has shown that for 1070 ISVV only 731 could be correctly classified considering ODC type, trigger and impact attributes (the remaining 31.7% justify the improved, more applicable, ODC taxonomy to be orthogonally classified).

Table 13 presents the results of the classification of the 731 defects using the standard ODC. We can observe that the main classified types of defects are Documentation (36.11%), Function/Class/Object (21.34%) and Algorithm/Method (11.35%). These three defect types (that cover more than two thirds of the defects) arise from the fact that the systems under analysis are heavily based on documentation and thus the larger set of defects is naturally of documentation type. Then, several defects are related to the function and the proper implementation of algorithms, being more “functional” defects.

The main classified triggers for the defects under analysis are Document Consistency/Completeness (Internal Document) (22.30%), Test Coverage (19.84%) and Backward Compatibility (19.29%). These three triggers uncovered almost two thirds of the defects. It is worth mentioning that while the first trigger is evident due to the nature of the artefacts under assessment, the second one is related to the fact that most of the testing activities performed for space systems tend to prove coverage of the requirements, and the third trigger is related with traceability analysis results (backward traceability checks).

The impacts identified are Capability (30.23%), Reliability (24.76%), Maintainability (18.74) and Documentation (18.60%). These impacts cover more than 90% of all the

impacts in the dataset. Capability relates to limited functioning of the system, Reliability to the dependability properties that are not met, Maintainability is related to updates, patches and maintenance activities, and Documentation is a minor severity impact related to documentation imparities.

Table 13: Original ODC classification results (731 defects)

Defect Type	Qty	%	Defect Trigger	Qty	%	Defect Impact	Qty	%
Documentation	264	36.11%	Document Consistency/Completeness (Internal Document)	163	22.30%	Capability	221	30.23%
Function/Class/Object	156	21.34%	Test Coverage	145	19.84%	Reliability	181	24.76%
Algorithm/Method	83	11.35%	Backward Compatibility	141	19.29%	Maintainability	137	18.74%
Checking	48	6.57%	Operational Semantics (Understanding flow)	95	13.00%	Documentation	136	18.60%
Interface	48	6.57%	Design Conformance	86	11.76%	Performance	28	3.83%
Understandability	35	4.79%	Lateral Compatibility	46	6.29%	Usability	19	2.60%
Environment	35	4.79%	Combinatorial Path Coverage (Complex Path)	20	2.74%	Migration	6	0.82%
Assignment/Initialization	28	3.83%	Rare Situation	15	2.05%	Standards	2	0.27%
Timing/Serialization	26	3.56%	Language Dependencies	7	0.96%	Installability	1	0.14%
Build/Package	8	1.09%	Test Sequencing	7	0.96%			
			Recovery / Exception	3	0.41%			
			Test Interaction	1	0.14%			
			Test variation	1	0.14%			
			White box path coverage	1	0.14%			
Total	**731**	**100%**	**Total**	**731**	**100%**	**Total**	**731**	**100%**

5.3 Proposed Adaptations (ODC Enhancements)

This section presents the proposed adaptations to the original ODC (v5.2) to make the defect type, defect trigger and defect impact more fit for classifying defects in critical systems. The distinct backgrounds of the tables in this section highlight the changes from the standard ODC taxonomy: a) white - unchanged; red - deleted; yellow - merged; and green - new.

5.3.1 ODC Attributes – Activity

The ODC specification defines activities as defect removal activities. In our understanding, we benefit to expand this definition to also encompass defects that appear in field operation, hence we added the activity ‘operation monitoring’.

The list of activities are the ones performed during the ISVV phases described in section 2.2 and includes operation monitoring: a) Requirements verification; b) Design

verification; c) Code verification; d) Test verification; e) Test execution; and f) Operation monitoring. See Table 12 for the full list of considered Activities.

The main divergence to the standard ODC is the inclusion of test verification and operation monitoring as activities which can be a source of defects. Test verification was included as it is an activity extensively performed in ISVV and other defect detection activities, and was not considered in the standard ODC. Operation monitoring simply represents the issues that have been identified after the system is in operation. This means, the issues or defects detected during the system execution.

In order to properly accommodate these activities, the taxonomy of the other attributes was also adapted (such as the Testability/Verifiability impact and the Documentation type). For more details please refer to the modification descriptions in sections 5.3.2, 5.3.3 and 5.3.4.

5.3.2 ODC Attributes – Type

The type attribute represents where the defect was fixed. Since not all defect types were possible to determine with the original ODC taxonomy, some gaps have been identified mostly to simplify the classification and avoid confusions, but also a new type has been added in order to be able to classify some of the defects. We adapted the ODC v5.2 classification and extended it with the new value when appropriate for our needs, as follows:

- **Algorithm/Method, Checking, Function/Class/Object, Timing/Serialization, Documentation** – same meaning as in the original ODC 5.2.
- **Assignment/Initialization** – similar to the original ODC v5.2, but extendable to cases where, for instance, variable names are changed to be in compliance with coding standards or coding rules (frequently required for critical systems).
- **Build/Package/Environment** – new classification to be applied in defects related to the build process, packaging of data/functionality, and environment setup or configuration. Libraries that are never used or large modules of dead code should also be classified here. Note that cases of code paths that are never reached and with a small scope, such as inside a function, should be classified with the 'Function/Class/Object' type, by default.
- **Interface** – this classification is the result of the merge of '*Interface*' and '*Relationship*' into one single type, as both relate to interfacing (internal or external) and the encountered cases were all related to interface problems – even when they were a relationship issue.
- **Understandability** – removed and merged with '*Documentation*' as this type raised confusion during the classification activities and all the encountered examples could be covered by documentation fixes.

Table 14 depicts the mapping of the standard ODC type taxonomy to the proposed adaptation of the ODC type attribute values.

Table 14: Standard ODC Type to Adapted Taxonomy

Standard ODC Defect Type	Adapted ODC Defect Type
Algorithm/Method	Algorithm/Method
Assignment/Initialization	Assignment/Initialization
	Build/Package/Environment
Checking	Checking
Documentation	Documentation
Understandability	
Function/Class/Object	Function/Class/Object
Interface	Interface
Relationship	
Timing/Serialization	Timing/Serialization

5.3.3 ODC Attributes – Trigger

Triggers classify what actions or checks can reveal the defect. Some changes to the triggers were made from the standard ODC specification in order to simplify and streamline as much as possible (for each trigger a small description provides the rationale in order to better clarify when to use it):

- **Design conformance** – trigger that indicates that the defect was detected while comparing the design, code or test with their specifications and assessing the design and the specification conversion into design or implementation (similar to ODC v5.2).
- **Standards conformance** – this trigger replaces the original '*Language Dependency*' trigger, renaming it and broadening the scope to better suit issues in critical systems (often based on standards). It is applicable to defects that arise when checking items for standards compliance (which typically do not exist for the systems for which ODC was originally defined). This includes requirements not written according to specific rules, and implementation concerns such deviation from best practices. These issues may arise from manual or tool assisted inspection.
- **Logic/Flow** – this trigger identifies incorrect flow of logic or data in the design, implementation or procedure details (similar to ODC v5.2).
- **Traceability/Compatibility** – this trigger replaces both '*Backward Compatibility*' and '*Lateral Compatibility*' in the ODC v5.2 specification. It is applicable in cases where traceability is unclear or missing, or system blocks have compatibility issues. This merge and adaption was deemed necessary to

cover specific requirements related to critical systems and requirements imposed by standards that extensively use traceability.

- **Consistency/Completeness** – this trigger replaces the '*Internal Document*' trigger, providing a more appropriate terminology, such as the one critical systems engineers are used to. Defects related to incorrect information, inconsistency or incompleteness should be mapped here. This trigger is mostly related to inspections and assessment activities.
- **Rare situation** – this trigger is the result of the merge of both '*Side Effects*' and '*Rare Situation*' from the ODC v5.2 specification. We did not find relevant to separate the two in our case study due to their low frequency and similarity.
- **White box path coverage** – merged '*Simple path coverage*' and '*Complex path coverage*', applicable in unit testing when the tester is trying to exercise specific code paths, which is very common in testing strategies for critical systems and generally ruled by the testing strategies. Only path coverage is now considered without distinguishing between simple and complex because no advantage was seen and it is not easy to classify the complexity of path coverage (simplification of the classification).
- **Concurrency, Test coverage, Test variation, Test sequencing, Test interaction, Workload/Stress, Start-up/Restart, Recovery/Exception, Blocked test** – same meaning as ODC v5.2 and no modifications required.
- **HW/SW configuration** – merged '*Hardware configuration*' and '*Software configuration*' to cover configuration issues at large, as no major difference was found that require to keep them separate. Also, hardware and software in embedded systems are strongly coupled, thus making such distinction while classifying issues may lead to many doubts without relevant added value.

Table 15 shows the mapping of the standard ODC taxonomy for triggers with our proposed adaptation.

Table 15: Standard ODC Trigger to Adapted Taxonomy

Standard ODC Defect Trigger	Adapted ODC Defect Trigger
Design conformance	Design conformance
Logic/Flow	Logic/Flow
Backward compatibility	Traceability/Compatibility
Lateral compatibility	
Concurrency	Concurrency
Internal document	Consistency/Completeness
Language dependency	Standards conformance
Side effects	Rare situation
Rare situation	
Simple path	White box path coverage
Complex path	
Test coverage	Test coverage
Test variation	Test variation
Test sequencing	Test sequencing
Test interaction	Test interaction
Workload/Stress	Workload/Stress
Recovery/Exception	Recovery/Exception
Start-up/Restart	Start-up/Restart
Hardware configuration	HW/SW configuration
Software configuration	
Blocked test	Blocked test

5.3.4 ODC Attributes – Impact

This attribute depicts the impact that the defect would have had upon the end user if it was not detected during ISVV (or the defect detection phase), or in the case of defects detected during operation, what was the impact of the failure. The adaptations proposed to the impact attribute are the following:

- **Capability, Documentation, Installability, Integrity/Security, Migration, Performance, Reliability, Requirements, Standards, Usability** – same meaning as in ODC v5.2, no modification required.
- **Maintenance** – merged '*Serviceability*' into this impact attribute, as for critical systems the definition of the two is similar and refer to diagnosing issues and applying corrective/preventive actions.
- **Safety** – added for the special cases where defects in critical systems can directly impact the safety of humans or of the environment (these are specific requirements for many critical systems). There are sets of requirements that are exclusively related to safety and any misinterpretation or failure of these

requirements will endanger the system safety, thus these situations need to be carefully analyzed for these systems.

- **Security** – new taxonomy element to cover the security and cybersecurity growing concerns of modern systems. The security analysis are not yet very common for safety critical systems but there is a growing concern as the systems become online and interfacing with more and more systems. For the case of our study we had mostly on-board systems, so no security flaws have been detected.
- **Testability/Verifiability** – added to fulfil the need to classify defects with an impact in testability/verifiability of the systems. This is important to the applicable standards conformance in critical systems since testability and verifiability are commonly strict requirements that need to be part of the system.
- **Accessibility** – removed this impact that was related to ensuring that successful access to information and use of information technology is provided to people who have disabilities. For our case study, and for most critical systems, this impact is not applicable and can be supported by either "Safety" or "Capability" impacts depending on the situation and applicable requirements.

Table 16 presents the mapping of the standard ODC impact taxonomy to the proposed adaptation.

Table 16: Standard ODC Impact to Adapted Taxonomy

Standard ODC Defect Impact	Adapted ODC Defect Impact
Accessibility	
Capability	Capability
Documentation	Documentation
Installability	Installability
Integrity/Security	Integrity/Security
Maintenance	Maintenance
Serviceability	
Migration	Migration
Performance	Performance
Reliability	Reliability
Requirements	Requirements
	Safety
	Security
Standards	Standards
	Testability/Verifiability
Usability	Usability

5.4 Enhanced ODC Classification Results

The results of the application of the enhanced ODC for space defects are summarized in Table 17. The top 5 defect types, triggers and impacts cover about 90% of the issues analyzed. This observation suggests that actions can be taken to quickly improve the quality of systems, by tackling a limited amount of properties.

Table 17: Enhanced ODC classification results (1070 defects)

Defect Type	Qty	%	Defect Trigger	Qty	%	Defect Impact	Qty	%
Documentation	515	48.13%	Traceability/Compatibility	309	28.88%	Capability	308	28.79%
Function/Class/Object	203	18.97%	Test Coverage	227	21.21%	Maintenance	264	24.67%
Algorithm/Method	96	8.97%	Consistency/Completeness	206	19.25%	Reliability	252	23.55%
Checking	69	6.45%	Logic/Flow	119	11.12%	Documentation	157	14.67%
Interface	56	5.23%	Design Conformance	119	11.12%	Performance	39	3.64%
Build/Package/Environment	52	4.86%	Rare Situation	26	2.43%	Usability	28	2.62%
Assignment/Initialization	46	4.30%	Test Sequencing	16	1.50%	Requirements	9	0.84%
Timing/Serialization	33	3.08%	Standards Conformance	14	1.31%	Migration	8	0.75%
			HW / SW Configuration	13	1.21%	Standards	4	0.37%
			Recovery / Exception	10	0.93%	Installability	1	0.09%
			Test interaction	4	0.37%			
			Test variation	3	0.28%			
			Start-up/Restart	2	0.19%			
			Concurrency	1	0.09%			
			White box path coverage	1	0.09%			
Total	**1070**	**100%**	**Total**	**1070**	**100%**	**Total**	**1070**	**100%**

The ‘Documentation’ defect type represents now almost half of the defects and ‘Function/Class/Object’ represents almost 20% of the defects. This can be justified by the fact that critical systems highly depend on documentation and documented evidences to prove the accomplishment of requirements and standards and to ensure qualification/certification of the systems by external entities. Furthermore, some of the defects not classified in the first round (with the original ODC taxonomy) have now been classified as ‘Documentation’ type, and those who were supposed to be of ‘Understandability’ defect type are also classified as ‘Documentation’ due to the merge operated in the enhanced ODC taxonomy. ‘Function/Class/Object’ identifies functionality implementation deficiencies, especially at implementation level.

For the defect triggers the results are a bit different from the previous as ‘Traceability’ has been clearly identified as a trigger, ‘Traceability/Compatibility’ became the most frequent trigger (28.88%) as it is also one of the most used defect finding activity for critical systems, the ‘Test Coverage’ (21.21%) is similar to the previous classification, and ‘Consistency/Completeness’ (19.25%) is still a quite high trigger due to the fact that a lot of artefacts under analysis are documents or documented that require consistency and completeness checks.

For the defects impacts 'Capability' (28.79%), 'Maintenance' (24.67%), 'Reliability' (23.55%) and 'Documentation' (14.67%) still represent over 90% of the impacts altogether.

Next, we present a detailed analysis of the classification results, providing a summary of the main findings. As mentioned before, this gives the first hints about the dataset, which can provide some quick feedback to the implementation and V&V teams.

5.4.1 Defect Type Results

The defect type is classified according to the fix that will remove it. If the defect has already been fixed (in the moment we did the analysis), then it is quite straight forward to determine its type. As observed in Table 17, there are 8 different types of defects, from the most frequent (Documentation) with 48.13% of the cases to the least frequent (Timing/Serialization) with only 3.08%. However, for the root cause analysis, we focus in every defect type, even the least frequent, since any failure can compromise a mission, with severe consequences. Table 18 presents the relation of every defect type with the classified impacts.

Table 18: Specific Impact distribution for every defect type

IMPACT	Documentation	Function / Class / Object	Algorithm / Method	Checking	Interface	Build / Package / Environment	Assignment / Initialization	Timing / Serialization
Capability	21.56%	51.24%	46.88%	8.82%	26.79%	16.00%	20.45%	43.75%
Reliability	13.97%	24.88%	33.33%	70.59%	23.21%	38.00%	29.55%	21.88%
Maintenance	32.14%	17.41%	11.46%	8.82%	28.57%	38.00%	27.27%	12.50%
Documentation	30.14%	0.50%	1.04%	0.00%	1.79%	4.00%	0.00%	3.13%
Performance	1.60%	2.99%	6.25%	10.29%	0.00%	2.00%	15.91%	12.50%
Usability	0.60%	2.99%	1.04%	1.47%	19.64%	2.00%	6.82%	6.25%

The following paragraphs present an analysis for each type of defect (latter in the thesis we will identify their primary root causes):

- **Documentation (48.13%)**: Documentation is an essential asset for these systems and is mandatory according to the standards (ECSS). It represents essential artefacts for the system implementation that are passed from phase to phase, starting from system specification and finishing with acceptance, operation and maintenance. It is important to highlight the defects in documentation that have an impact in Maintenance – 32.14% (quite important for space systems due to frequent changes and corrections that are required when the spacecraft is already in orbit, such as patches and dumps), Capability – 21.56%, and Reliability – 13.87% (which represent essential properties of

space systems: the correct implementation of functional and non-functional properties).

- **Function/Class/Object (18.97%)**: This type represents mainly the changes that need to be applied in system functionality to correct non-compliances with requirements. In short, it is the defect type that represents implementation problems. The analysis shows that Function defects are related with the Capability of the system – 51.24% (this is absolutely natural, since the defect type indicates a functional error), Reliability – 24.88% (many functionality problems have consequences in the reliability of critical systems), and Maintenance – 17.41% (autonomous and dynamic systems frequently require remote corrections and updates).
- **Algorithm/Method (8.97%)**: These defects are usually the result of efficiency or correctness problems that affect the task and can be fixed by (re)implementing an algorithm or a method. They are also related to system Capability – 46.88%, Reliability – 33.33%, and Maintenance – 11.46%.
- **Checking (6.45%)**: These defects are the result of the omission or incorrect validation of parameters or data, usually in conditional statements. Checking defects are related to Reliability – 70.59%, Performance – 10.29%, Capability – 8.82%, and Maintenance – 8.82%. Since Reliability is mostly achieved through Fault Detection Isolation and Recovery (FDIR), error checking and redundancy, checking defects have a profound impact on Reliability.
- **Interface (5.23%)**: These defects are the result of communication problems between subsystems, modules, components, operating system, or device drivers, requiring changes, for example, to macros, call statements, control blocks, parameter lists, or shared memory. Interface defects relate to Maintenance – 28.57%, Capability – 26.79%, Reliability – 23.21%, and Usability – 19.64%. The latter is mostly due to the fact that interfaces are related to operation, control and usability of the system.
- **Build/Package/Environment (4.86%)**: These defects are the result of problems on the build process and on change management and version control. They relate to Maintenance – 38.00%, Reliability – 38.00%, and Capability – 16.00%. The processes and tools used for these systems have a significant impact in such activities.
- **Assignment/Initialization (4.30%)**: Assignment/Initialization defects are the result of values not assigned or incorrectly assigned, as well as wrong initializations. These defects relate to Reliability – 29.55%, Maintenance – 27.27%, Capability – 20.45%, and Performance – 15.91%. The main impact is in the system reliability, as some of these defects might stay undetected until the occurrence of some very specific functional or non-functional situations.
- **Timing/Serialization (3.08%)**: These defects are the result of timing errors between systems, modules or components, or problems accessing shared

resources. They relate mainly with Capability – 43.75% and Reliability – 21.88%, as well as Performance – 12.50% and Maintenance – 12.50%.

5.4.2 Defect Trigger Results

Some of the root causes for the defects detected during ISVV are related with problems in the efficiency of the verification and validation activities applied during development. When we look at the triggers of the enhanced ODC, we might question about the reason why those defects have not been caught earlier by the development or V&V teams. Some questions arise in this case: has the independence so much importance that makes it easier to find defects? why the same techniques used by the ISVV team were not applied at an earlier stage of the project? or have they been ineffectively applied?

This section provides an analysis of the triggers that detected the 1070 defects at the ISVV stage of the projects. Since the list of triggers is extensive, we will focus on the most relevant in terms of number of detected defects. This analysis contributes to pinpoint the weaknesses of the regular development V&V activities and provide suggestions to change/improve V&V activities to become more efficient and detect more problems before they are passed on to the ISVV teams (this will be done specifically in Chapter 6). Table 19 summarizes the impacts of the defects identified by specific triggers.

Table 19: Specific Impact distribution for every defect trigger

IMPACT	Traceability/ Compatibility	Test Coverage	Consistency/ Completeness	Logic/Flow	Design Conformance
Capability	72.28%	25.49%	95.00%	18.95%	35.64%
Reliability	14.98%	37.75%	2.00%	21.05%	36.63%
Maintenance	4.12%	24.51%	1.50%	14.74%	9.90%
Documentation	0.75%	2.45%	0.00%	42.11%	15.84%
Performance	7.87%	9.80%	1.50%	3.16%	1.98%

The following paragraphs present an analysis for each defect trigger:

- **Traceability/Compatibility (28.88%)**: The Traceability/Compatibility trigger allows the detection of almost one third of all the defects. Such activities look for inconsistencies of information across phases, incomplete or outdated traceability matrices, or untraced artefacts. Regular V&V activities should be enough to detect these defects during the development. Apparently, a clearly stated and formally implemented traceability analysis is required and, if supported by the appropriate toolset, it would not require significant additional effort to the development and V&V teams. This trigger identifies most Capability defects (72.28%) and some Reliability ones (14.98%).

- **Test Coverage (21.21%)**: Test Coverage is the trigger that evaluates the completeness of the tests performed to validate the different units or functionalities by considering different values or possibilities. As expected, a significant number of defects is uncovered by Test Coverage activities. A significant number of Reliability (37.75%) impact defects are uncovered, but also Maintenance (25.51%) and Capability (25.49%) ones.
- **Consistency/Completeness (19.25%)**: Another trigger with significant impact related to inspections and reviews is Consistency/Completeness. All project artefacts must be consistent between each other, as well as complete. Due to the number of artefacts required by the lifecycle process and by the applicable standards (ECSS), consistency and completeness analysis of these artefacts very frequently reveal discrepancies. This trigger finds mostly defects with impact on Capability (95.00%).
- **Logic/Flow (11.12%)**: Logic and Flow analysis are triggers that allow the detection of control and data flow defects by assessing the logical paths of the software for program flow and control but also for data and variables flow. This trigger finds mostly defects with impact on Documentation (42.11%).
- **Design Conformance (11.12%)**: The Design Conformance trigger is a specific set of activities applied to the architecture that allows the review of design artefacts versus the specifications and the environment constraints. This trigger finds mostly Reliability (36.63%) and Capability (35.64%) related defects.

5.4.3 Defect Impact Results

The results of the defect impacts are a reflection of the severity of the defects even if the majority will never have a real impact because they will be solved and prevented before. The distributions of the impacts include Capability, Maintenance and Reliability as the most common ones, then Documentation with still a significant frequency, and finally Performance, Usability, Requirements, Migration, Installability and standards, with a much lower frequency.

The ODC Impact analysis can be used to prioritize the defect types/triggers to identify the development and V&V activities that might conduct to the defects with a high impact in the system. As “high impact”, we consider equally the impacts in Capability, Reliability, and Maintenance, as they are the most severe since they represent three essential requirements of critical space systems: functional quality, non-functional reliability assurance, and maintainability. For our analysis, we have considered Capability, Reliability and Maintenance as the most important impact types.

The following paragraphs present an analysis for each defect impact:

- **Capability (28.79%)**: It is normal that Capability (i.e. functionality) is the most affected property but, in space critical systems, maintenance has a significant importance as well as the reliability requirements. Capability represents the

defects that will affect the functionality of the system without having an impact on the non-functional properties, these can be seen as the normal "bugs" that will lead the system to work with limitations or not properly, due to wrong design or wrong implementation affecting the normal functioning.

- **Maintenance (24.67%)**: A large amount of the maintenance defects from an impact perspective) are originated in the implementation phase (most source code and source code documentation), thus, the larger the amount of source code defects the larger this impact will be, as it affects future maintenance of the source code itself.
- **Reliability (23.55%)**: The defects that have an impact on Reliability are extremely important as they represent problems that affect one of the essential properties of safety critical systems (not necessary or not always functional property). The nature of the systems and defects selected for this work lead to this high value, as some of the failures could have more than a Capability (or functional) impact and could affect the dependability properties such as Reliability and Safety that is also considered to be added as an impact for future usage of the enhanced ODC. Safety has not been added to the enhanced ODC taxonomy because the original dataset of defects did not lead to safety impact, but this was considered during the analysis of the enhanced ODC classification results as a possibility for future datasets. The same situation may apply to security-related defects.
- **Documentation (14.67%)**: As in all systems that heavily depend on documentation, there are some minor defects that will impact simply the documentation. This is the case for critical systems where documentation is required at every lifecycle phase as artefacts to specify, design or document evidences and results of the process.
- **Performance (3.64%)**: Impact of defects that affect the system performance, namely defects that increase the CPU load, the memory utilization, or increase the application timings due to extensive or unintended data or control flows originated from the defect.
- **Usability (2.62%)**: These critical systems have a rather low set of defects that impact Usability, but they still do exist. Other systems will certainly have larger percentages for this impact, in particular systems with user interfaces or command and control consoles. Our dataset was mostly of embedded automated and autonomous systems, thus this value is quite low.
- **Requirements (0.84%)**: Some of the defects affected the requirements and their specification. This can happen up to the validation and testing phases, but these are quite rare cases for critical systems due to the care that is taken to have mature and stable sets of requirements and milestones to review extensively those specifications.
- **Migration (0.75%)**: This is a rare impact identified when a system or a software is reused in a different environment, migrated to different hardware or configured in a different manner.

- **Standards (0.37%)**: Some impacts are related to the application or interpretation of standards during the system development lifecycle. There are several levels of standards that can be involved, namely the domain specific standards (e.g., the ECSS), the generic level standards (the Quality Assurance standards or ISO 9001) and the specific engineering standards (requirements standards, coding standards). Defects with an impact on standards can provide feedback to improve the standards or make them more precise.
- **Installability (0.09%)**: Impact originated from defects that affect the installation, the configuration or the modification for safe use of the systems.

5.4.4 Combined Results

The previous sections have presented the main results from the defect type, defect trigger and defect impact points of view. These results provide a good overview of the system quality and already indicate some of the weaknesses that need to be tackled to make the systems better and to avoid or prevent either most of the defects or the defects with more severe impacts. Next, we will provide a more integrated view of the defects that have slipped through review and defect detection phases of the lifecycle, and show how the results presented before integrate as a whole.

5.4.4.1 Defect Introduction vs Detection Phase

To support the RCA, we have analyzed the introduction versus detection phases of the defects. If a defect is not detected during, or right after, the phase when it was inserted, that means that the V&V or defect detection activities between at least two different phases failed. Table 20 presents the ISVV activities that detected the defects introduced in previous stages and that slipped through detection on at least one defect detection activity. The top heading ("Phase of Introduction") represents the development phase when the defects have been introduced, while the first vertical column contains the activities (and phases) when the defects have been actually detected. Each row contains the number of defects that have been detected in the phase specified in the first column and that have been originated in the phase specified in the first row of the table (Requirements, Design, etc.).

Table 20: Phase of introduction versus phase of detection

	Phase of Introduction	Requirements	Design	Implementation	UT/IT	System Tests	Operation	Total late detected	Total
Phase of Detection									
Requirements Verification		162						-	162
Design Verification		6	106					6	112
Implementation Verification		10	77	290				87	377
UT/IT Verification		18	0	9	351			27	378
System Tests Verification		2	0	0	9	10		11	21
Operation Monitoring		1	6	0	0	9	4	16	20
Total late detected		37	83	9	9	9	-	147	-
Total		199	189	299	360	19	4	-	1070

The large majority of defects (86%) were detected right after being introduced (shaded diagonal). However, a large number of defects (147) escaped both V&V and ISVV (light blue background), being caught by later ISVV activities only. In Table 20 we divided the testing activities in 2 phases as they provide an additional view showing that even within the testing activities there are defects that could be caught earlier.

We can observe that a relevant number of defects that escaped previous ISVV or V&V activities were detected during Implementation Verification (60%) – 10 defects were introduced during Requirements and 77 during Design. A closer look, shown in Table 21, reveals that almost 80% are Documentation defects and 10% are Function defects, which is in-line with the overall results presented earlier in this chapter. Thus, tackling documentation issues might greatly reduce defect propagation.

The most important observation from the results in Table 21 is related with defect triggers. Traceability/Compatibility accounts for 60% of the detected defects, while Design Conformance and Consistency/Completeness account for 20% each. As source code is more detailed and more concrete than architectural design components and requirements descriptions, it is normal that traces can detect more inconsistencies, especially missing or incoherent information.

The defects detected during the testing phases that originated from previous phases include 20 defects injected during requirements specification and 9 defects introduced during implementation, from which 14 defects are Function/Class/Object defects and 10 are Documentation defects.

Table 21: Defects detected late, after Implementation

Defect Type	Design	Requirements
Documentation	65	4
Function/Class/Object	7	2
Algorithm/Method	2	3
Checking	1	0
Interface	1	0
Timing/Serialization	1	1
Defect Trigger	**Design**	**Requirements**
Traceability/Compatibility	46	5
Design Conformance	16	2
Consistency/Completeness	15	3

5.4.4.2 Enhanced ODC Defect Type vs Impact Analysis

The defect types that have higher impacts in the system (affecting Capability, Reliability and Maintenance) are depicted in Figure 12. Defects with impact in Capability (blue line) are mainly related with Function/Class/Object, Documentation and Algorithm/Method types, confirming that the functionality specification/implementation, the documented artefacts and the design decision in what concerns algorithms and methods to apply are the main contributors to defects that influence the system capability and normal functionality.

Defects with impact in Reliability (orange line) are originated mostly from Documentation, Checking, Function/Class/Object and Algorithm/Method defect types. In this case, there is a new defect type that contributes significantly to reliability issues: Checking. It is clear that reliability (including redundancy, fault detection/monitoring, isolation and recovery) is often implemented with checks and verifications for monitoring and detection of errors and so the importance of avoiding this type of defects to guarantee higher reliability.

Defects with impact in Maintenance (gray line) originate essentially from the Documentation defect type. This is an expected result due to the fact that maintenance depends on source code documentation and comments and documented artefacts that include installation and download instructions, user and developer manuals, and maintenance procedures.

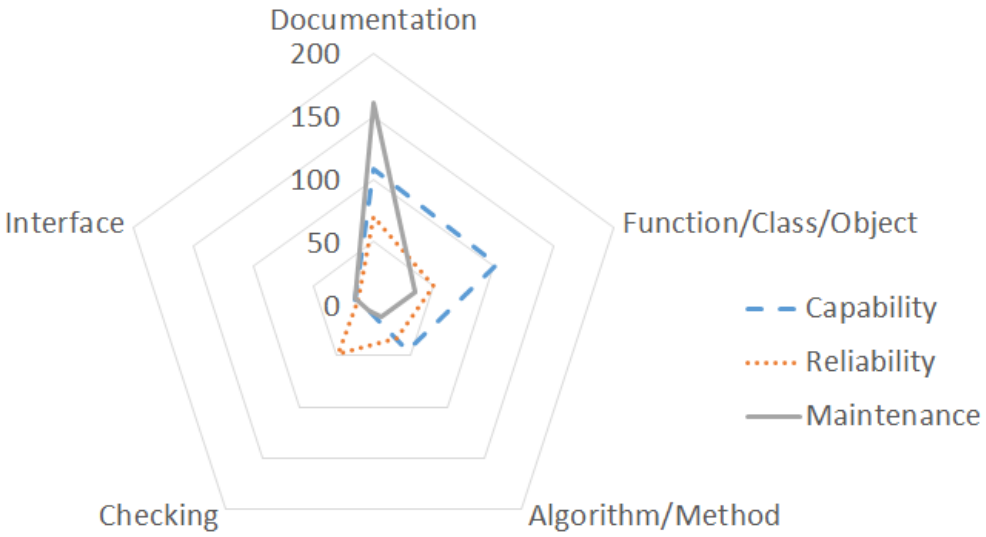


Figure 12: Defect type versus defect impact

Table 22 summarizes the defects that have a high impact in the system (regarding Capability, Reliability and Maintenance). The defect types are ordered according to the total number of defects with such impacts. Table 22 shows that the two most frequent defect types (Documentation and Function/Class/Object) account for almost 50% of all defects (64% regarding the defects with high impact). Actions to avoid these defects, such as Technical Writing trainings, improvement of documentation reviews or automation of verification of documentation issues, could significantly reduce the number of defects and improve Capability, Reliability and Maintenance.

Table 22: Defect types with high impact (Capability, Reliability and Maintenance)

Type	Capability	Reliability	Maintenance	Total defects	% overall defects	% defects with high impact
1. Documentation	108	70	161	339	31.7%	41.2%
2. Function/Class/Object	103	50	35	188	17.6%	22.9%
3. Algorithm/Method	45	32	11	88	8.2%	10.7%
4. Checking	6	48	6	60	5.6%	7.3%
5. Build/Package/Environment	8	19	19	46	4.3%	5.6%
6. Interface	15	13	16	44	4.1%	5.3%
7. Assignment/Initialization	9	13	12	34	3.2%	4.2%
8. Timing/Serialization	14	7	4	25	2.3%	3.0%
Total	308	252	264	824	77.0%	100%

5.4.4.3 Enhanced ODC Defect Trigger vs Impact Analysis

The defect triggers that allow the detection of the defects with a high impact are represented in Figure 13. The graph reinforces the importance of the three main triggers as the most important (frequent) triggers (overall they allowed the detection of 77.0% of the issues): a) Consistency/Completeness, b) Test Coverage, and c) Traceability/Compatibility. For this particular case, Reliability can be ensured with better Traceability/Compatibility analysis, Test Coverage and Logic/Flow analysis. Capability shall be assessed more efficiently with Test Coverage, Traceability/Compatibility assessment and Design Conformance Analysis. The Maintenance defect impact can be mitigated with Traceability/Compatibility and Consistency/Completeness analysis.

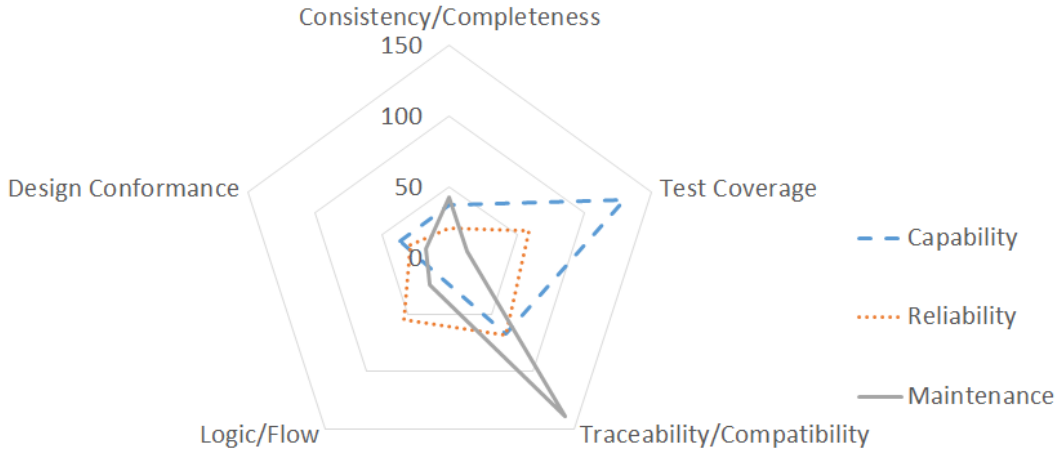


Figure 13: Defect triggers versus defect impacts

Table 23 prioritizes the triggers that detected the defects with high impact. We can observe that the two triggers that detect the most impacting defects are Consistency/Completeness analysis and Test coverage, allowing the detection of 57.8% of the high impact defects (44.5% of all defects). Although the list of triggers that enable detection of high impact defects is extensive, the 5 more meaningful ones allowed the detection of 91.6% of the defects with high impact (about 70% of all defects in our dataset). This is the reason why it is efficient to focus on the top 5 triggers, as shown in Table 23: Consistency/Completeness, Test Coverage, Traceability /Compatibility, Design Conformance and Logic/Flow.

Table 23: Defect triggers with high impact

Trigger	Capability	Reliability	Maintenance	Total defects	% overall defects	% defects with high impact
Consistency/Completeness	67	68	139	274	25.6%	33.3%
Test coverage	130	59	13	202	18.9%	24.5%
Traceability/Compatibility	37	20	42	99	9.3%	12.1%
Design conformance	15	55	24	94	8.8%	11.4%
Logic/Flow	37	29	18	84	7.9%	10.3%
Rare situation	8	10	6	24	2.2%	2.9%
Test sequencing	4	5	6	15	1.4%	1.8%
HW/SW Configuration	3	0	6	9	0.8%	1.0%
Standards conformance	0	2	6	8	0.7%	0.9%
Recovery/Exception	4	1	3	8	0.7%	0.9%
Test interaction	2	0	1	3	0.3%	0.4%
Test Variation	0	2	0	2	0.2%	0.3%
Concurrency	0	1	0	1	0.1%	0.1%
Path coverage	1	0	0	1	0.1%	0.1%
Total	308	252	264	824	77.0%	100%

5.4.4.4 Enhanced ODC Defect Trigger vs Type Analysis

It is also interesting to understand which defect triggers lead to the detection of which defect types. This is what is shown in Figure 14 for the three most frequent defect types. The graph shows that defect triggers Traceability/Compatibility and Consistency/Completion allow the detection of a very large number of Documentation defects (dashed blue line). As these triggers apply mostly to documentation and documented artefact, this is an expected result. Another observation we can make is that the Test Coverage trigger allows mostly the detection of Function/Class/Object defects. In fact, the objective of Test coverage is to cover/test essentially the specifications, which represent the functionality of the system. Thus, if the specifications are not totally covered, functionality defects are probable to be missed (Function/Class/Object). The same trigger (Test Coverage) also allows the detection of Algorithm/Method defects in a larger scale than the other triggers.

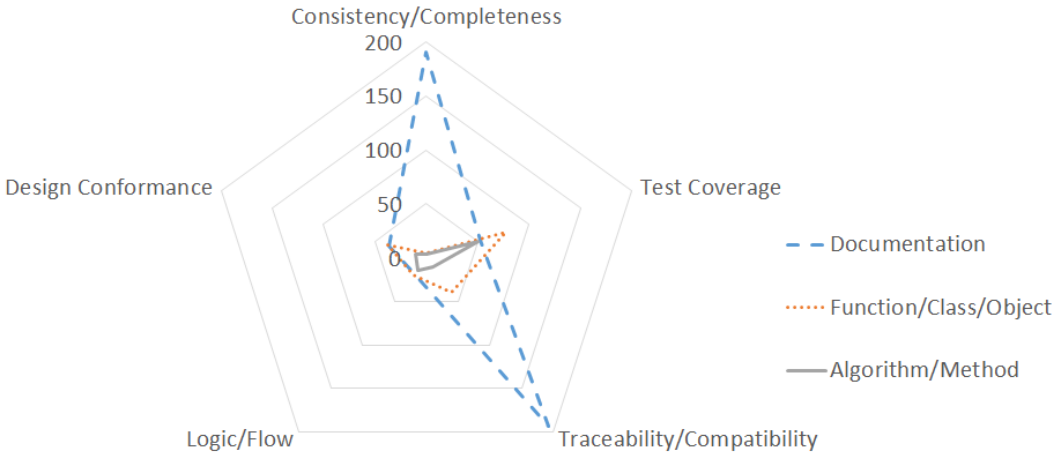


Figure 14: Defect triggers versus defect types

From the overall distribution of which triggers detect the 5 most frequent defect types (Table 24) we can observe that some triggers are more efficient in detecting some defect types, such as Design Conformance and Logic/Flow, which are good to detect Checking defects, and the Test Coverage and Traceability/Compatibility triggers that detect most of the Interface defects.

Table 24: Defect triggers detecting specific defect types

Trigger	Documentation	Function/ Class/Object	Algorithm/Method	Checking	Interface	Total defects	% overall defects
Consistency/Completeness	190	4	3		3	200	18.69 %
Test coverage	52	77	50	5	20	204	19.07 %
Traceability/Compatibility	193	40	11	2	21	267	24.95 %
Design conformance	36	37	10	16	2	101	9.44%
Logic/Flow	18	20	14	40	3	95	8.88%
Rare situation	6	12	1	3		22	2.06%
Test sequencing	2	4	4		5	15	1.40%
HW/SW Configuration	2		2		2	6	0.56%
Standards conformance	7	1		2		10	0.93%
Recovery/Exception	3	6		1		10	0.93%
Test interaction	3		1			4	0.37%
Test Variation	3					3	0.28%
Concurrency		1				1	0.09%
Path coverage		1				1	0.09%
Total	515	203	96	69	56	939	87.76 %

5.5 Validation of the Enhanced ODC

As mentioned, to validate the enhanced ODC taxonomy we re-applied it to our dataset. In addition to the important fact that no issue was left unclassified with this new taxonomy and that most of the issues could be classified in an easier way (avoiding confusions or doubts by merging similar taxonomy elements), the results highlighted the following:

a) The results with the enhanced ODC taxonomy revealed a higher percentage of '*Documentation*' when compared to the original ODC classification. This can be justified by the fact that critical systems highly depend on documentation and documented evidences to prove the accomplishment of the requirements and the standards. Furthermore, the classification performed by using the enhanced ODC taxonomy allowed to consider 339 new defects, most of them previously classified with the "Documentation" type, but that could not be added to the results since either trigger of impact where not correctly classified.

b) '*Traceability/Compatibility*' is the more frequent trigger and even '*Test Coverage*' became a trigger more efficient than '*Consistency and Completeness*'. This suggests that the most efficient defect triggers are the simplest and more logical ones, namely the ones related to traceability and testing activities.

c) The '*Maintenance*' defect impact became more frequent than '*Reliability*', and the '*Documentation*' impact frequency has been reduced. In fact, maintainability is an important property for the systems in our dataset, and the results show that issues impact more system maintainability than system reliability.

The results also show that the 5 more frequent types, triggers and impacts cover about 90% of the issues analyzed. This does not mean that the others are not important (in fact, ODC does not take into account the issue severity in its classification scheme) or that they do not need careful analysis, but, with up to 5 taxonomy values, we are covering the large majority of the issues, which suggests that actions can be taken to quickly improve the quality of the systems, which is of extreme importance for the industry.

These results, however, have not been compiled easily. The effort spent for the 1070 classifications was around 800 man hours (only for the classification task). This effort includes the original ODC classification that exposed issues with the classification of 31.7% of the defects, the effort of enhancing the ODC attributes values, and the effort of reclassification. The main noted difficulties during the whole process are related to:

- **The amount of ODC classifications possible** – even though only 3 ODC attributes have been selected for the classification effort, the number of possibilities is still significant – we found the standard taxonomy of the selected attributes a bit generic when applied to a safety critical domain;
- **The lack of fit of the ODC taxonomy for critical systems issues** – this lead to additional effort to try to classify the attribute originally, a need to scan and check all attributes possibilities and consequent rework;
- **The precision, completeness and detail level of the defects description** – some RIDs are very telegraphic, some others are extremely technical, some include very limited information or lots of references that need to be checked to perform the correct classifications;
- **The lack of uniformity in the description of the defects** - due to the fact that they had been compiled between 2005 and 2014, by different teams of engineers and they related to different types of systems or subsystems.
- **The amount of supporting documentation required** – the classification required often the reference to the original documentation (specifications, architecture, source code, testing artefacts, etc.) and these artefacts are quite extensive, exist in different versions, had to be recovered from the projects archives, and so on.

Table 25 shows that the second round was about twice as fast as the first round with the original ODC (**3.9 defects/hour** versus **2.1 defects/hour**). This is naturally due to the fact that the defects were already known, the ODC taxonomy was clearer and the practice of the analysts had increased. Another exercise, taken later, to classify 120 issues from railway control and management systems support the validation of the enhanced taxonomy by two facts: a) the 120 defects have all been classified smoothly;

and b) the effort spent in that classification (defect type, trigger and impact) was 34 hours (**3.5 defects/hour**).

Table 25: Effort Spent for the different ODC related activities

Activity	Description	Effort (hrs)
Data preparation and training	Data collection/clean-up, training. 2 expert engineers in ISVV and Critical systems and one junior researcher.	110
ODC phase I	Classification of defects and review of the classification. Full classification of 721 defects, remaining 349 defects have been partially/doubtfully classified. 2 expert engineers in ISVV and Critical systems and one junior researcher.	520
Analysis/ Proposal of ODC Adaptations	Analysis of the 349 defects have been partially/doubtfully classified in the previous phase and identification of modifications to the original ODC taxonomy. 2 expert engineers.	80
ODC phase II	Reclassification and review of the defects (about 33%). The remaining 349 defects have been classified and reviewed. 2 expert engineers.	90
Total		**800**

5.6 Final Remarks

This chapter presented the defects classification results prior to the root cause analysis. Both results including the original ODC classification and the enhanced ODC classification were presented, but only detailed results of the enhanced ODC taxonomy have been described. The classification issues identified by the ISVV teams using ODC allowed the classification of 739 issues (out of 1070 defects). The remaining 31.7% could not be classified and required an improved, more applicable taxonomy, which was proposed and applied.

The proposed adjustments to the ODC taxonomy had several objectives, namely: to promote a fit for critical systems issues classification and study, to maintain the orthogonality of the classifications, to propose only the minimum amount of changes possible to the ODC taxonomy, to simplify the classification work, and to allow easy root-cause analysis for the future. The process to determine the adaptations was based on the missing classification for the ODC defect Type, Trigger and Impacts as explained before, but also on the difficulty to classify some of the issues according to these attributes, thus including: a) new types, triggers and impacts; b) merged types, triggers and impacts; and c) adjustment of some previous classifications due to a better interpretation of the attributes. While the *activity* attribute was updated generically to be adjusted to the commonly used lifecycle phases, the other three attributes (type, trigger and impact) suffered some enhancements to improve the classification and reduce the amount of doubts while classifying the defects.

The results presented can help the space industrial community in focusing on the weakest points of the engineering process to improve them. Also, by using the enhanced ODC, ISVV teams can work much more efficiently with the triggers that catch more problems and even develop appropriate and more precise V&V tools or defect detection processes. As the systems involved cover most of the development activities performed for those systems, and involve different companies (at geographic, size and management level), we consider these results to be quite general for this domain. A similar study for other domains (e.g. aeronautics, railway, automotive) is foreseen as future work, but it will not be as easy as the existing data might not be as structured as for space systems. Data confidentiality will be a challenging issue.

The enhanced ODC classification is done based on the opinion and knowledge of experts and not following precise algorithms or criteria. However, it is important to note that the original classification (the one that could not classify all the issues) was performed by two engineers, whose work was also checked by a third space domain expert. This domain expert also performed the reclassification himself (verified and discussed with another space domain expert engineer in the case of doubts).

Finally, the results present interesting data to support the root cause analysis, and also for immediate feedback to the engineering of critical systems, namely, what are the most frequent defect types, and how they are reflected in terms of impact (severity), what are the most efficient and frequent defect triggers (and again how they detect issues with specific impacts), what type of triggers allow the detection of specific defect types, and what are the defects that have slipped between lifecycle phases without being detected (in order to identify if it is possible to detect them faster in the near future).

Chapter 6
Defects Root Cause Analysis

"There are a thousand hacking at the branches of evil for one who is striking at the root, and it may be that he who bestows the largest amount of time and money on the needy is doing the most by his mode of life to produce that misery which he strives in vain to relieve." – Henry David Thoreau, 1854.

This chapter presents the root cause analysis process and the results (root causes and suggested measures) of the enhanced ODC application to the dataset of 1070 defects from space projects and identifies the root causes for the majority of the defects based on the ODC results and the knowledge of the space domain environment, processes, methods and tools. As it is not possible to identify the root cause for every single defect, we focused on the more frequent and more severe defect types.

The presentation of results and root causes is divided in five main lines of analysis: a) the enhanced ODC results (the inputs to the RCA); b) the analysis of the defect types (eventually detects implementation problems); c) the analysis of the defect triggers (identifies inefficient V&V activities); d) the analysis of the defects identified at a later SDP phase (inefficient ISVV or V&V activities); and e) the prioritization according to the defect impacts (Capability, Reliability and Maintainability). Note that the root causes have not been identified at the moment of resolution of the issues but at the moment of the analysis of the enhanced ODC results. In practice, they are the result of an expert analysis done on the defects, performed by the authors and complemented and reviewed by the industrial partners. Note also that several defects do not have a clear and unique root cause but a set of related root causes.

The root causes analysis consisted in a structured process based on a fishbone analysis for the most frequent defect types and defect triggers, and a specific root cause analysis for the defects that have slipped from detection in the phase where they have been introduced. We have first analyzed the resulting defect types and identified root causes related to development issues, then we have identified the main defect issues according to defect triggers which gave us the V&V weaknesses. In a subsequent step, we have proposed a dedicated list of measures to tackle both sets of root causes.

6.1 Overview of the Process

A fundamental law in science is the *Law of Cause and Effect*: it states that every effect has a cause. From that law follows the *Law of Root Causes*: stating that all problems arise from their root causes. This is then the fundamental principle of *Root Cause Analysis*, which intends to tackle the root of a problem by finding and resolving its root causes. Root cause analysis is a kind of problem solving method aimed at identifying the root causes of problems or events. It is common belief that problems are best solved by correcting or eliminating their root causes, as opposed to merely addressing the immediately obvious symptoms, which is common industrial practice.

Root cause analysis also helps on the identification of why an issue or problem occurred and promotes the creation of a knowledge base that can be used to prevent or reduce the impact of root causes in the future. When a root cause is permanently or completely eliminated or controlled, then immediate or remedial rework is avoided and the future occurrence of the issues caused by the root cause tackled are proactively addressed (or avoided).

In the frame of this work, the root cause analysis has been designed to fit the process described in Chapter 3 and to rely on defect classification from three different perspectives: *a)* identification of root causes taking into account the more frequent or more severe defect types; *b)* identification of root causes based on the more frequent or more efficient defect triggers; and *c)* identification of root causes applicable to the late detection of the defects in the development (or V&V) lifecycle. With the combination of these three perspectives (see Figure 15) we can ensure a general but also embracing analysis that is certain to identify the more important root causes and help in improving the quality for future developments.

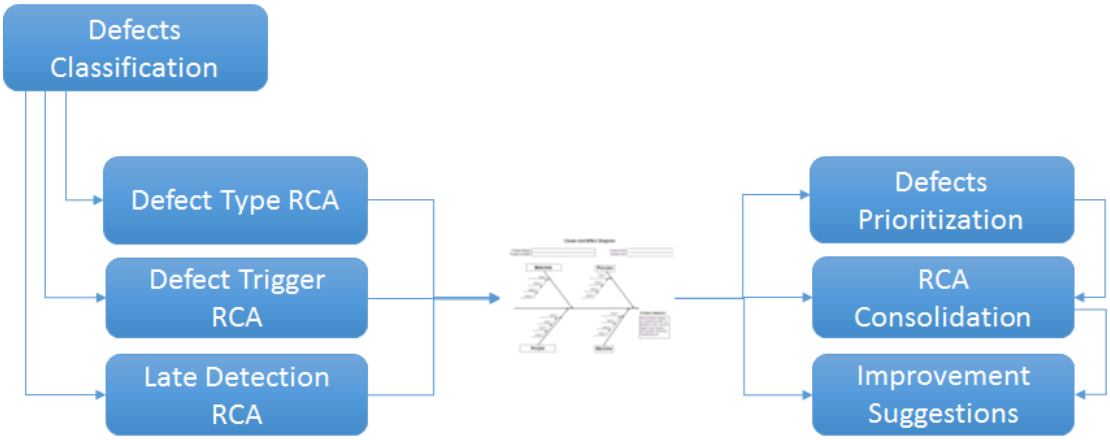


Figure 15: Root Cause Analysis Overview

Figure 15 provides a short overview of the root cause analysis as integrated in the Chapter 3 defects analysis process. With the support of data from the defects classification, three types of root causes are identified, not necessarily for every single defect, but more generally to groups of similar defects, by considering defect types, or

by identifying triggers and V&V techniques that detected the defects, or by peeling off the root causes of the late defect detection. Root causes can be identified by several ways and techniques as described later in this chapter. Once the list of root causes is defined, it can be prioritized based on different criteria (e.g. the amount of defects caused, the severity of the impact of the caused defects, or removal of specific types of undesired defects), consolidated by aggregating root causes common to the three different root cause identification perspectives (defect, type, defect trigger and late detection) or even similar root causes, and finally, complemented with a step of matching the root causes to real suggestions for avoiding or eliminating them. The improvement suggestions should be communicated to the engineering stakeholders, implemented and monitored in order to confirm that the root causes are eliminated/reduced for future development and V&V cycles.

For the root cause analysis, several techniques can be applied, as long as they are mastered and applied by experienced analysts. Examples of techniques include: Five Whys; Failure mode and effects analysis; Fishbone (cause and effects, Ishikawa) diagrams; SIPOC (Suppliers, inputs, processes, outputs, customers diagram); Flowcharting of the process flow, system flow, and data flow; Critical to quality metrics; Pareto chart; and Statistical Correlation. The most commonly used of these techniques are detailed in Section 2.4. Fishbone (cause and effects, Ishikawa) diagrams is the technique used in the remainder of this chapter.

6.2 Root Cause Analysis Results

This subsection presents the results obtained from the root cause analysis activities performed on the classified defects data. Root causes have been identified according to the grouped defect types and defect triggers. The same root causes have been also mapped to the defects that have not been detected within the phase they have been introduced.

6.2.1 Enhanced ODC Defect Type RCA

The defect type is classified according to the fix that removes it. If the defect has already been fixed (in the moment we did the analysis), then it is quite straight forward to determine its type, as information about the fix is available. The defects classification lead to the identification of 8 different types of defects (refer to Table 17), from the most frequent (Documentation) covering 48.1% of the cases to the least frequent (Timing/Serialization) covering only 3% of the defects. However, for the root cause analysis, we focus in every defect type, even the least frequent, since the number of defect types is acceptable and it is feasible to perform the analysis for 8 different types. This way, the following subsections present an analysis for each type of defect, and identify their primary root causes, based on the knowledge of several experts with many years in development, V&V and ISVV activities in the space domain.

It is important to note that the root cause analysis was not performed on the individual defects (due to the amount of work that that would require), but sampling of defects have been used to confirm the applicability of the defined root causes.

Table 26 presents the summarized and harmonized root causes that apply for each defect type. These root causes represent the most common causes of the defects of each type. Some examples from our dataset are also presented (the defect title is simple enough to be understood without a detailed description). For example, looking at the "Checking" defect type, the root causes identified include ambiguous/missing/incorrect architecture and design artefacts; incomplete specifications in what concerns FDIR and erroneous situations (commonly non-functional specifications), etc. For this case, several situations where erroneous situations can occur have not been taken into account, such as the examples presented in Table 26: missing validation of input parameters and index value not checked.

Table 26: Root Causes vs Defect Types with real examples

Defect Types	Root Causes	Examples of Defects
Documentation	lack of basic documentation skills (e.g. technical writing); oversimplified documentation planning procedures; lack of time to produce, review and accept documentation artefacts (pressure on schedules); lack of importance given to some documentation artefacts (prioritization); lack of completeness and consistency of documentation in previous phases (ambiguous information, missing information, incomplete documents); limited domain knowledge (understanding of the system); simplification of the product assurance processes related to documentation artefacts; and limitations of the tools or toolsets that deal with documentation, especially across development lifecycle phases or lack of preparation to use such tools.	- Insufficient information to validate implementation - Code documentation not consistent with implementation and the design - Requirement covered by test procedure not indicated in test specification
Function/Class/Object	ambiguous/missing/incorrect artefacts (documentation, requirements, design, tests); inefficient/insufficient reviews; limited engineers' domain knowledge – lack of appropriate skills; lack of system knowledge (to understand the overall functionalities); lack of tools knowledge, programming languages, design languages; and insufficient unitary tests.	- Mismatch between function documentation and design/implementation - Requirement not completely validated
Algorithm/Method	ambiguous/missing/incorrect artefacts (documentation, requirements, design, tests); inefficient/insufficient reviews; limited engineers' domain knowledge – lack of appropriate skills; lack of system knowledge (to understand the overall functionalities); lack of tools knowledge, programming languages, design languages; and insufficient unitary tests.	- Inconsistency between requirement and function implementation - Requirement may not be validated in test step
Checking	ambiguous/missing/incorrect architecture and design artefacts; incomplete specifications in what concerns FDIR and erroneous situations (commonly non-functional specifications), inefficient/insufficient reviews; insufficient/wrong tests (unit, integration, system, fault injection); lack of system knowledge; and lack of reliability and safety culture.	- Validate the input parameters before writing or reading from array - Index value not checked (Defensive programming).

Defect Types	Root Causes	Examples of Defects
Interface	ambiguous/missing/incorrect architecture and design artefacts; ambiguous/missing/incorrect Interface Control Documents (ICD) or protocols definition; incomplete specifications in what concerns interfaces, environment and communications; limited definition of the operation, usability, maintainability requirements (user, operation, installation manuals); inefficient/insufficient reviews; insufficient/wrong tests (unit, integration, system); and lack of system knowledge (interfaces).	- Incoherence between requirements, design and code - Unused input parameter
Build/Package/Environment	version and configuration management procedures inappropriately implemented; complexity of the build, versioning or change control procedures; complexity of the tools used for build, change or version control; and lack of knowledge on how to properly use the tools, build warnings/errors not resolved.	- Unused macros - MISRA C Violations: Functions not defined
Assignment/Initialization	lack of specification of initial and default values; incorrect implementation by forgetting basic initializations; and lack of checking of values and results of operations.	- Initialized variables not found in state, parameter or Housekeeping Data - Output pointer not validated
Timing/Serialization	lack of appropriate architecture detailing the timing properties; and lack of knowledge of subsystems, modules, resources, including hardware behavior.	- Housekeeping execution time is 400ms instead of 500ms - Transmission rate inconsistency between design and code

6.2.2 Enhanced ODC Defect Trigger RCA

Some of the root causes for the defects detected during ISVV reveal problems in the efficiency of the verification and validation activities applied during development. When we look at the triggers of the enhanced ODC (refer to Table 17), we might question the reason why those defects have not been caught earlier by the development or V&V teams, or simply by the application of those same triggers: has the independence so much importance that makes it easier to find these defects? why the same techniques used by the ISVV team were not applied at an earlier stage of the project? or have they been ineffectively applied?

This section provides a qualitative analysis of the triggers that detected the 1070 defects at the ISVV stage of the projects (see Table 17). Since the list of triggers is extensive, we focus on the most relevant in terms of number of detected defects. This analysis contributes to pinpoint the weaknesses of the regular development V&V activities and provide suggestions to change/improve V&V activities to become more efficient and detect more problems before they are passed on to the ISVV teams. From the identification of triggers that should had been there at the development/V&V activities we derive root causes in a simple manner.

This listing is presented in a detailed way since the triggers can be more directly and concretely affected by the removal of the root causes. This means that we can apply very specifically the recommendations and see an immediate effect, which is usually

the detection of defects at the moment of application of the trigger. We also present the title of a few defect examples from our dataset in order to demonstrate the type of defects uncovered by the specific trigger-related root causes.

6.2.2.1 Traceability/Compatibility

The identified generic root causes that lead to Traceability/Compatibility trigger not being effective before the ISVV detection are:

- **Lack of traceability verification culture** – most of the development / V&V engineering activities related to traceability do not use properly the traceability as the powerful tool it can be. Traceability can be used to support design, implementation and testing activities, as well as the reviews of the artefacts between lifecycle phases;
- **Lack or inefficient usage of tools that support traceability across lifecycle phases** – the traceability is usually performed either on very simple tabular format or integrated in other tools without automated/regular checks and validations, in particular when assessing the artefacts of a specific lifecycle phase.

Some examples of defects that could have been uncovered with appropriate application of the Traceability/Compatibility trigger include:

- Inconsistency between Function Comments, Design and Code;
- Inconsistencies between test procedure and test log;
- Traceability mismatch between Procedure and test specification.

6.2.2.2 Test Coverage

The identified generic root causes that lead to a Test Coverage trigger not being effective before the ISVV detection are:

- **Lack of appropriate test planning and test strategy** – test strategy and test planning are commonly misconceived as the activity of defining test procedures and test cases, implement, debug and execute them and then report the results. The test strategy is essential as it shall define the testing methodologies, the test planning, testing types, testing approaches, testing tools, test environment, test data strategy, staffing needs and trainings, and so on;
- **Lack of appropriate testing tools and testing environment support** – the testing tools and the testing environment are quite often deficient, archaic sometimes, and should provide confidence in the testing, as well as be able to automated and support the test implementation, execution and reporting activities;

- **Poor test specification and execution** – test procedures specification can also be limited due to the strategy and the available tools and environments. However, it is still quite difficult for testing engineers to master the art of defining appropriate and complete test specifications, while, for example using full traceabilities to ensure coverage of the requirements or design artefacts. Special care in the execution and logs collections is also required;
- **Insufficient testing** – as the test strategy is usually quite weak, the testing team end up not doing enough testing. This must be defined upfront and later on a case by case situation based on the particular needs of each requirements in terms of logic flow or data testing needs to achieve functional and non-functional coverage of "all" situations.

Some examples of defects that could have been uncovered with appropriate application of the Test Coverage trigger include:

- Requirement not completely validated;
- Incomplete list of requirements covered by the test;
- Test Steps with no clear validation goal (missing requirement association).

6.2.2.3 Consistency/Completeness

The identified generic root causes that lead to Consistency/Completeness trigger not being effective before the ISVV detection are:

- **Documentation related root causes** - lack of basic documentation skills (e.g. technical writing); oversimplified documentation planning procedures; lack of time to produce, review and accept documentation artefacts (pressure on schedules); lack of importance given to some documentation artefacts (prioritization); lack of completeness and consistency of documentation in previous phases (ambiguous information, missing information, incomplete documents); limited domain knowledge (understanding of the system); simplification of the product assurance processes related to documentation artefacts; and limitations of the tools or toolsets that deal with documentation, especially across development lifecycle phases or lack of preparation to use such tools;
- **Review process related root causes** – namely review simplifications (due to lack of time excuse) and inappropriate/no usage of traceability assessments;
- **Deficient usage of tools and applicable processes** – oversimplification of documentation processes, verification and validation processes and difficulty to accept comment on own' work. The usage of tools is also dependent on the experience and training acquired in the tool usage and tool features, so, for

several situations, a more appropriate usage and application of available tools would produce a great improvement;

- **Unclear or missing specifications** – specifications that do not describe clearly the requirements (specifications are not specific, measurable, attainable/achievable/actionable/appropriate, realistic, time-bound/timely/traceable) or missing specifications about some important steps that will be left to the designer or implementation teams;
- **Lack of domain knowledge** – the ignorance of some processes, functional and non-functional details about the system or the domain are some of the most common causes for consistency and completeness problems during all phases of the lifecycle.

Some examples of defects that could have been uncovered with appropriate application of the Consistency/Completeness trigger include:

- No results in the UT report;
- There is not enough information to validate implementation;
- Inconsistency with (/within) design.

6.2.2.4 Logic/Flow

The identified generic root causes that lead to Logic/Flow trigger not being effective before the ISVV detection are:

- **Incomplete specifications** – specifications that do not provide the means to clearly design the logic or flow of operations/actions or missing specifications about some important steps that will be left to the designer or implementation teams;
- **Ambiguous or unclear architecture definition** – incorrect or incomplete architecture definition have an effect on the implementation, and in particular in the interfaces between modules where commonly discrepancies and problems are identified;
- **Lack of usage of tools that support data and control flow analysis** – tools to analyze the systems from a logic or data flow perspective would avoid several problems related to interfaces or performance, tools similar to static code analysis tools would be very beneficial for the logic/data flow analysis of modern systems.

Some examples of defects that could have been uncovered with appropriate application of the Logic/Flow trigger include:

- Releasing semaphores that haven't been locked;

- Optimize the validation of the mode transition table;
- Possible incorrect tracking of failed thrusters.

6.2.2.5 Design Conformance

The identified generic root causes that lead to Design Conformance trigger not being effective before the ISVV detection are:

- **Inappropriate architecture support tools or tool usage** – tools that automate checks, in particular for the architecture/design, could support on the checking of conventions, completeness, interfaces and better assessment of the systems;
- **Deficient specification or design artefact that lead to wrong implementations** – these are the common "bugs" from a design perspective, if the design has flaws they might be replicated in the implementation. This problem is also related to the expertise and technical knowledge of the designer.

Some examples of defects that could have been uncovered with appropriate application of the Design Conformance trigger include:

- Code does not follow the design;
- Not validated state variable and documentation inconsistent with code;
- Cyclomatic complexity higher than expected.

6.2.3 Late Detection RCA

There are defects that for some reasons are not detected within the same phase they are introduced, and this might lead to a severe leakage of the defects over phases. In case a defect is detected later (it might not be detected) then the effort to fix it is much larger as several artifacts and several lifecycle phases need to be revisited.

It is then of utmost importance to determine why (the causes) certain defects have not been spotted and solved before, and why they have slipped through phases. The root cause analysis of these specific slipped defects helps in identifying specifically the causes of the failures in the V&V and ISVV techniques that allowed the defects to propagate until a later stage in the lifecycle without being spotted.

We have analyzed the introduction versus detection phases of the defects. If a defect is not detected during, or right after, the phase when it was inserted, that means that the V&V activities from at least 2 phases failed detecting it, and that the ISVV activities from at least one phase also failed. Table 27 presents the ISVV activities that detected the defects introduced in some of the previous stages. The top heading ("Phase of Introduction") represents the development phase when the defects have been

introduced, while the first vertical column contain the activities (and phases) when the defects have been actually detected. Each row contains the number of defects that have been detected in the phase specified in the first column and that have been originated in the phase specified in the first row of the table (Requirements, Design, Implementation, Unit Tests and Integration Tests (UT/IT), System Tests, Operation).

Table 27: Phase of introduction versus phase of detection

	Phase of Introduction	Requirements	Design	Implementation	UT/IT	System Tests	Operation	Total late detected	Total
Phase of Detection									
Requirements Verification		162						-	162
Design Verification		6	106					6	112
Implementation Verification		10	77	290				87	377
UT/IT Verification		18	0	9	351			27	378
System Tests Verification		2	0	0	9	10		11	21
Operation Monitoring		1	6	0	0	9	4	16	20
Total late detected		37	83	9	9	9	-	147	-
Total		199	189	299	360	19	4	-	1070

The large majority of defects (86.3%) were detected right after being introduced (shaded diagonal). However, a significant number of defects (147, or 13.7%) escaped both V&V and ISVV, being caught by later ISVV activities only. In Table 27, we divided the testing activities in two phases (the Unit/Integration and the System tests) as they provide an additional view showing that even within the testing activities there are defects that could have been caught earlier.

We can observe that an important number of defects that escaped previous ISVV or V&V activities were detected during Implementation Verification (60%) – of these, 10 defects were introduced during Requirements and 77 during Design. A closer look, depicted in Table 28, reveals that almost 80% of these are Documentation defects, and 10% Function defects, which are in-line with the overall results (Table 17). Thus, tackling documentation issues might greatly reduce defect propagation (see Sections 6.2.1 and 6.2.2 for the applicable list of root causes). Another important observation is that the large majority of slipped defects are introduced in the Design phase and not properly detected by design V&V activities, directing the analysis to indicate design conformance root causes (Section 6.2.2.5) as the ones to tackle first.

Table 28: Defects detected late, after Implementation

Defect Type	Design	Requirements
Documentation	65	4
Function/Class/Object	7	2
Algorithm/Method	2	3
Checking	1	0
Interface	1	0
Timing/Serialization	1	1
Defect Trigger	**Design**	**Requirements**
Traceability/Compatibility	46	5
Design Conformance	16	2
Consistency/Completeness	15	3

Furthermore, analyzing the data from Table 28, the defect triggers Traceability/Compatibility are accountable for 60% of the detected defects, while Design Conformance and Consistency/Completeness account for 20% each. As source code is more detailed and more concrete than architectural design components and requirements descriptions, it is normal that traces can detect more inconsistencies, especially missing information. Code analysis (Implementation Verification) is also largely supported by tools for static, dynamic and metrics analysis, and this is certainly the main reason why this phase catches a large amount of defects introduced previously and not detected in the appropriate phase.

The defects detected during the testing phases that originated from previous phases include 20 defects injected during requirements specification and 9 defects introduced during implementation, from which 14 defects are Function/Class/Object defects and 10 are Documentation defects. The requirements-related defects are mostly due to requirements quality root causes, while the defects introduced at implementation represent generally “bugs” or implementation mistakes, thus the root cause is related to the experience of the programmers and the efficiency of the code reviews.

This section does not contain a list of root causes but they can be found within sections 6.2.1 and 6.2.2, which contain a very comprehensive list of root causes applicable in this situation as well. For example, for the defect type related root causes a mapping with the first row of Table 26 can be performed (Documentation). For the defect trigger related root causes, the Traceability/Compatibility root causes are listed in Section 6.2.2.1.

6.2.4 Prioritization of the Root Cause Analysis

The prioritization is an optional step that is very practical for industry usage, if, for example, the objective is to save money and get the best return on investment by tackling the most important and/or the most frequent defects. It is arguable how we identify the importance of defects based on a classification that has reduced and

aggregated defects. However, for this purpose the selected Enhanced ODC classification scheme provides and interesting tool in the form of the Impact classification.

When we look at the possible impacts, we can identify a list that should get priority versus the other one due to the possible effect on the critical systems. This way we have labelled the impacts Capability, Reliability, Maintenance, Safety and Integrity/Security as the most important, and Documentation, Installability, Migration, Performance, Requirements, Standards, Testability/Verifiability and Usability as the less important. As it is quite easy to map the relation between defect types and their foreseen defect impacts, it is also possible to prioritize the identified root-causes in a similar way.

The defects with impact on Capability, Reliability and Maintenance (in our dataset there were no defects with impact on Safety nor Integrity/Security), identified in Table 17, represent 77% of the total dataset. From these, if we consider the top 6 defect types and the top 5 defect triggers we are already covering more than 90% of the high impact defects. Thus, we can select Capability, Reliability and Maintenance impacts together with the 6 more frequent types and 5 more frequent defect triggers safely for a more simplified root cause consolidation process.

The results of the defects types and triggers prioritization are presented in detail in sections 5.4.4.2 and 5.4.4.3.

6.2.5 Improvements Suggestions

The defects with impact on Capability, Reliability and Maintenance, as referred in Section 6.2.4 and detailed in sections 5.4.4.2 and 5.4.4.3, represent 77% of the total dataset. From these, we considered the top 6 defect types and the top 5 defect triggers (see Table 22 and Table 23), each of them accounting for more than 90% of the defects with high impact. Then, crossing these defects with the root causes identified in sections 6.2.1 to 6.2.3, we were able to filter the main root causes for the most important defect types (Table 29) and the most important defect triggers (Table 30).

This analysis results on a list of the most important causes of the defects identified during ISVV, and of the most important causes of failure in the verification and validation activities during the development lifecycle. For defects with high impact, the listed causes show that software engineering processes, methods and tools require some adjustments in order to become more efficient to produce more dependable and safe systems. The identified root causes are all related to existing development and V&V activities that require more careful application, especially in what concerns schedule and planning pressures, rigor and caution on the application of engineering processes, and V&V activities importance. The quality/product assurance strategies and the guidance from applicable processes and required standards are essential to ensure that these root causes are minimized.

Table 29: Summary of root causes for main defect types

Root Cause	Defect Types
Inefficient/insufficient reviews	Documentation; Function/Class/Object; Algorithm/Method; Checking; Interface
Ambiguous/missing/incorrect artefacts (documentation, requirements, design, tests)	Function/Class/Object; Algorithm/Method; Checking; Interface
Insufficient/Wrong tests (unit, integration, system, fault injection)	Function/Class/Object; Algorithm/Method; Checking; Interface
Limitations of the tools or toolsets that deal with documentation	Documentation
Lack of Completeness and consistency of system level (or previous phases) documentation	Documentation; Function/Class/Object; Algorithm/Method
Oversimplified documentation planning procedures Lack of time to produce, review and accept documentation artefacts Lack of importance given to some documentation artefacts Simplification of the product assurance processes related to documentation artefacts	Documentation
Limited engineers' domain knowledge – lack of appropriate skills	Function/Class/Object; Algorithm/Method
Incomplete specifications in what concerns FDIR and erroneous situations	Checking
Lack of reliability and safety culture	Checking
Incomplete specifications in what concerns interfaces, environment and communications	Interface
Limited definition of the operation, usability, maintainability requirements	Interface
Lack of tools knowledge, programming languages, design languages	Function/Class/Object; Algorithm/Method
Version and configuration management procedures inappropriately implemented	Build/Package/Environment

The root causes presented (in Table 29 and Table 30) have been ordered according to expert knowledge and experience applicable to the high impact defects, and intend to provide a preliminary ordering in what concerns their contribution to the high defect impacts.

The identified root causes for defect triggers indicate that improvements to the current processes, both development (to avoid the introduction of defects) and V&V (to detect the defects within the phase they are introduced) might be possible. At a higher level, the leading safety standards might require additional guidance to support development and V&V in order to reinforce that the product/quality assurance (PA/QA), and safety

and dependability assessments should be properly realized, reducing the number of defects caught by ISVV. The proposed improvements are guidelines derived directly from the root causes summarized in Table 29 and Table 30 and from domain and expert knowledge of the authors and industrial contributors to this work. Their intent is to fulfil the needs of the development and V&V processes to avoid the most important and more frequent defects as those in our dataset.

Table 30: Summary of root causes for main defect triggers

Root Cause	Defect Trigger
Lack of traceability verification culture	Traceability/Compatibility
Lack or inefficient usage of tools that support traceability across lifecycle phases	
Lack of appropriate test planning and test strategy definition	Test Coverage
Lack or inefficient testing tool and testing environment support	
Incomplete tests specification and execution	
Review process related root causes	Document Consistency/Completeness (Internal Document)
Documentation related root causes	Document Consistency/Completeness (Internal Document)
Deficient usage of tools and applicable processes	Document Consistency/Completeness (Internal Document)
Unclear or missing/incomplete specifications	Document Consistency/Completeness (Internal Document); Logic / Flow
Ambiguous or unclear architecture definition	Logic / Flow
Lack of usage of tools that support data and control flow analysis	Logic / Flow
Inappropriate architecture support tools or tool usage	Design Conformance
Deficient specification or design artefacts	Design Conformance

From the development perspective, and based on Table 29, the following measures (proposed improvements) should be considered to reduce/eliminate the root causes:

- Define/redefine appropriate review methods, processes and tools and enforce their application at every stage of the SDP;
- Implement automated documentation generation processes and tools to avoid inconsistencies between artefacts/lifecycle phases;
- Use tools that integrate and manage all the phases of the lifecycle, such as concept specifications, requirements, architecture, source code, tests, etc.;
- Introduce/use tools with automatic validations (documentation completeness, design consistency, code analysis, control and data flow analysis);

- Provide training to the engineering teams, to improve the domain knowledge, the system or interfacing systems knowledge, standards knowledge and techniques and tools practice;
- Promote workshops or meetings to present the specifications/requirements, to discuss and clarify them before advancing to the following phase;
- Introduce additional guidelines or even specific requirements (e.g. by defining and specifying the reasoning behind the standards requirements and how to achieve them in full conformance) in the applicable standards (PA/QA, version and configuration control and development).

From the V&V perspective, and based on the results in Table 30, the following measures (proposed improvements) should be considered to increase the defects detection efficiency:

- Define appropriate test plans and strategies, especially unit and integration tests. The soundness of the test plans and strategies will reflect in the success of the validation;
- Ensure appropriate (or automated) traceability analysis at every stage of the development lifecycle;
- Improve the testing completeness, coverage and reviews;
- Implement non-functional tests (fault detection, fault injection, redundancy, etc.);
- Apply or develop tools to verify and validate the implementation and design compliance.

6.3 Effort Spent on the Root Cause Analysis Activities

The root cause analysis was a very efficient activity as can be seen in Table 31. In practice, the dataset root cause analysis was performed in about 2 weeks of effort, and for any future analysis, the analyst can already reuse the identified root causes and measures defined, if they still apply. A large amount of the effort was spent on the brainstorming and identification of root causes for every specific defect type and for every specific defect trigger. The activity ended with the identification of 26 root causes (13 from the development perspective and 13 from the V&V perspective) and led to the suggestion of 12 measures (7 to improve the development part and 5 to improve the V&V practices).

Table 31: Effort Spent for the root cause analysis activities

Activity	Description	Effort (hrs)
Process Data Preparation	Collection of the Defect Types and Defect Triggers resulting from the enhanced ODC classification. Training on the selected root cause analysis methods (fishbone and 5 Whys).	8
Root Cause Analysis for Defect Types	Brainstormings and definition of the possible root causes for the 8 defect types existent in the defects dataset. Summary of the root causes by aggregating similar ones and simplifying the resulting list of causes. Spot check on some defects to confirm the applicability of the defined root causes.	24
Root Cause Analysis for Defect Triggers	Brainstormings and definition of the possible root causes for the 15 defect triggers existent in the defects dataset. Summary of the root causes by aggregating similar ones and simplifying the resulting list of causes. Spot check on some defects to confirm the applicability of the defined root causes.	22
Root Cause Analysis for the Late Detected Defects	Brainstormings and definition of the possible root causes for the late detected defects. Merging of these root causes with the previous analysis (for defect type and defect triggers).	6
Prioritization of the Root Causes	Analysis of the specific types and triggers and what impacts they generally cause. Selection of the top 5 most "important" impacts (Capability, Safety, Reliability, Maintainability and Integrity/Security). Filtering of the root caused based on this criteria.	8
Consolidation of the Root Causes	Simplification and aggregation of root causes in broader sets according to affinities and similarities. The result = 13 root causes groups for the defects types and 13 root causes for the defect triggers.	6
Identification of Measures	Definition of sets of implementation suggestions to the consolidated list of root causes. Analysis of the 26 resulting root causes.	8
Total		**82**

6.4 Final remarks

This chapter presented the root cause analysis results, one of the main outcomes of the defects assessment process defined in Chapter 3. The root cause analysis consisted in a structured process based on a fishbone analysis for the most frequent/important defect types and triggers. We have first analyzed the resulting defect types and identified root causes related to development issues, then we have identified the main defect issues according to defect triggers which gave us the V&V weaknesses. We have also (particularly) applied the root cause analysis to the defects that have slipped through at least one lifecycle phase. In a subsequent step, we have proposed a dedicated list of measures to tackle both sets of root causes (development and V&V). The applied root cause analysis could produce manageable results due to the orthogonal classification provided by the defined enhanced ODC taxonomy. This classification simplified

(aggregated) the defect types and triggers and allowed the analysis to be focused on typed subsets (we can consider that we identified common root causes, instead of individual ones).

From the resulting root causes, the main issues affecting critical systems in the space domain are related to engineer's mistakes, requirements and constraints, test strategies and completeness, and lack of appropriate tools. To solve the problems caused by the main root causes improvement on review processes, traceability activities and test plans, strategies and completeness are the most relevant.

The root cause analysis technique used (fishbone + 5 whys), can be replaced by any other root cause analysis technique as long as it is mastered and efficient and provides complete results. The set of root causes identified either in the individual analysis (per defect type or per defect trigger) or after consolidation can perfectly be reused for future analysis, or simply be used as a starting point to feed the root cause analysis process. The measures defined can also be reused as they are naturally connected to the identified root causes. The reuse of root causes and measures will make the process much more efficient and complete for the next time it is applied, and it will allow also the organizations to measure the progress by studying the trends in the changes of defect types and defect triggers (reflected in changes in the root causes).

Finally, even if the root causes defined for a specific dataset are a representation of the problems that lead to that dataset, and the proposed measure represent the delta or the improvements that are needed for the engineering teams that produced the systems, these root causes and measures can be also used as a reference for root cause analysis on other systems and on other domains. For more details on this topic see the next chapter that discusses the topics of validation of the defects assessment procedure and the application of the procedure to other domains.

Chapter 7
Process Validation and Application in Multiple Domains

"Program testing can be a very effective way to show the presence of bugs, but is hopelessly inadequate for showing their absence." - Edsger Dijkstra

The validation of a process shall be performed based on the applicability and acceptance of the process and of the process results (output). The first part requires the implementation in an industrial environment, some training and the application of the defects assessment process to sets of defects. The second part requires the existence of defects and much larger cycles, as suggested measures for improvement need to be implemented (this can be done at different levels and require different periods of time: new or modified engineering processes, techniques or tools, human resources training, guidelines, templates and standards, and so on) and their results need to be assessed by a second round of application of the defects assessment process or by the analysis of the reduction of the number of defects or of their severity. This is very costly and time-consuming or time dependent, for this reason we followed a different method to validate the process and the process results.

The method defined to support the validation of the results of our research was to create a dedicated questionnaire and have it answered by a selected and significant number of experts. The questionnaire covered the definition and acceptance of the process, and its applicability, simplicity and the results, and intended to collect the opinion of experts on the validity of the root cause and measures. The questionnaire also allowed the experts to share their experience by providing additional root causes and measures that could be useful for future root cause analysis.

The survey also had the intention to provide to the industry and academia the understanding of the applicability and need of a RCA based process, to compare and complete the root causes defined with additional/complementary root causes experienced by the experts, and to present the obtained measures and collect additional

ones from the exerts experience. The responses have allowed us to improve and adapt the defects assessment process, and to rank, in terms of perceived importance, the root causes and the proposed measures in order to improve software engineering processes. A side effect of the results obtained, also visible in the textual responses collected in Annex B, is the enhancement of the root causes and measures with additional common root causes suggested by the experts. In practice, our root causes and measures together with the root cause and measures proposed by the experts provide a very complete set of information that can be used in future root cause analysis, and that support the claim that the process provides results that are independent from the business domain.

The outline of this chapter is as follows. Section 7.2 presents the detailed results obtained from the survey provided to the experts and the discussion of these results, section 7.3 discusses some concerns about the applicability of both the process and the reutilization of the root causes and measures to multiple business domains. Section 7.4 covers the process improvements as suggested by the experts in the answered surveys and section 7.5 presents the final remarks about the process validation and the application to different domains.

7.1 Overview of the Process

The process for validating and collecting the approval of experts concerning very specific (and sometimes sensitive) topics is not an easy task. We had four main objectives with our validation questionnaire:

- to confirm that the proposed process is acceptable and usable by industry;
- to confirm the validity of the identified root causes and measures;
- to obtain feedback and promote improvements to the process and to the lists of root causes and measures, even if those come from different technical domains (this might be useful in future defect analysis for other datasets); and
- to measure the applicability of the process and results to different business domains (how generic these can be).

The first activity was to formulate the questions and have the questionnaire validated (see Section 7.1.1). Next, we selected the target groups of respondents of the survey (see Section 7.1.2). A large group of experts from academia and industry worldwide has been contacted, in particular experts working with dependability and critical systems. Finally, the responses to the questionnaire have been collected and the data analyzed (see results in Section 7.2).

7.1.1 Definition and Validation of the Questionnaire

Once we have defined the generic objectives (above) and research questions, we have formulated the survey specific questions. The survey intended essentially to gather the

view of the industry and academia in what concerns defect analysis of critical systems, in particular the occurrence of failures in systems and software development and V&V processes. The research questions included the confirmation of the applicability and need of a RCA based process (Research Question RQ1), the types of causes that were experienced by experts (RQ2), and what the challenges are to advance the methods and practice of RCA (RQ3).

The applicable systems are usually developed with utmost care, under very strict and mature processes and methods and usually require independent assessment in order to be fit to the purpose (qualified, certified, and homologated). These systems also follow well-defined quality assurance rules and originate a great deal of evidences (mostly documented artifacts) that can be consulted by teams and independent assessors. The survey operates around the results presented in the previous chapters, but also intends to collect additional expertise to improve the classification and the root-cause identification tasks.

The questionnaire (presented thoroughly in Annex A) is composed by the following parts:

- A short introduction to the topics, a relevant list of acronyms, a note about the anonymity of the answers, instructions on how and when to deliver the questionnaire, and a short presentation of the defects analysis process;
- Three core parts:
    - A set of General Questions (to quantify the technical domain and experience, the perceived importance of the role of standards, budget, schedule, external assessment and defects analysis);
    - A set of textual questions collecting feedback about the process (positive and negative) and a request for additional recommendations (the results are in Annex B), and questions about the level of acceptability/recommendation of the proposed defects assessment process, the strengths and weaknesses identified, and some additional suggestions to simplify or improve the process;
    - Four sets of questions related with: a) defect development root causes and suggestions; b) root causes and suggestions regarding defect detection failure; c) defect avoidance measures and suggestions; and d) V&V measures and suggestions.
- Additional comments to the questionnaire.

Once the draft structure and questions have been formulated, the questionnaire has been reviewed internally, then provided to a group of 4 experts (two from academia and two from industry) to have the questionnaire tested, reviewed and commented for validation. The results of this validation were not used since the questionnaire suffered some adjustments based on the feedback from the experts. The questions have then been adjusted and simplified with the feedback from the four experts, as it should be

complete and straightforward and answerable in less than 1.5 hours, without compromising confidentiality (ensuring anonymity of the answers).

Further validations have been done on each question in order to simplify the answers, support the respondents and avoid confusion or misunderstandings while responding. These validations included: proof reading of all questions, merging of similar questions, simplification by dividing one complex question into two or more, addition of helping text and guidelines to guide the answers, and appropriate reordering of the questions.

7.1.2 Distribution of the Questionnaire

The target audience of the survey has been selected based on the author's experience and list of contacts. The industrial background of the author provided a large and dedicated list of relevant contacts in the Aeronautics, Space, Defense and Transportation domains. An important tool used to support the selection of industrial contacts was LinkedIn [138]. A large list of academic experts has also received the questionnaire. This list was constructed based on contacts acquired over several international research projects such as Critical Step [139], CECRIS [140], VALCOTS-RT [141], and AMBER [142], just to name a few, and from the previous participation in international conferences such as DASIA, ISSRE, DSN, SAFECOMP, etc. These experts all belong to at least one of the following business domains: aeronautics, space, automotive, railway, defense, nuclear; and had experience in the following technical areas: critical systems, dependability, reliability, fault injection, V&V, RAMS.

In practice, the questionnaire was delivered directly by email to 171 experts both in MS Word and PDF formats and the respondents were given one month to answer. Some of the experts replied stating that they would not be available to provide feedback or they did not felt confident to respond to the survey, some others have shown their interest in the survey and in getting the results of the overall survey at the end. Finally, 36 surveys have been received and compiled, the results have been summarized and presented (see Section 7.2). We estimate that these experts have spent around 60 hours overall to provide their feedback.

7.1.3 Characterization of the Respondents

The questionnaire has been answered by 36 experts. Not all experts answered the full set of questions, or had opinion on every question where a textual response was required (Annex B). From the responses received, 17 were completely filled, 14 had more than 92% of the questions answered (corresponding to 20 out of 22 questions), and 5 had between 56% and 76% of completion of the responses. Some experts had no opinion concerning either one of the questionnaire areas or some of the proposed root causes or measures (we cannot conclude that no answer means total agreement with the proposed root causes/measures, but we can suspect that they had no clear or evident additions to our list). Overall, all questionnaires have been used to extract useful data.

In the analysis of the questionnaires we have simplified the domains to Academia, Transportation, Aerospace and Others. The distribution of respondents that claimed to work on these domains is: Academia – 19; Transportation – 18; Aerospace – 20; Other – 15. Only 7 researchers specified that they have experience in Academia, however, for this metric we have also considered researchers who claimed at least 5 years of experience in the academic field. Note that several experts have expertise in several domains, this is why the total adds up to more than 36. These results provide an interesting base for industrial analysis of the importance and extensibility our work. The average experience per expert was about 6.7 years for the 7 Academic respondents and 16.7 years for the 29 Industrial respondents (see Annex C for the experience distribution).

7.2 Validation Results

The data collected from the 36 questionnaires provided the validation arguments that we needed to improve and confirm the value of our process and results. The results of the survey are documented in Annex B and Annex C.

7.2.1 Relevance of RCA

The importance and relevance of application of a root cause analysis based process was at the center of the objectives of the survey. It is very clear from the answers provided to the general questions that root cause analysis is considered as extremely relevant. Most of the experts consider that the usage of standards is extremely important, consider that budget and schedule restrictions are very important, and that external assessment (such as ISVV or certifications) as well as root cause analysis are also extremely important for critical systems. These results clearly confirm that our motivations for acting upon such systems quality is shared by the community. The resulting answers are shown in Table 32.

Table 32: General Questions Summary

Questions	ER	VR	SR	NR	NO
Q3: Standards Importance	**53%**	33%	11%	0%	3%
Q4: Budget Importance	30%	**53%**	14%	0%	3%
Q5: Schedule Importance	**42%**	**42%**	11%	0%	6%
Q6: External Assessment Importance	**56%**	30%	11%	0%	3%
Q7: RCA Importance	**61%**	36%	3%	0%	0%

ER (Extremely Relevant), VR (Very Relevant), SR (Somehow Relevant), NR (Not Relevant), NO (No Opinion)

From questions Q8 and Q9 (Table 33) we observe that 28% of the inquired experts have already used ODC in the past, and 50% have used some kind of defect analysis technique. This is quite surprising for such a selected population, since only a little

more than half has ever used a structured way to study (recurrent) defects. Defects classification is also not common, but generally it is done with a very simple internal taxonomy to each organization.

Table 33: Experts Background knowledge Questions

Questions	YES	NO
Q8: Used ODC before?	**10 (28%)**	26 (72%)
Q9: Used Defect Analysis before?	18 (50%)	18 (50%)

7.2.2 Feedback on the Defects Assessment Process

This subsection presents the critics, suggestions and recommendations from the experts in what concerns questions Q10, Q11, Q12. Again, the full set of answers is included in Annex B.

When we asked if the experts would recommend such a process, we got 60% of recommendations and 33% of possible recommendation, while only 7% would not recommend such a process (See Figure 16 which does not include the 6 respondents who did now answered this question).

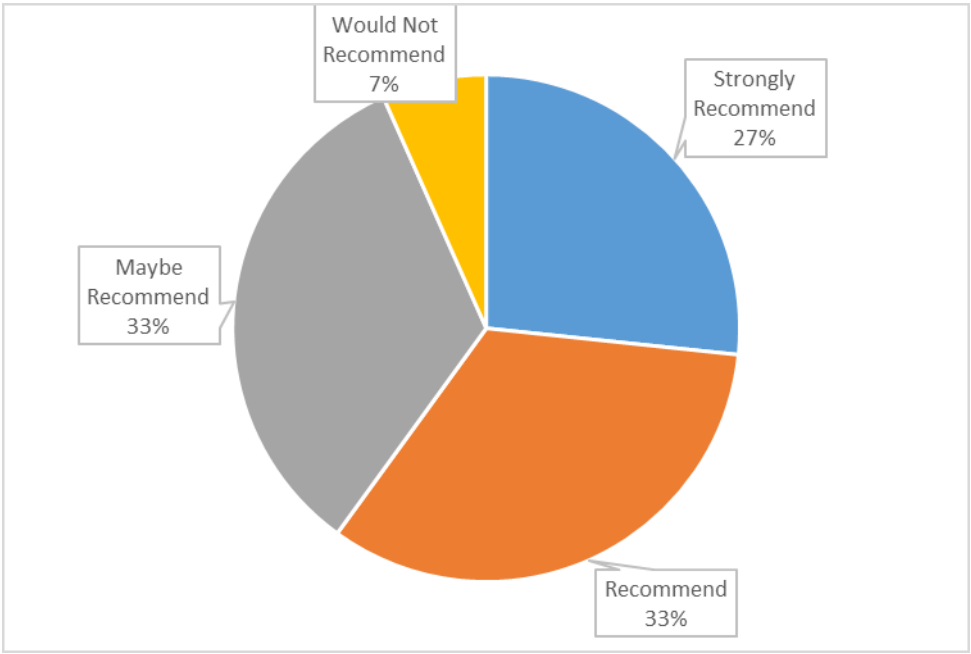


Figure 16: Process Recommendation Distribution

The comments on the process strengths, weaknesses and suggestions are presented in the following sub-sections.

From the 32 responses (89%) obtained on the **process strengths (Q10)** we can highlight the following ones: i) being a structured approach; ii) inclusion of root cause analysis; iii) inclusion of a classification scheme; iv) provision of improvements and feedback; and v) relying on high quality of data. These strengths are effectively the ones we wanted to achieve and demonstrate with the defects assessment process.

From the 32 responses collected on the **process weaknesses (Q11)** we can summarize the most relevant/frequent as: i) concerns about how to guarantee the quality of defect data (also related to strength v); ii) large amount of steps in the process; iii) missing feedback to the process (identified also as strength iv); and iv) difficulty in implementing/enforcing such a process.

These comments are all very relevant. First, the process will only work if the defect data is appropriate. For this, we shall relate comment i) with comment iv), as a cultural enforcement must be broader than just the application of the process, but also cover the defect data collection, the quality checks of the data, and so on. The large number of steps is required to have the process well detailed with simple blocks – the process can however be simplified as some steps can be considered optional (e.g. 6 – Late Detection RCA, 7 – RCA prioritization). Furthermore, the process contains feedback to both the process itself and to the development and V&V processes (strength iv).

Only 7 (19%) experts have not provided any additional **process suggestions (Q12)**. Overall, out of the three requested suggestions, we obtained a response rate of 58%, which is a significant amount of comments and suggestions and demonstrates the interest of the experts in the topics. The main and more commonly agreed process suggestions include (see Annex B for the complete set):

- **Data collection improvements (process, database, quality guarantee)** – some experts have demonstrated concerns about the quality and availability of good defects data. This topic has also been discussed earlier in Chapter 4;
- **Classification/validation activities and data quality checks between phases of the process** – this is a very relevant topic. We had to implement a validation of the classification of defects due to many classification doubts in early phases. We have also used the questionnaire to be able to generically validate the root causes and the measures proposed, but this is a general validation and not a validation between phases. We believe however that with the appropriate training, and with expert support, any organization can comfortably implement such a process and guarantee quality of the results after each phase;
- **Consideration of projects details/specifics and team dynamics (skills, experience, motivation)** – these parameters need to be considered mostly during the root cause analysis and measures definition. If the root causes point to any of these topics, then actions need to be defined to deal with these less technical properties;
- **Assessment covering also management related issues** – the management cannot be dissociated from the success or failure of the teams, and it can be included as well in the root causes, if considered necessary. Questions about

budget and schedule pressures have been included in the survey to measure the level of influence of management in the success of a critical system project.

We can observe that we have suggestions on the environment and prerequisites, which make absolute sense (data quality, projects details), and also on the validation of the internal process activities, namely regarding the quality of the classification, which cannot be easily automated as per today's technologies.

7.2.3 Evaluation of the Quality of the Root Causes

The following analysis presents the results and discussion regarding the experts' opinions on the main results of the application of the defects assessment process to the 1070 ISVV space defects, namely, de development defects root causes (based on defect types analysis) and the failure of detecting defects root causes (based on defect triggers analysis). This analysis is based on the 4 questions Q14, Q15, Q16 and Q17. While Q14 and Q16 represent the classification of the root causes identified and proposed by us to the experts, Q15 and Q17 represent newly suggested root causes according to the knowledge and experience of the experts.

The objective of this section is to show that the experts have a common view on the root causes identified for critical systems. Although the presented root causes are naturally associated to the defects dataset and the aerospace domain, we intended to determine if these results were still widely acceptable and potentially recurrent also in other domains. As we will see, this assumption can be corroborated by the presented results.

We have analyzed the frequency of words/groups of words and clustered the suggestions of the experts, and those are presented hereby for all the questions where a written opinion was requested. For the details on the proposed root causes see Annex B.

For the questions Q15, Q17, the number of provided suggestions (root causes) can be observed in Table 34 (we have simplified the questions text to make it fit in the table. Further textual description can be found in the questionnaire in Annex A). The experts provided more suggestions to root causes for development problems (62) than for V&V problems (36), and a relatively similar amount of Extremely Relevant/Frequent and Very Relevant/Frequent set of overall suggestions (38 and 43). The first observation that can be made is that the development is naturally more exposed to defects and experts have seen different types during their career. The V&V activities that are associated with the "lack of defects detection" are a limited set in terms of causes and have been quite well captured by our analysis. Moreover, the experts tend to spend time proving suggestions that they really consider important and relevant, according to their domains and expertise. As we will see next, although some of their suggestions are already covered by the results of our analysis, they have expressed them in a slightly different wording.

Table 34: Amount of the Proposed Root Causes and Measures

Questions	ER	VR	SR	NR	NO	Total
Q15: Development Defects Root Causes Suggestions	23	26	11	2	46	**62**
Q17: Failure of Detecting Defects Root Causes Suggestions	15	17	4	0	72	**36**
TOTAL	**38**	**43**	**15**	**2**	**118**	

ER (Extremely Relevant/Frequent), VR (Very Relevant/Frequent), SR (Somehow Relevant/Frequent), NR (Not Relevant/Frequent), NO (No Opinion).

7.2.3.1 Root Causes on Development Defects

We requested the experts' perception of the frequency of the identified root causes from a development perspective. These have been identified in the context of our dataset and for the space domain, but we expect these to be quite well acknowledged by the experts. Q14 presented the main root causes that we found during the development lifecycle. Figure 17 shows the order of "preference" or "importance" given by the experts to those root causes and the distribution of the relevance level. The colors represent the percentage (the left axis of the image) of responses of each importance/relevance level, while the green line provides an average of the overall responses for each root cause based on the weights given to the response (ER = 3, VR = 2, SR = 1, NR = 0, NO = excluded). The list of root causes is ordered by this average, showing the root causes considered more important/relevant to the left and the less important/relevant to the right.

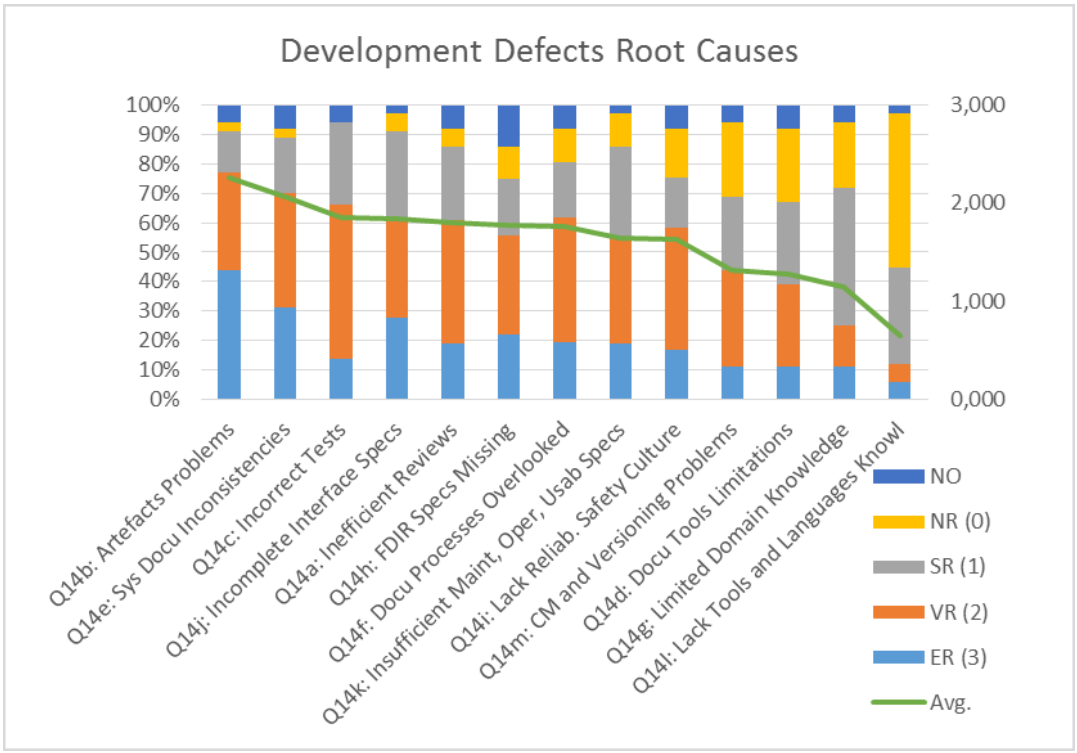


Figure 17: Development Defects Root Causes

Aerospace experts consider documentation tools limitations (decrease of 21%[5]) and lack of tools and programming languages knowledge (increase of 21%) as less relevant than other experts, and incomplete interface specifications (decrease of 15%) as more relevant than experts from other domains. Besides these causes both groups classify the root causes in a very similar way in terms of importance. Table 35 shows the relative difference between the classifications made by Aerospace experts and experts from other domains (based on the values presented in Figure 17).

[5] This means that for question Q14d: Docu Tools Limitations, Aerospace experts have provided a lower importance, according to the sum of the weights of NO, NR, SR, VR and ER, to this root cause than other experts. The difference of the aggregated sum in this case is 21%.

Table 35: Difference between Aerospace experts answers and others for Q14

Question	Difference
Q14a: Inefficient Reviews	9.0%
Q14b: Artefacts Problems	4.5%
Q14c: Incorrect Tests	11.0%
Q14d: Docu Tools Limitations	-21.0%
Q14e: Sys Docu Inconsistencies	3.6%
Q14f: Docu Processes Overlooked	10.5%
Q14g: Limited Domain Knowledge	6.7%
Q14h: FDIR Specs Missing	0.3%
Q14i: Lack Reliab. Safety Culture	0.8%
Q14j: Incomplete Interface Specs	15.0%
Q14k: Insufficient Maint, Oper, Usab Specs	6.6%
Q14l: Lack Tools and Languages Knowl	-20.8%
Q14m: CM and Versioning Problems	-9.3%

In what concerns the causes for defects introduced during "development", ordered according to the frequency of occurrence/importance, we can observe that 9 of the 13 root causes achieve an Avg. greater than 1.5, meaning that in overall they were evaluated at least as relevant/frequent by the set of experts.

We also observe that experts do not seem to consider the lack of domain knowledge, tools limitations and configuration management problems as the most relevant or frequent root causes. On the contrary, they consider that wrong implementations, incomplete reviews, incomplete tests and specifications for the error situations are the more frequent/important root causes.

As a complement we requested the experts to provide some additional root causes from their experience (which would not necessarily be applicable to our dataset) in order to compile a more complete and equilibrated list of root causes that can be used for the future independently from the business domain.

The aggregated summary of the new root causes (Q15) has been collected from the suggestions of 27 experts (75% of the total sample). The main topics for new root causes suggested are:

- Pressures impacting development artefacts (management, schedule/delivery, budget);
- Development/developers overconfidence or carelessness, as well as skills, motivation and interest in the project;
- Systems increasing complexity (technical and team sizes);
- Communication issues (customer, developers, V&V, management);
- Constraints on quality, availability of artefacts, and resources skills.

Several experts suggested root causes similar to the ones proposed in our survey, making the clustering easy. In fact, we observed that some of the suggested root causes are related specifically to some of our root causes proposed (described in the questionnaire), e.g. domain knowledge, poor documentation, Fault Detection, Isolation and Recovery (FDIR). Others, as those stated in the bullets above, are sometimes quite broad but constitute an interesting starting point for consideration in future RCA, and thus to be added to our list of 13 root causes to serve as a reference for future analysis.

7.2.3.2 Root Causes on Failure of Detecting Defects

While analyzing the defect triggers we have identified root causes for failing defect detection or for demonstrating how V&V activities have failed. The perception of the frequency/importance of these identified root causes was asked in Q16 and is depicted in Figure 18. The colors represent the percentage (left axis of the image) of responses of each importance/relevance level, while the green line provides an average of the overall responses for each root cause based on the weights given to the response (ER = 3, VR = 2, SR = 1, NR = 0, NO = excluded). The list of root causes is ordered by this average, showing the root causes considered more important/relevant to the left and the less important/relevant to the right.

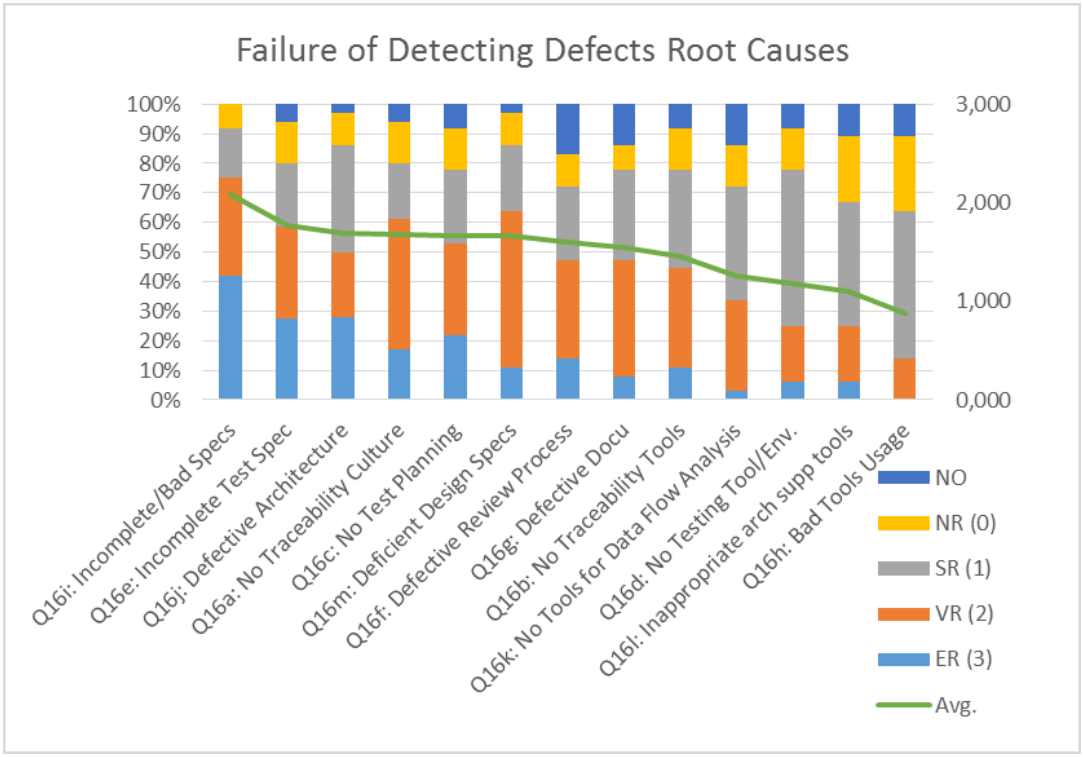


Figure 18: Failure of Detecting Defects Root Causes

Aerospace experts consider defective review processes (increase of 22%), non-application of traceability support tools (increase of 20%) and incorrect usage of tools (increase of 16%) as more relevant than experts from other domains. Besides these

causes both groups classify the root causes in a very similar way in terms of importance. Table 36 shows the relative difference between the classifications made by Aerospace experts and experts from other domains (based on the values presented in Figure 18.

Table 36: Difference between Aerospace experts answers and others for Q16

Question	Difference
Q16a: No Traceability Culture	6.7%
Q16b: No Traceability Tools	20.2%
Q16c: No Test Planning	7.9%
Q16d: No Testing Tool/Env.	2.1%
Q16e: Incomplete Test Spec	3.1%
Q16f: Defective Review Process	22.4%
Q16g: Defective Docu	11.1%
Q16h: Bad Tools Usage	16.4%
Q16i: Incomplete/Bad Specs	1.2%
Q16j: Defective Architecture	-3.1%
Q16k: No Tools for Data Flow Analysis	6.6%
Q16l: Inappropriate arch supp tools	-4.2%
Q16m: Deficient Design Specs	10.9%

Experts do not seem to consider that tools (again) are an important source for failing to detect problems, since the 5 lowest average root causes pointed to tools-related problems or lack of tools. On the contrary, they consider that specifications-related problems, test specifications and implementation of planning and review processes lead to the lack of defects detection much more frequently than the other root causes. From these values, we can observe that 8 of the 13 root causes achieves an average greater than 1.5, meaning that in overall they were evaluated at least as relevant/frequent by the set of experts.

The aggregated summary of the root causes proposed by experts for failure of detecting defects (Q17) has been collected from the suggestions of 18 experts (50% of the total sample). The main topics for new root causes suggested are:

- Knowledge limitations (domain, project, environment, faults, constraints, technologies, safety culture);
- V&V processes and culture (also considering dedicated standards);
- Staff inexperience (lack of testing expertise, lack of verification training);
- Management related issues (planning, pressures, risks, milestones, schedules);
- Size and complexity of systems (untestable requirements, long testing periods or impossibility of full coverage, lack of early involvement of safety/V&V teams).

Some of the root causes suggested are related to specific root causes already in the questionnaire, e.g. testing planning/strategy, or testing tooling/environment issues (also problems with tools). Others, as those stated above, can also be allocated to the development phases but constitute another interesting addition for consideration in future RCA, and thus to be added to our list of 13 root causes to serve as a reference for future analysis.

7.2.4 Evaluation of the Quality of the Measures

The following analysis presents the results and discussion regarding the experts' opinions on the main results of the application of the defects assessment process to the 1070 ISVV space defects, namely, the measures to improve development and V&V. This analysis is based on the 4 questions Q18, Q19, Q20 and Q21. While Q18 and Q20 represent the classification of the measures identified and proposed by us to the experts, Q19 and Q21 represent additional measures proposed by the experts.

The objective of this section is to show that the experts have a common view on required measures for critical systems. Although the presented measures are naturally associated to the defects dataset and the aerospace domain, we intended to determine if these results were still widely acceptable and potentially recurrent also in other domains. As we will see, this assumption can be corroborated by the presented results.

We have analyzed the frequency of words/groups of words and clustered the suggestions of the experts, and those are presented hereby for all the questions where a written opinion was requested. For the details on the proposed measures see Annex B.

For the questions Q19 and Q21, the number of provided suggestions (measures) can be observed in Table 37 (we have simplified the questions text to make it fit in the table. Further textual description can be found in the questionnaire in Annex A). The experts provided an equivalent number of additional measures for development and V&V (40 versus 42). The experts tend to spend time proving suggestions that they really consider important and relevant, according to their domains and expertise. As we will see next, although some of their suggestions are already covered by the results of our analysis, they have expressed them in a slightly different wording.

Table 37: Amount of the Proposed Measures

Questions	ER	VR	SR	NR	NO	Total
Q19: Development Measures Suggestions	21	10	7	2	68	**40**
Q21: V&V Measures Suggestions	21	13	5	3	66	**42**
TOTAL	**42**	**23**	**12**	**5**	**134**	

ER (Extremely Relevant/Frequent), VR (Very Relevant/Frequent), SR (Somehow Relevant/Frequent), NR (Not Relevant/Frequent), NO (No Opinion).

7.2.4.1 Development Measures

Following the root cause analysis, we have identified concrete measures to tackle the most frequent root causes and thus avoid the same type of defects in the future. The perception of the relevance/importance of these proposed measures was asked in Q18 for the development related root causes. Figure 19 presents the ordering of importance/relevance of these measures according to the classification performed by the experts.

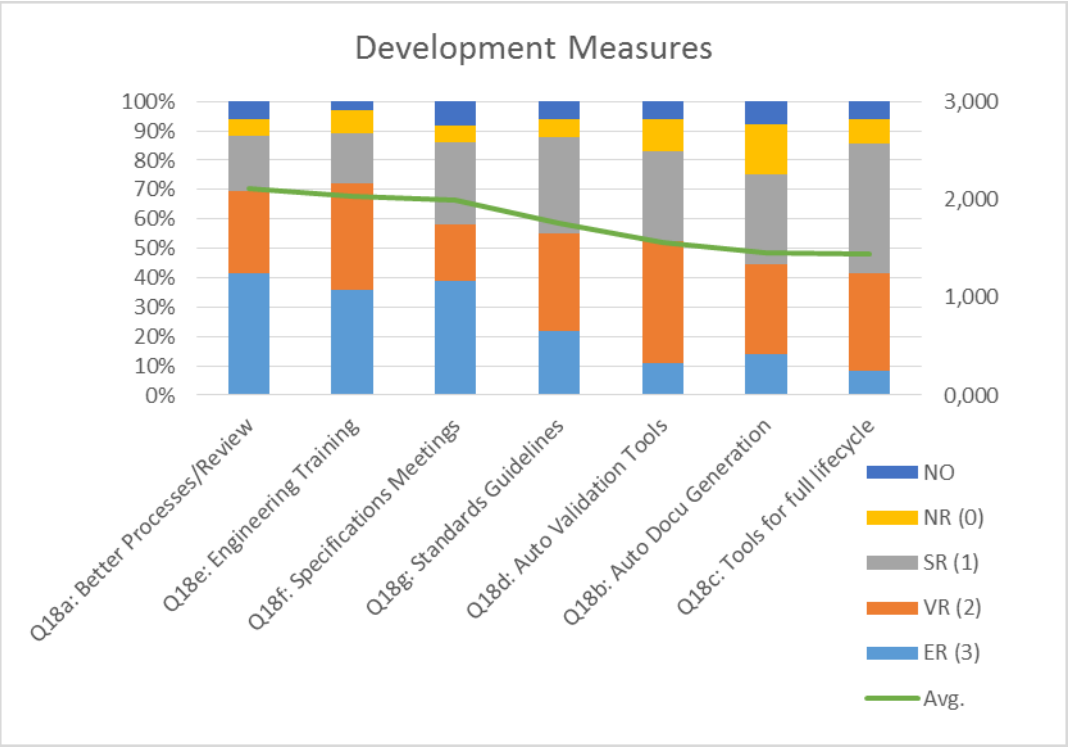


Figure 19: Development Measures

We observed that aerospace experts consider that Engineering Training (Q18e) is a measure that could help in reducing the introduction of defects more than the overall experts do (Table 38). The difference is about 25% increase in the importance of this root cause. The proposals of meetings to clarify the specifications, automated documentation generation and additional standards guidelines are also more emphasized by the aerospace experts, having about 16% increase in these measures. Most of the remaining measures are similar between the aerospace and the overall responses of the experts.

Table 38: Difference between Aerospace experts answers and others for Q18

Question	Difference
Q18a: Better Processes/Review	2.5%
Q18b: Auto Docu Generation	17.0%
Q18c: Tools for full lifecycle	0.9%
Q18d: Auto Validation Tools	-5.6%
Q18e: Engineering Training	24.8%
Q18f: Specifications Meetings	16.5%
Q18g: Standards Guidelines	16.3%

Experts do not seem to consider that tools (yet again) are the most important asset for reducing defects during the lifecycle. On the contrary, they consider that better processes, more frequent clarification meetings with the customer and dedicated engineering trainings are the main solutions. From these values, we can observe that 5 out of the 7 measures achieve an average greater than 1.5, meaning that, overall, they were evaluated at least as relevant/important by the experts.

Additional suggestions regarding development measures have been provided by 20 of the respondents (56% of the total sample) and allowed the identification of a large set of new measures:

- Require early results, early quality guarantee of requirements (or formal specifications);
- Promote defects knowledge and analysis, fault awareness (FDIR) training, compensations, etc.;
- Simplify systems, technologies, team management, lifecycles;
- Improve communication, meetings efficiency, mile-stones with more quality;
- Traceability simplification and implementation;
- Promote teams training and involvement at different phases;
- Use standards but plan for necessary improvements / additional tasks;

Once again there is a relation between some of the suggestion and measures already defined, namely the enforcement of strict review processes/development processes/documentation processes and the topics regarding standards. Moreover, some of the suggestions would also fit the V&V measures since there are commonalities, such as the traceability related measures.

7.2.4.2 V&V Measures

Following the RCA we have performed based on the defect triggers, we also mapped concrete measures to tackle the lack of defect detection. The perception of the relevance of these proposed measures was asked in Q20 and is depicted in Figure 21.

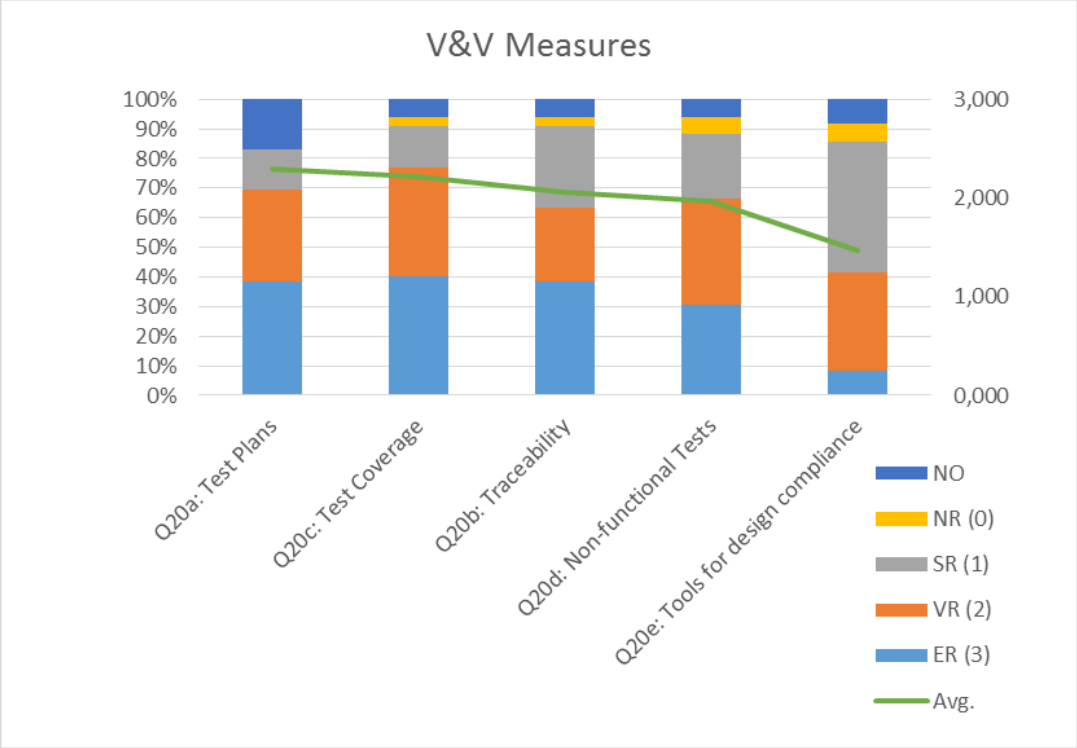


Figure 20: V&V Measures

Aerospace experts consider that Q20c Test Coverage (increase of 27%), Q20d non-existence of non-functional tests (increase of 23%) and Q20b Traceability are more important tools/measures than non-aerospace experts (depicted in Table 39).

Table 39: Difference between Aerospace experts answers and others for Q20

Question	Difference
Q20a: Test Plans	1.3%
Q20b: Traceability	15.2%
Q20c: Test Coverage	26.9%
Q20d: Non-functional Tests	22.8%
Q20e: Tools for design compliance	3.9%

Experts still do not seem to consider that tools are the most important asset for V&V to help detecting defects. More importantly, experts suggest that better test plans, improved test coverage criteria and better traceability analysis are the most important V&V actions to implement. From these values, we can observe that 4 out of the 5

measures achieve an average much greater than 1.5, meaning that in overall they were evaluated at least as relevant/important by the set of experts.

Additional suggestions on V&V measures were obtained from 21 of the respondents (58% of the total sample):

- More customer involvement in testing;
- Experts involvement for requirements testing;
- Tools to support V&V processes, static analysis, automation of testing tasks;
- Better communication between V&V and development;
- Monitor and measure quality;
- Enforce early defect detection (model-driven techniques, formal methods, test driven development).

As before, there were commonalities between the newly proposed measures and some of the measures already included in the questionnaire. For example, experts also proposed better test plans and strategies and more robustness testing (non-functional) – exploration of informal V&V methods. Moreover, some of the V&V measures proposed were already covered as part of the development measures on the questions of the survey (Q18), as is the case of better review processes, better code inspections and team training, focus, specialization, defects awareness.

7.3 Application to Multiple Domains

Several domains have been depicted in Figure 2 (Chapter 2). All the depicted domains have in common that they require a controlled and recognized development and V&V set of processes based on international standards and guidelines. In fact, the differences between software engineering in different domains is not so significant. Only some techniques and taxonomies actually differ from domain to domain as confirmed by a study performed over different safety critical standard [23]. Another study, within CRITICAL Software, S.A. that operates in different domains, has demonstrated that the differences of techniques, tools and skills from teams working in one domain to move on to another domain are very manageable and require and acceptable amount of effort and time [13]. This shows that a common set of skills, techniques, tools and practices are totally reusable and so should be the defects assessment process.

Our dataset contains defects from space projects and some defects from aeronautics (airborne) projects. During the dataset clean-up, we have removed the aeronautics defects as they did not contained structured nor enough information to be used. Once the defects assessment process defined in Chapter 3 has been applied to the space defects (especially with the enhanced ODC taxonomy) and validate it can be applied to different domains, an on-going activity is being performed with a set of about 150 railway defects. The preliminary results demonstrate that the enhanced ODC taxonomy is properly adapted to the railway domain, and that the process can be applied to another

domain with no modifications required. Nonetheless, the process allows for the modification or replacement of the classification taxonomy, as long as it remains orthogonal. Thus, a different classification taxonomy can be used or the currently proposed enhanced ODC can also be adapted if there is the need (for example due to the impact of the usage of new technologies, emergent behaviors or cybersecurity issues). Any adaptation, should, at this point, be very precise and localized. Another area of the process that can use different techniques is the root cause analysis. We have applied the fishbone diagrams and the 5 whys techniques, but any technique based on expertise and knowledge of the systems and environment can be used to determine the root causes.

In order to further collect feedback from the applicability of the process to an enlarged set of technical domains, we have compiled a questionnaire (see Annex A) that was delivered to a large set of experts (industry and academia) working in a diverse set of domains. Both the process and the analysis results, have been widely accepted by the experts, meaning that the root causes identified for the space defects can also be applicable for other domains. This has three main reasons: first, the engineering processes and tools are not that different; second, the guidance or standards have quite similar objectives; and third, the identified root causes and the associated improvement measures as identified by our activity have been defined in a very generic and thus broadly-accepted way.

Since the engineering processes and tools used are very common, the result depends on the maturity of the organizations and the experience of the engineers, thus some very pervasive problems occur. The standards and guidance requirements having common objectives (e. g. safety, security, availability, reliability, dependability, quality, certification or qualification) also lead to the usage of common processes, techniques, tools, reuse, and this will again lead to problems that can be found across domains. Finally, our lists of development root causes, V&V root causes, and the proposed improvements are not specific to any specific defect (it would be very extensive to do it for the 1070 defects) and is purposely generic being this way applicable to any engineering context (engineering domain). This genericity was one of our essential requirements for this work.

As the implementation is mainly performed by engineers, they tend to make common mistakes and require the same type of improvements.

The results of the survey, presented in before, have highlighted that the opinion of experts is not very divergent in what concerns the applicability of the identified roots causes and the proposed measures. Our dataset contains a diverse set of defects from all lifecycle phases, form the aerospace domain. Nonetheless, we observed that the perception of the importance of the root causes for development and V&V is very similar between aerospace experienced respondents and the global population (as shown in Table 35, Table 36, Table 38 and Table 39). Besides the differences, none of them is higher than 30%, we observe only small divergence of opinions. There are only two proposed root causes where non-aerospace experts have given a higher importance than the aerospace experts, and these are: *Q14d: Documentation Tools Limitations* and *Q14l: Lack of Tools and Languages Knowledge*. However, these root causes are also

classified as the least relevant ones for both aerospace and non-aerospace. Similarly, regarding the measures proposed, the only ones that had a significant difference between the aerospace and non-aerospace are: *Q18e: Engineering Training* and *Q20c: Test Coverage*. These two measures are however considered among the most relevant, but aerospace experts consider that they are much more relevant than non-aerospace experts.

The experts have also expressed their opinion on the process that led to those results. The acceptance rate was positive and there were no differences observed between experts of the different domains, nor between academia and industry. The tasks that compose the process are also generic enough to be applied to any engineering project, independently from the domain. It is currently being applied to the transportation domain without any issue. The same process, with the same ODC taxonomy is being applied to a set of 150 railway train control and management system defects.

We conclude that the process with the Enhanced ODC taxonomy and the root cause tasks is potentially independent from the domain, so it can be applied generically. Moreover, the set of root causes and measures seems to be also well accepted by experts from the different domains. Further suggestions have been proposed but they do not seem to be connected to any business domain, as they are also very applicable to all domains.

7.4 Process Improvements as a Result of the Process Validation Activities

Based on feedback collected from the empirical and practical application of the process to a large set of space software defects, and then with additional feedback from industry and academia, we have derived the process depicted in Figure 10. Although our dataset and our experience is mainly from space software, this generalization can support the evaluation and root cause analysis of any critical system, independently from the domain (as confirmed from the set of industry and academic surveys performed).

7.4.1 Data Collection and Preparation Improvements

Data collection seems to be a straight forward activity; however, it is not always easy to get access to the right or the complete data necessary to classify the defects and to determine the root causes. Some suggestions that have been included in the defects analysis procedure are:

- **Engineering training on how to properly raise defects** – the engineers should be prepared to promptly and correctly report defects, and to document them completely in a self-contained manner, with the appropriate references and with a common and simple language;
- **Definition of a template with the basic information required to be collected for every defect (this can also be implemented on a defect tracking tool)** –

definition of the mandatory information fields required for each defect that is raised and provide hints or multiple choices whenever possible for the filling of those fields;

- **Procedure for early evaluation of the quality and completeness of the defects data** – define some metrics or develop a tool to evaluate the completeness and quality of the contents of the defects, the natural language used, etc.;
- **Pre-selection of the defects to decrease analysis effort** – with the definition of a criteria (for example based on the priority and/or severity of the defect) prioritize and select a subset of the defects to promote the classification and the root cause analysis on a smaller and more important set of defects;
- **Usage/update of a domain specific database of defects** – generalize the defect types or examples and complete the analysis, record it on a database and reuse it for future occurrence of similar defects of defects related to the same topics.

The contents of the template with the basic information required to be collected shall include, at least: reference artefact, defect title and defect detailed description, phase where the defect was identified, phase where the defect was introduced, activity that detected the defect, defect author, defect severity, and defect relevant keywords.

7.4.2 Defects Classification Improvements

In this work, defects classification was based on the selected orthogonal defect classification (ODC) and the proposed enhancement. The classification process depends on the selected scheme and taxonomy (see Chapter 5 for the proposed classification and relevant details). Some suggestions have been made by industry and academic experts to our proposed classification process:

- **Define and provide training on the classification of defects and on the applied taxonomy** - to make the classification more efficient, easy and homogeneous, since understanding of the classification taxonomy is essential for a sound classification;
- **Explore the usage/configuration of tools to support the defects classifications (e.g. JIRA, Bugzilla, …)** – tools must support the classifications and analysis of the defects, as well as the access and storage of the defects data;
- **Adapt ODC for object/service/aspect oriented developments and for agile** – to support new development paradigms and be able to cover future needs of the project teams;
- **Promote continuous improvement of ODC taxonomy (from feedback)** – as stated in the process last step (11. Process Validation and Improvement), regular feedback to the whole process is important to support the technological changes in system development;

- **Improve confirmation of the proper defect classification (confirmation by another expert – validation of the classification)** – defects classification validation and confirmation is necessary, and this is commonly done by additional experts, different solutions for the appropriateness of the classification shall be explored;
- **Use and update the defects database (classified defects)** – the usage of a defects database, and relevant classifications and constant updates/improvements to that database are required;
- **Explore the possibility of getting defects automatically classified with support of the defect author (e.g. by using keywords)** – either by improving the templates and the guidelines for writing defects or by requesting the defect author to fill in some additional fields, collect extra information to support on the defects classification activities.

7.4.3 Root Cause Analysis Improvements

The root cause analysis requires expertise and effort to be properly performed. The proposed process contains different steps and different root cause analysis activities. Some of the suggestions made by the industrial and academic experts consulted are:

- **Provide root cause analysis training** – training and practice on root cause analysis is important to properly perform the root cause analysis activities and avoid doing only a high level and flawed analysis;
- **Use of a root causes database as a baseline to identifying the defects causes** – the defects database with associated classifications is a key element for future automation and guidance on classifications of similar defects or groups of defects;
- **Promote some type of automation for the root cause analysis activities** – with additional information provided by the defects authors or with the support of the classified defects database some automation or classification suggestions can be performed;
- **Define and derive root causes per domain or system type (together with the database)** – the database should have categories or groups of defects (similar defects, subsystem types, defects from specific lifecycle, phases, etc.) that can support classification of similar defects in the future;
- **Separate root causes depending on target groups (development, V&V, management)** – the root causes separated by type can simplify their distribution, implementation and acceptance by the target groups;
- **Promote a quick comparison with previous defects root causes to see if the systems are improving** – verify that the frequency of previously identified root causes is reduced and conclude on the suggestions and actions implemented.

7.4.4 General Process Improvements

The root cause analysis lead to the identification of changes to prevent or avoid defects from existing or from slipping between lifecycle phases. The implementation of these changes (improvements) might lead to the expected results, but might also reveal different type of problems, that will need to be tackled in a subsequent cycle. For engineering cycles that are long (projects that have, for example, 5 years duration), it will take some time to measure the effect of the implementation of the improvement, and some improvements might not be easy to implement because they will depend on training, cultural changes, technology evolutions, tools that need to be purchased, guidelines or standards that need to be updated and followed, etc.

Some of the suggestions made by the industrial and academic experts consulted are:

- **Discuss and agree the changes with management before presenting them to the affected groups, to ensure their commitment and sponsorship** – a formal discussion and acceptance of the suggestions based on the identified root causes is the best way to have the solutions implemented;
- **Monitor the improvements implementation, maybe through a simplified and lighter process that quick confirms the reduction of the defects** – monitor and track the metrics of the defects rate and the root causes occurrences to conclude in the effectiveness of the implemented measures;
- **Define and adopt some metrics to measure the quality of the results** – define specific metrics such as the defects recurrence, the root causes frequency, by severity, etc., in order to measure the quality and effectiveness of the implemented modifications.

7.5 Final remarks

We have created a survey to measure the acceptability of the defects analysis process and to confirm that the obtained results are in-line with the community experience and background. This survey has been answered by 36 experts and provided us not only a ranking of the root causes and possible measures but also some feedback on the process and some additional root causes and measures.

The main outcomes from the survey results were discussed in this chapter. The community (both academia and industry) considers that defect analysis, external assessments and the use of standards are of utmost importance. In parallel, budget and schedule constraints play an important role in the quality of the projects outcomes. Almost half of the respondents had never performed defect analysis, so this is an area that really needs improvements to be implemented.

The proposed assessment process got acceptance from a majority of the respondents. They claimed that the process was structured and could provide valuable feedback; however, they were concerned about how to obtain good quality of defects data and how to implement a process with a large number of steps. Some of the proposed

suggestions were exactly concerning the data quality and how to validate the obtained results. We claim that usage over the time and monitoring of the defects frequency and impact will be a good validation method.

The root cause results (development and V&V defects) have allowed the ranking of our identified root causes, and this has shown that the experts consider wrong software implementations, incomplete reviews, incomplete tests or test plans and lack of consideration of error situations to be the most common problems, while lack of domain knowledge and tools do not seem to be the causes for most defects. This is an interesting result that shows that the critical software domain is still heavily dependent on manual actions, on human skills, and on traditional development (from specifications to tests).

For the ranking of the measures to apply, the most voted were review processes, improved communications, test plans and tests coverage, better traceability and engineering trainings. Again, tools and automation are generally the V&V and development assets that seem less important for the experts. This shows once that the manual and human based techniques (that are simple) are still the ones the community is considering more applicable. There is maybe a cultural change to operate in these areas.

Additional suggestions for root causes and measures have been proposed by the experts. Although these were very much in-line with those that we proposed based on our dataset, they also provide an interesting addition for future RCA. The proposed root causes and measures arise from the specific experience of the experts and provide a good overview of the problems (and possible solutions) for critical systems at large. The conclusion we can make is that our process produced a very well accepted set of root causes and measures, and this supports positively the evaluation of the process and the methods that compose it.

Chapter 8
Conclusions and Future Work

The topics of quality, dependability, reliability, integrity and safety of critical systems, such as space, railways and avionics systems, are of utmost importance. These systems operate vital functions and cannot fail. It is however not possible to have a system that does not fail or does not contain defects, and this is observed, for example, when independent assessments of these systems are performed. The significant and frequent amount of defects (sometimes major defects) is the main motivation to develop a process to support the engineering teams in reducing the amount of defects by learning from previous mistakes.

A complete process was defined, applied, validated and refined, in this work, covering four main phases, from the defect data collection and preparation, to the defects classification, the root causes analysis and measures identification, to the validation of the measures implementation. The defects assessment process was applied to a set of 1070 space systems defects, and validated with the support of a survey provided to different academic and industrial experts. The process can be used and applied in industry in a simple way and independently from the industrial domain. In practice, the process takes as input defects identified on the engineered systems, and supports the analysis of these defects towards identifying their root causes and defining appropriate measures to avoid them in future systems.

Within the defects assessment process activities, the adaptation of the Orthogonal Defect Classification (ODC) for critical issues is a key step. The original ODC was used for an initial classification and then it was tuned according to the gaps and difficulties found during the defects classification activities. The improvements were necessary and covered the defect types, defects triggers and defect impacts. Improved taxonomies for these three parameters have then been devised and applied to the full set of defects and validated with the support of experts, demonstrating their appropriateness.

The most important part of the process is, however, the application and integration of a set of root cause analysis steps. These steps produce results that show the origins of the defects. The identified causes are related to different parameters, such as human

and technical resources, events occurred, processes followed, methods applied, tools used and standards followed. The fishbone root cause analysis proposed and applied to the defects dataset has proven to be quite effective since the obtained results were well accepted by the experts during the validation phases.

As a result of the root cause analysis, a specific set of root causes and applicable measures to improve the quality of the engineered systems (removal of those causes) have been identified. These root causes and proposed measures allow the provision of quick and specific feedback to the industrial engineering teams as soon as the defects are analyzed. A list/database has been compiled from the dataset and includes the feedback and contributions from the experts that responded to a process validation survey. The root causes and the associated measures represent a valuable body of knowledge to support future defects assessments as confirmed by the answers and classifications of the experts.

The measures proposed to improve systems have shown the importance of using empirical data (of defects in this case) to contribute to technical and processual improvements in order to get better overall quality and improved dependability levels for systems that are critical. In fact, the outcomes of the field study show that, although critical systems are already guided by appropriate development and V&V techniques and processes, most of the defects are caused by an inefficient usage or implementation of these techniques and processes. Appropriate guidance, additional requirements and constraints, better test strategies and tools that are able to help in the application of the techniques and processes are essential to obtain better results (less defects).

We can conclude that the main issues affecting critical systems in the space domain are related to engineer's mistakes, requirements and constraints, test strategies and completeness, and lack of appropriate tools. The results also suggest that our process (supported by the classification scheme and the root cause analysis) allows the identification of improvements for specific areas and groups of defects, and that review processes, traceability activities and test plans, strategies and completeness are the most relevant V&V tasks to be improved/enforced for better detection of the defects in space systems.

As a result of this study, and due to the industrial cooperation behind it, several parties have already shown interest in the results of the current analysis to promote internal awareness and process improvements.

8.1 Discussion

Some topics are worth being mentioned as part of the conclusions and of the obtained results.

Firstly, the proposed ODC adaptation. The classification scheme is a core element of the process and we have selected ODC due to its implantation in industry, orthogonality and comprehensiveness. However, soon we identified difficulties in classifying issues without ambiguity: we had a first classification with 27% of ambiguous classifications, but once this classification was reviewed and completed by an expert engineer the

ambiguity increased to 31.7%. We solved this issue by proposing an adaptation of the taxonomy, which was based on the study of the ambiguous classifications and on the judgement of experts.

Secondly, the reduction of the ODC attributes. The selected subset of ODC attributes come from the fact that not all attributes bring useful information to support RCA. A key objective was to make the classification easier (for Type, Trigger and Impact) and to cope with the fact that the classifiers had some troubles categorizing certain defects. We believe that the proposed taxonomy allows a more efficient and faster classification (even if losing some specificity), as we are joining some classes that had similarities (probably removing ambiguity and human error probability, but that is not easy to validate). The reclassification (with the enhanced ODC taxonomy versus the original ODC) lead to:

- More 'Documentation' defect types being observed (12% increase). We have classified some of the defects in this category after a deeper analysis of the defects where we had defect type doubts.
- 'Traceability/Compatibility' became the most frequent trigger and even 'Test Coverage' became a trigger which lead to the identification of more defects than 'Consistency and Completeness'.
- 'Maintenance' defect impact became more frequent than 'Reliability', and the 'Documentation' impact frequency has been reduced.

Thirdly, the classification certainty/precision. We are aware that such a process cannot lead to a fully precise classification. Human error/bias shall be taken into account. Although the classifications have been performed by experienced engineers, and confirmed/reviewed by a second engineer, some of the classifications are always arguable – this is due to the understanding of the problem, to the expertise/background of the engineer, and to the classification taxonomy itself (this was a problem we tried to tackle). Moreover, the proposed/used adapted taxonomy is meant to be a step towards a more applicable classification for this type of defects/systems and demonstrates that such a classification taxonomy can (and will be) adapted in the future. If we take, for example, the results of the enhanced ODC classification of our dataset, even with a 20% error interval we can conclude that the two main defect types, the three main defect triggers and the four main defect impacts would remain the same, thus giving us a group of consistent taxonomy priorities to tackle for root cause analysis.

Fourthly, the efficiency and employability of the process. The 850 hours required to apply the original ODC include the whole process, that means, the data collection and clean-up, additional data collection (phase, activity), and defects classification, where several doubts arose (in fact, almost 1/3 of the classifications have been challenged between the classifier and the reviewer). The classifier had to take note of his doubts and analyze the defects that could not be classified together with the reviewer, by aggregating new class types. Later, the not classified defects have been reclassified, the ODC classification was enhanced with the new class types and rechecked by the

reviewer. This last part (reclassification and review) took under 100 hours as most of the effort was in the first pass of the activity.

Fifthly, the empirical Software Engineering nature. The process is based on data, but actually several of the steps depend on training and expert judgment such as the classification, the root cause identification, and the suggestion of improvements. We thus acknowledge that replication of the results will be hardly possible, as they depend on the experts involved and on their expertise. For this reason, a questionnaire to validate the process and the results has been defined and answer by a group of academic and industry experts.

Sixthly, the defined process. The proposed and applied process is heavily dependent on the availability of good quality data. Data clean-up activities have been performed not in the sense of removing any defect, but to remove the names of the projects/missions, companies, systems/sub-systems, customers, in order to avoid that information to be known publicly for confidentiality reasons. No manipulation or modifications on the defects text have been performed. The data clean-up was also enriched by the gathering of additional data and complementing the defects data with the phase detected, phase introduced, detection activity, etc., for better supporting the ODC classification. In the future, if guidelines and rules for defect writing are defined (e.g. with some lessons learned from this work), the data clean-up step might be simplified or automated.

Seventhly, the obtained results. A list of root causes was identified by the experts, and not necessarily concrete or recurrent problems. From our knowledge of the systems and of the defects resolutions, we believe that some of the root causes are more frequent than others. We can comment on the quality of the systems under analysis and the application of standards, and this (non-measured) question was what lead to this work in the first place: as the number of issues was deemed too high for these systems, we decided to concretely identify why, in order to support the engineering team in correcting these deficiencies. With this work, we have concrete improvements to avoid the same issues and reduce their frequency.

Finally, the process validation. The process, the defects classification, the root causes and the identified measures have been exposed to a group of academic and industrial experts that expressed their opinions by answering a survey, by prioritizing the root causes and the measures and by proposing process improvements, suggesting additional root causes and additional measures according to their expertise. The results of the survey allowed the validation of the process and of the results themselves.

8.2 Threats to Validity

The main threats to validity of our work (construct, internal and external validity), due to some limitations and confidentiality issues, are:

- The fact that the issues data cannot be shared nor publicized, as no company wants their issues exposed, makes this work harder and demands a great effort

of anonymization. Also, the acceptance of the results may be challenged. (external validity)

- The space systems involved cover most of the development activities performed for those systems, and involve different companies (at geographic, size and management levels), thus we consider the results to be quite general for this domain. A similar study for other domains (e.g. aeronautics, railway, automotive) is foreseen as future work, but it will not be so easy since the existing data is not as structured as for space systems. Again, data confidentiality will be a challenging issue. (internal and external validity)
- The classification was done based on expert knowledge. However, it is important to note that the original classification (the one that could not classify all the issues) was performed by two engineers, whose work was also checked by a third space domain expert. This domain expert also performed the reclassification himself (verified and discussed with another space domain expert). (internal validity)
- Implementation of the suggestions will take a long time as it needs to go through process improvements and this is foreseen as current/future work. However, what is important here is the justification of these suggestions and also the acceptance and acknowledge of them by the involved industries. Once this is done, they can pay more attention to these root causes, and some months after that we may try to measure again the defects occurrences, types and triggers. Note that the provided lists of improvements are already the ones selected for tackling the most frequent and most critical defects. (construct validity)
- Finally, the adaptations were performed based on the 31.7% of the issues that could not be classified with ODC. This required several rounds of discussion, and the majority of changes are merges where terms were not well distinguishable for these systems. Also, details about the systems requirements (namely non-functional, safety and dependability, etc.) originated doubts about the original ODC classification. (internal validity)

8.3 Future work

Specific topics for future work are detailed in Section 7.4, aggregated by group of process activities, and reflected as improvements to the process and development of additional tools and guidelines to support the process. Some of these future works are highlighted hereafter:

- The validation of the process by applying it to datasets from other safety-critical domains. This will be achieved by doing a similar field study for railway (already on-going, based on 150 railways train management and control system defects), and for other domains such as automotive or avionics.
- It might be worthwhile to map the ODC defect triggers with the activities of the SDP in the lifecycle following the V-model (they should be fully mapped and

feedback can be provided both ways, to the ODC list of triggers or to the techniques applied in the lifecycle phases), and also map the ODC defect impacts with the ISO25010 [143] metrics (they should also be fully mapped and feedback can be provided both ways, to the ODC list of impacts or the to the ISO standard).

- By tracing the applicable standards, processes and techniques to the root causes and the proposed measures, we can work on proposing improved or new processes, techniques and tools to reduce the amount of issues and the probability of their occurrence.
- It is worth trying to automate some of the steps of the process, in order to promote automatic classification of defects or automated suggestions for root causes that can be stored in a database and associated to the classification taxonomies.
- The development of templates, guidelines and training to support the different activities of the process, in order to harmonize the classifications and the root cause analysis.
- To create and maintain a database or a body of knowledge of the root causes, associated measures, in order to reuse them or help in future defects analysis.
- The achieved results shall be consolidated to provide feedback to the ESA ISVV guide [65], regarding the results and triggers efficiency (according to the V&V techniques efficiency), and to the ECSS standards – to enforce additional PA/QA activities and evidence analysis that the requirements on the standards are appropriately being implemented. Note that this guide will be updated to become and ECSS handbook.

References

[14] N. G. Leveson, Safeware: System Safety and Computers. AddisonWesley, 1995.

[15] J. L. Lions, “ARIANE 5: Flight 501 failure,” Ariane 5 Inquiry Board Report, Paris, Tech. Rep., 1996.

[16] Australian Government, Australian Transport Safety, Bureau, “In-flight upset event, 240 km north-west of Perth, WA, Boeing Company 777-200, 9M-MRG” http://www.atsb.gov.au/publications/investigation_reports/2005/AAIR/aair200503722.aspx, August 1, 2005. Visited: 13 October 2016.

[17] Department of Transportation, Federal Aviation Administration, Airworthiness Directive 2015-10066, May 1, 2015, http://federalregister.gov/a/2015-10066. Visited: 13 October 2016.

[18] Toyota: Software to blame for Prius brake problems, CNN News, http://edition.cnn.com/2010/WORLD/asiapcf/02/04/japan.prius.complaints/index.html, February 5, 2010. Visited: 13 October 2016.

[19] US Department of Transportation, National Highway Traffic Safety Administration, “Technical Assessment of Toyota Electronic Throttle Control (ETC) Systems”, February 2011, http://www.nhtsa.gov/staticfiles/nvs/pdf/NHTSA-UA_report.pdf. Visited: 13 October 2016.

[20] NASA Engineering and Safety Center, “Technical Support to the National Highway Traffic Safety Administration (NHTSA) on the Reported Toyota Motor Corporation (TMC) Unintended Acceleration (UA) Investigation”, January 18, 2011, http://www.nhtsa.gov/staticfiles/nvs/pdf/NASA-UA_report.pdf. Visited: 13 October 2016.

[21] Weinberg, Gerald M. Perfect software--and other illusions about testing. New York, NY: Dorset House Pub, 2008.

[22] Ebert, C.; Jones, C., "Embedded Software: Facts, Figures, and Future," Computer , vol.42, no.4, pp.42.52, April 2009.

[23] Esposito, C., D. Cotroneo, and N. Silva. 2011. “Investigation on Safety-Related Standards for Critical Systems.” In Software Certification (WoSoCER), 2011 First International Workshop on, 49–54. doi:10.1109/WoSoCER.2011.9.

[24] W.H. Maisel et al., “Recalls and Safety Alerts Involving Pacemakers and Implantable Cardioverter-Defibrillator Generators,” JAMA, vol. 286, 15 Aug. 2001, pp. 793-799; http://jama.ama-assn.org/cgi/content/abstract/286/7/793.

[25] Silva, N.; Lopes, R., "10 Years of ISVV: What's Next?," Software Reliability Engineering Workshops (ISSREW), 2012 IEEE 23rd International Symposium on , vol., no., pp.361.366, 27-30 Nov. 2012

[26] B. Boehm and R. Turner, Balancing Agility and Discipline: A Guide for the Perplexed. Addison-Wesley, 2004.

[27] Bennatan, E.M, “On Time Within Budget - Project Management Practices and Techniques”, 3rd Edition, ISBN: 978-0-471-37644-6, 368 pages, Addison-Wesley, June 2000.

[28] Eurocontrol, Eurocontrol Experimental Centre, “Review of Root Causes of Accidents due to Design”, EEC Note No. 14/04, Project Stafbuild, Ocotber 2004, http://www.eurocontrol.int/sites/default/files/library/027_Root_Causes_of_Accidents_Due_to_Design.pdf. Visited: 14 October 2016.

[29] N. Storey, Safety-Critical Computer Systems. Addison-Wesley, 1996.

[30] Richard H. Cobb, Harlan D. Mills: Engineering Software under Statistical Quality Control, IEEE Software 7(6): 44-54 (1990).

[31] Keating, Michael, The Simple Art of SoC Design, Springer, 2011, 234p., ISBN: 978-1-4419-8586-6.

[32] McDermid, J.A. and Kelly, T.P. (2006) 'Software in safety critical systems: achievement and prediction', Nuclear Future, Thomas Telford Journals.

[33] Barnard, J., The Value of a Mature Software Process, United Space Alliance, presentation to UK Mission on Space Software, 10th May 1999.

[34] RTCA (1992) DO 178B - Software Considerations in Airborne Systems and Equipment Certification, Radio and Technical Commission for Aeronautics.

[35] German, A., Mooney, G., Air Vehicle Static Code Analysis - Lessons Learnt, in Aspects of Safety Management, F Redmill, T Anderson (Eds), Springer Verlag, 2001.

[36] Fenton, N.E., Ohlsson, N., Quantitative Analysis of Faults and Failures in a Complex Software System, IEEE Transactions on Software Engineering, 26(8), 797-814, 2000.

[37] Reifer, D.J., Industry Software Cost, Quality and Productivity Benchmarks, 2004, https://www.csiac.org/wp-content/uploads/2016/02/2004_07_01_SoftwareCosts.pdf. Visited: 14 October 2016.

[38] Hatton, L., Estimating Source Lines of Code from Object Code: Windows and Embedded Control Systems, 2005, http://www.leshatton.org/Documents/LOC2005.pdf. Visited: 14 October 2016.

[39] Ellims, M., On Wheels, Nuts and Software, in proceedings of the 9th Australian Workshop on Safety Critical Systems, Australian Computer Society, August 2004.

[40] Shooman, M.L., Avionics Software Problem Occurrence Rates, in proceedings of the 7th International Symposium on Software Reliability Engineering, White Plains, NY, 1996. pp 53-64.

[41] RTCA DO-178C, Software Considerations in Airborne Systems and Equipment Certification (Washington, DC: RTCA, Inc., December 2011).

[42] ECSS-E-ST-40C, Space engineering - Software, 06/03/2009

[43] ECSS-Q-ST-80, Space Product Assurance - Software Product Assurance, 06/03/2009

[44] NASA-STD-8719-13B: Software Safety Standard – NASA Technical Standard, NASA Office of Safety and Mission Assurance, 08/07/2004.

[45] RTCA DO-254 (EUROCAE ED-80), Design Assurance Guidance for Airborne Electronic Hardware, RTCA Inc., Washington, DC, April 2000.

[46] ARP4761; Guidelines and methods for conducting the safety assessment process on civil airborne systems and equipment. SAE International SC-18. 1996.

[47] ARP4754A; Guidelines for Development of Civil Aircraft and Systems. SAE International SC-18. 2010.

[48] ISO/DIS 26262, Road vehicles -- Functional safety. International Standard. 2011.

[49] EN 50126:2012; Railway Applications - The specification and demonstration of Reliability, Availability, Maintainability and Safety (RAMS); CENELEC; 2012

[50] EN 50128:2011; Railway Applications - Software for railway control and protection systems; CENELEC; 2011

[51] EN 50129:2003; Railway applications - Safety related electronic systems for signalling; CENELEC; 2003

[52] IEC 61508; Functional safety of electrical/electronic/programmable electronic safety-related systems; IEC; second edition, 2010.

[53] IEC 61511-1; Functional safety - Safety instrumented systems for the process industry sector - Part 1: Framework, definitions, system, hardware and software requirements. First Edition. IEC. Publication date 2003-01-30.

[54] IEC 61511-2; Functional safety - Safety instrumented systems for the process industry sector - Part 2: Guidelines for the application of IEC 61511-1. First Edition. IEC. Publication date 2003-07-04.

[55] IEC 61511-3; Functional safety - Safety instrumented systems for the process industry sector - Part 3: Guidance for the determination of the required safety integrity levels. First Edition. IEC. Publication date 2003-03-18.

[56] IEC 62061; Safety of machinery - Functional safety of safety-related electrical, electronic and programmable electronic control systems. First edition. IEC. Publication date 2005-01-20.

[57] IEC-62304; International Electrotechnical Commission (2006). "Medical device software – Software life cycle processes". INTERNATIONAL IEC STANDARD 62304, First edition, 2006-05. IEC.

[58] IEC 60880; Nuclear power plants - Instrumentation and control systems important to safety - Software aspects for computer-based systems performing category A functions. INTERNATIONAL IEC STANDARD 60880, Second edition, 2006-05-09. IEC.

[59] Kjeld Hjortnaes, ESA Software Initiative, May 7, 2003, http://klabs.org/DEI/References/Software/software.htm. Visited: 14 October 2016.

[60] Paul Cheng: Strategic testing Lessons from Satellite Failures, Feb 3, 2003.

[61] Gerard Holzmann, “Making robust software: the "Mars Code" talk”, USENIX HotDep '12, October 7th, 2012.

[62] DoD Defense Science Board Task Force on Defense Software, November 2000.

[63] NASA, NASA Office of Chief Engineer, “NASA Study on Flight Software Complexity”, March 5th, 2009, https://www.nasa.gov/pdf/418878main_FSWC_Final_Report.pdf. Visited: 14 October 2016.

[64] John A McDermid, “Developing Safety Critical Software: Fact and Fiction”, http://www.cs.york.ac.uk/hise/seminars/McDermid13Nov.ppt. Visited: 14 October 2016.

[65] European Space Agency , ESA ISVV Guide, issue 2.0, 29/12/2008.

[66] ISO/IEC 12207:2008 Systems and software engineering – Software life cycle processes.

[67] IEEE Computer Society, IEEE 1012-2004 - IEEE Standard for Software Verification and Validation.

[68] NASA, NASA IV&V Quality Manual, version Q, August 28th, 2014, https://www.nasa.gov/sites/default/files/ivv_qm_-_ver_q.doc. Visited: 22 October 2016.

[69] Silva, N.; Lopes, R., Overview of 10 Years of ISVV Findings in Safety-Critical Systems, Software Reliability Engineering Work-shops (ISSREW), 2012 IEEE 23rd International Symposium on , vol., no., pp.83.83, 27-30 Nov. 2012.

[70] Silva, N.; Lopes, R., Independent Assessment of Safety-Critical Systems: We Bring Data!, Software Reliability Engineering Workshops (ISSREW), 2012 IEEE 23rd International Symposium on , vol., no., pp.84.84, 27-30 Nov. 2012.

[71] Silva, N.; Lopes, R., 10 Years of ISVV: What's Next?, Software Reliability Engineering Workshops (ISSREW), 2012 IEEE 23rd International Symposium on , vol., no., pp.361.366, 27-30 Nov. 2012.

[72] Nuno Silva, Rui Lopes, "Independent Test Verification: What Metrics Have a Word to Say", 1st International Workshop on Software Certification (WoSoCER), ISSRE, 30 November 2011, Hiroshima, Japan.

[73] N. Silva, R. Lopes, A. Esper, R. Barbosa, "Results from an independent view on the validation of safety critical space system", DASIA 2013, 14-16 May, 2013, Oporto, Portugal.

[74] Steve EasterBrook, "The Role of Independent V&V in Upstream Software Development Processes", in proceedings, 2nd World Conference on Integrated Design and Process Technology (IDPT) Austin, Texas, December 1-4, 1996.

[75] CENELEC EN 50128: Railway applications - Communication, signalling and processing systems - Software for railway control and protection systems.

[76] ECSS-E-10-06C - Space engineering - Technical requirements specification, March 2009.

[77] ECSS-E-ST-10-03C - Space engineering - Testing, June 2012.

[78] ECSS-Q-ST-20-10C - Space product assurance - Off-the-shelf items utilization in space systems, October 2010.

[79] ECSS-Q-ST-30C - Space product assurance-Dependability, March 2009.

[80] FAA HDBK 006A - Reliability, Maintainability, and Availability (RMA) Handbook, Jan. 2008.

[81] Copeland L., Software Defect Taxonomies, http://flylib.com/books/en/2.156.1.108/1/. Visited: 22 October 2016.

[82] Lee Copeland, A Practitioner's Guide to Software Test Design, Artech House, 2004, ISBN: 9781580537322.

[83] Vallespir, Diego, Fernanda Grazioli, and Juliana Herbert. "A Framework to Evaluate Defect Taxonomies." In XV Congreso Argentino de Ciencias de La Computación, 2009. http://sedici.unlp.edu.ar/handle/10915/20983.Visited: 22 October 2016.

[84] R. Chillarege, I. S. Bhandari, J. K. Chaar, M. J. Halliday, D. S. Moebus, B. K. Ray, and M.-Y. Wong, 1992. Orthogonal Defect Classification-A Concept for InProcess Measurements, IEEE Transactions on Software Engineering, vol. 18, pp. 943-956.

[85] Kaner, C., Falk, J., Nguyen, H.Q.: Testing Computer Software (2nd. Edition). International Thomson Computer Press (1999).

[86] Beizer B.: Software Testing Techniques. Second Edition, Van Nostrand Reinhold New York (now: Coriolis Press). (1990).

[87] Binder, R.V.: Testing Object-oriented Systems Models, Patterns, and Tools. Addison-Wesley (1999)

[88] Grady, R.B.: Practical Software Metrics For Project Management and Process Improvement. Hewlett-Packard (1992).

[89] IEEE: IEEE 1044-1993 Standard Classification for Software Anomalies. Institute of Electrical and Electronics Engineers (1993).

[90] IEEE: IEEE 1044-2009 Standard Classification for Software Anomalies. Institute of Electrical and Electronics Engineers (7 January 2010).

[91] Chillarege, R.: Handbook of Software Reliability Engineering. IEEE Computer Society Press, McGraw-Hill Book Company (1996).

[92] Orthogonal Defect Classification v 5.2 for Software Design and Code, IBM, September 12, 2013.

[93] Freimut, B.: Developing and using defect classification schemes. Technical report, Fraunhofer IESE (2001).

[94] Bridge, Norm, and Corinne Miller. "Orthogonal defect classification using defect data to improve software development." Software Quality 3, no. 1 (1997): 1-8.

[95] El Emam, Khaled, and Isabella Wieczorek. "The repeatability of code defect classifications." In Software Reliability Engineering, 1998. Proceedings. The Ninth International Symposium on, pp. 322-333. IEEE, 1998.

[96] Dalal, Siddhartha, Michael Hamada, Paul Matthews, and Gardner Patton. "Using defect patterns to uncover opportunities for improvement." In Proc. Int'l Conf Applications of Software Measurement. 1999.

[97] Butcher, Mark, Hilora Munro, and Theresa Kratschmer. "Improving software testing via ODC: Three case studies." IBM Systems Journal 41, no. 1 (2002): 31-44.

[98] Leszak, Marek, Dewayne E. Perry, and Dieter Stoll. "Classification and evaluation of defects in a project retrospective." Journal of Systems and Software 61, no. 3 (2002): 173-187.

[99] R. Lutz and C. Mikulski. "Empirical analysis of safety-critical anomalies during operations", IEEE TSE, vol. 30, no. 3, March, 2004.

[100] Seaman, Carolyn B., Forrest Shull, Myrna Regardie, Denis Elbert, Raimund L. Feldmann, Yuepu Guo, and Sally Godfrey. "Defect categorization: making use of a decade of widely varying historical data." In Proceedings of the Second ACM-IEEE international symposium on Empirical software engineering and measurement, pp. 149-157. ACM, 2008.

[101] Wagner, Stefan. “Defect Classification and Defect Types Revisited.” In Proceedings of the 2008 Workshop on Defects in Large Software Systems, 39–40. ACM, 2008. http://dl.acm.org/citation.cfm?id=1390829.

[102] Walia, Gursimran Singh, and Jeffrey C. Carver. "A systematic literature review to identify and classify software requirement errors." Information and Software Technology 51, no. 7 (2009): 1087-1109.

[103] Kristiansen, J. "Using orthogonal defect classification in a Norwegian software company." Master project thesis, Norwegian University of Science and Technology (2009).

[104] Kristiansen, Jan Maximilian Winther. "Software Defect Analysis: An Empirical Study of Causes and Costs in the Information Technology Industry." (2010).

[105] J. Lemaitre and J. Hainaut, "Quality Evaluation and Improvement Framework for Database Schemas - Using Defect Taxonomies"; in Proc. CAiSE, 2011, pp.536-550.

[106] LiGuo Huang; Ng, V.; Persing, I.; Ruili Geng; Xu Bai; Tian, J., "AutoODC: Automated generation of Orthogonal Defect Classifications," Automated Software Engineering (ASE), 2011 26th IEEE/ACM International Conference on , vol., no., pp.412.415, 6-10 Nov. 2011

[107] Lopes Margarido, I., João Pascoal Faria, Raul Moreira Vidal, and Marco Vieira. "Classification of defect types in requirements specifications: Literature review, proposal and assessment." In Information Systems and Technologies (CISTI), 2011 6th Iberian Conference on, pp. 1-6. IEEE, 2011.

[108] N. Li, Z. Li and X. Sun, "Classification of Software Defect Detected by Black-Box Testing: An Empirical Study," Software Engineering (WCSE), 2010 Second World Congress on, Wuhan, 2010, pp. 234-240. doi: 10.1109/WCSE.2010.28

[109] The Institute of Internal Auditors, Practive Advisory 2320-2: Root Cause Analysis, Analysis and Evaluation, 2003, http://iia.no/wp-content/uploads/2016/02/PA_2320-2-Root-cause-Analysis.pdf. Visited: 22 October 2016.

[110] J.J. Rooney, L.N. Vanden Heuvel, Root cause analysis for beginners, Qual. Prog., 37 (7) (2004), pp. 45–53.

[111] T.O.A. Lehtinen, M.V. Mäntylä, J. Vanhanen, Development and evaluation of a lightweight root cause analysis method (ARCA method) – field studies at four software companies, Inf. Softw. Technol., 53 (10) (2011), pp. 1045–1061.

[112] F.O. Bjørnson, A.I. Wang, E. Arisholm, Improving the effectiveness of root cause analysis in post mortem analysis: a controlled experiment, Inf. Softw. Technol., 51 (1) (2009), pp. 150–161.

[113] Timo O. A. Lehtinen, Risto Virtanen, Juha O. Viljanen, Mika V. Mäntylä, and Casper Lassenius. 2014. A tool supporting root cause analysis for synchronous retrospectives in distributed software teams. Inf. Softw. Technol. 56, 4 (April 2014), 408-437.

[114] Rao, Ramaa, "Root Cause Defect Classification (RCDC) for Documentation Defects", 2014, http://www.stc-india.org/conferences/2014/presentations/Root%20Cause%20and%20Defect%20Classification%20for%20Documentation%20Bugs%20-%20Ramaa%20Rao.pdf. Visited: 22 October 2016.

[115] Kumaresh, S, Baskaran, R, "Defect Analysis and Prevention for Software Process Quality Improvement", International Journal of Computer Applications (0975 – 8887), Volume 8– No.7, October 2010.

[116] Leszak, M., Perry, D.E., Stoll, D.: A case study in root cause defect analysis. Proceedings of the 22nd international conference on Software engineering (2000).

[117] Fenton, N.E.; Ohlsson, N., "Quantitative analysis of faults and failures in a complex software system," Software Engineering, IEEE Transactions on, vol.26, no.8, pp.797.814, Aug 2000.

[118] P. Jalote, N. Agrawal, Using defect analysis feedback for improving quality and productivity in iterative software development, in: Proceedings of the Information Science and Communications Technology (ICICT 2005), 2005, pp. 701–714.

[119] B. Andersen, T. Fagerhaug (Eds.), Root Cause Analysis: Simplified Tools and Techniques, Tony A. William American Society for Quality, Quality Press, United States, Milwaukee 53203, 2006.

[120] Latino, Robert J., Latino Kenneth, C. and Latino, Mark A., Root Cause Analysis: Improving Performance for Bottom Line Results. 4th Ed., 2011, c. 280 pp., ISBN: 9781439850923, Taylor & Francis. Boca Raton.

[121] Shrouti, C., Franciosa, P., and Ceglarek, D., 2013,"Root Cause Analysis of Product Service Failure Using Computer Experimentation Technique", The 2nd Through-lifecycle Engineering Services Conference, Procedia CIRP, 11: 44-49.

[122] Kaushal Amin, Applying Root Cause Analysis to Software Defects, Software Testing Magazine, 25 June 2013, http://www.softwaretestingmagazine.com/knowledge/applying-root-cause-analysis-to-software-defects/. Visited: 22 October 2016.

[123] Robyn R. Lutz. 1993. Targeting safety-related errors during software requirements analysis. In Proceedings of the 1st ACM SIGSOFT symposium on Foundations of software engineering (SIGSOFT '93), David Notkin (Ed.). ACM, New York, NY, USA, 99-106.

[124] I. Bhandari et al., "In-process improvement through defect data interpretation", IBM Systems Journal, Vol. 33, No. 1, 1994.

[125] Allen Peter Nikora, "Software System Defect Content Prediction from Development Process and Product Characteristics," PhD Dissertation, Department of Computer Science, University of Southern California, May 1998.

[126] Raninen, Anu, Tanja Toroi, Hannu Vainio, and Jarmo J. Ahonen. "Defect data analysis as input for software process improvement." In Product-Focused Software Process Improvement, pp. 3-16. Springer Berlin Heidelberg, 2012.

[127] Otto Vinter, Using Defect Analysis as an Approach to Software Process Improvement, http://ottovinter.dk/defana.doc, 2008.

[128] Katz, Richard B., Digital Engineering Institute: Lessons Learned - klabs.org, http://klabs.org/DEI/lessons_learned/.

[129] Neufelder, A. M., The Top Ten Things that have been Proven to Affect Software Reliability, 2012 IEEE 23rd International Symposium on, Industrial Track invited talk, http://www.softrel.com/downloads/TopTen.pdf, 27-30 Nov. 2012.

[130] S. Wagner. A Literature Survey of the Quality Economics of Defect-Detection Techniques. In Proc. 5th ACM-IEEE International Symposium on Empirical Software Engineering (ISESE '06). ACM Press, 2006.

[131] S. Wagner. A Model and Sensitivity Analysis of the Quality Economics of Defect-Detection Techniques. In Proc. International Symposium on Software Testing and Analysis (ISSTA '06), pages 73-83. ACM Press, 2006.

[132] Lutz, R.R.; Mikulski, I.C., Requirements discovery during the testing of safety-critical software, Software Engineering, 2003. Proceedings. 25th International Conference on, vol., no., pp.578, 583, 3-10 May 2003.

[133] R. Lutz, Analyzing Software Requirements Errors in Safety-Critical, Embedded Systems, Proc IEEE Intl Symp Req Eng, IEEE CS Press, 1993, pp. 126- 133.

[134] R. Lutz and I. C. Mikulski, Operational Anomalies as a Cause of Safety-Critical Requirements Evolution, The Journal of Systems and Software, vol. 65 (2003), pp. 155–161.

[135] Software Engineering: Are we getting better at it?, Michael Jones, ESA Bulletin 121, February 2005, pp. 52-57.

[136] K. A. Weiss, N. Leveson, K. Lundqvist, N. Farid, and M. Stringfellow, “An Analysis of Causation in Aerospace Accidents,” Space, 2001, Aug., 2001.

[137] Jäntti, Marko, Tanja Toroi, and Anne Eerola. 2006. “Difficulties in Establishing a Defect Management Process: A Case Study.” In Proceedings of the 7th International Conference on Product-Focused Software Process Improvement, 142–150. PROFES’06. Berlin, Heidelberg: Springer-Verlag. doi:10.1007/11767718_14.

[138] LinkedIn, https://www.linkedin.com/, visited: 28 January 2017.

[139] CRITICAL Software Technology for an Evolutionary Partnership (CRITICAL STEP), Marie-Curie Industry-Academia Partnerships and Pathways (IAPP), FP7-PEOPLE-2008-IAPP, http://www.critical-step.eu/, visited: 28 January 2017.

[140] CECRIS (CErtification of CRItical Systems), Marie-Curie Industry-Academia Partnerships and Pathways (IAPP), FP7-PEOPLE-2012-IAPP, http://www.cecris-project.eu/, visited: 28 January 2017.

[141] VAL-COTS-RT - Validation of Real-Time COTS products, https://www.cisuc.uc.pt/projects/show/56, visited: 28 January 2017.

[142] AMBER - Assessing, Measuring, and Benchmarking Resilience, https://www.cisuc.uc.pt/projects/show/98, visited: 28 January 2017.

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

Annex A. Defects Assessment Questionnaire

Introduction

Dear respondent,

This survey blossoms in a very particular situation. It intends to gather the view of the industry and academia in what concerns defect analysis of critical systems, in particular the occurrence of failures in systems and software development and V&V processes. It is essential to keep in mind that we are considering **systems currently developed with utmost care**, under very strict and **mature processes and methods** and that usually require **independent assessment** in order to be fit to purpose (qualified, certified, homologated, …). These systems also follow well-defined quality assurance rules and originate a great deal of evidences that can be consulted by teams and independent assessors.

In this context, previous work [1] on the **analysis of the defects identified at a late development stage** (either at the end of a development lifecycle phase or at the end of the validation) has shown that some defects are still not caught by the traditional V&V and QA activities. These defects have been classified with ODC [2] and some improvement suggestions have been identified, to both improve the development and avoid the defects, and to improve the efficacy of the V&V activities and thus identify them within the project internal quality assurance tasks. This survey operates around some of these results, but intends to also grab additional expertise to improve the classification and the root-cause identification tasks.

Note when answering that we are not looking for basic development methods, nor basic V&V processes, but to go beyond them, although, sometimes, the simplest solutions are known and wrongly applied. **Consider the defects as defects found at a near deployment phase or after deployment.**

Acronyms and Definitions

Relevant list of acronyms and definitions:

ISVV: Independent Software Verification and Validation

ODC: Orthogonal Defect Classification. ODC is a methodology that extracts information and provides feedback about a development process from the defects that occur during the development lifecycle. (Developed at IBM Research circa 1991)

PA: Product Assurance

QA: Quality Assurance

RCA: Root Cause Analysis

SDP: Software Development Process

SW: Software

V&V: Verification and Validation

Anonymity

In order to protect individual and organization privacy, the answers to this questionnaire will remain anonymous and will not in any way be used to identify the respondents. The survey data will not identify and will not be used, either alone or with other information, to identify survey participants.

Instructions

Please respond to the following 22 questions by editing this MS Word document or the PDF version and deliver your responses preferably before **April 15th, 2016** to the following email:

nsilva@criticalsoftware.com

Please feel free to request clarifications and thank you very much for supporting me in this analysis.

Coimbra, March 24th, 2016

Nuno Silva

Critical Software SA / University of Coimbra

Coimbra, Portugal

Questions

A. General Questions

Q1: Years of Academic/Research Experience or Years of Industry experience.

[If both, please indicate both separately, e.g.: Academic: 5y; Industry: 10y]

R1: Academic/Research: _____ Industry: _____

Q2: Industries where you (the expert) have been involved? Aeronautics, Space, Defense, Automotive, Railway, Energy, Others.

[Indicate in which industries/domains you have been involved for more than one year. Academic researchers might not have this distinction, thus use "Academia"]

R2: _____

Q3: Level of importance that you give to standards utilization (according to impact in software/system development).

[Leave blank if you have no opinion]

☐ 1 - Extremely Important

☐ 2 - Very Important

☐ 3 - Somehow Important

☐ 4 – Not relevant

Q4: Level of importance of budget/financial restrictions.

[Indicate how important are budgetary restrictions when you have to develop a critical system. Leave blank if you have no opinion or don't have experience on critical systems development]

☐ 1 - Extremely Important

☐ 2 - Very Important

☐ 3 - Somehow Important

☐ 4 – Not relevant

Q5: Level of importance of schedule/timings restrictions.

[Indicate how important are schedule restrictions when you have to develop a critical system. Leave blank if you have no opinion or don't have experience on critical systems development]

☐ 1 - Extremely Important

☐ 2 - Very Important

☐ 3 - Somehow Important

☐ 4 – Not relevant

Q6: Level of importance External Assessment of the developed systems or software

[When a critical system is developed, up to what level do you consider the importance of independent assessments and the certification/qualification processes that are commonly imposed in certain domains (e.g. railway, aerospace, nuclear)? Leave blank if you have no opinion]

☐ 1 - Extremely Important

☐ 2 - Very Important

☐ 3 - Somehow Important

☐ 4 – Not relevant

Q7: Level of importance of analysing software/system errors or failures and identify their root causes

[When developing or updating a critical system and issues, bugs, failures are detected, how important do you consider that industry should go beyond just correcting the issues, for example by analysing and understanding what lead to these problems (root cause)?]

☐ 1 - Extremely Important

☐ 2 - Very Important

☐ 3 - Somehow Important

☐ 4 – Not relevant

Q8: Have you ever used Orthogonal Defect Classification (ODC)?

☐ Yes

☐ No

Q9: Have you ever used defect analysis approaches?

☐ Yes

☐ No

B. Defect Analysis Process

Considering the following process composed by four main phases:

- Prerequisites: Defects data collection and preparation, aggregation of other data if necessary, such as complexity metrics, lifecycle data, etc. A (the issues) and B (the phase where the issue was found, the phase it was corrected, the type of project, etc.) represent process inputs.
- Defects Classification: Classification of individual defects according to ODC in order to identify the defect types, defect triggers and defect impacts. Note that ODC can be adapted for specific domain and technology purposes.
- Defects Root Cause Analysis: Based on the three perspectives (defect types, triggers and impact) identify the root cause analysis of the defect groups (e.g. per type, per trigger or even per impact). Steps 4 to 6 might be considered "optional", i.e. we can apply one, two or the three root cause analysis.
- Improvements and Validation: Act upon the identified root causes, at a process, organizational or resources (human and techniques/tools) level. Measure the effects of the implemented actions. Step 10 represents the actual improvements to the systems under analysis (both environment/organization and processes). Step 11 represents adaptations and improvements to the classification process of the issues.

Note: For a more detailed description of each process step refer to Annex 1 at the end of this questionnaire.

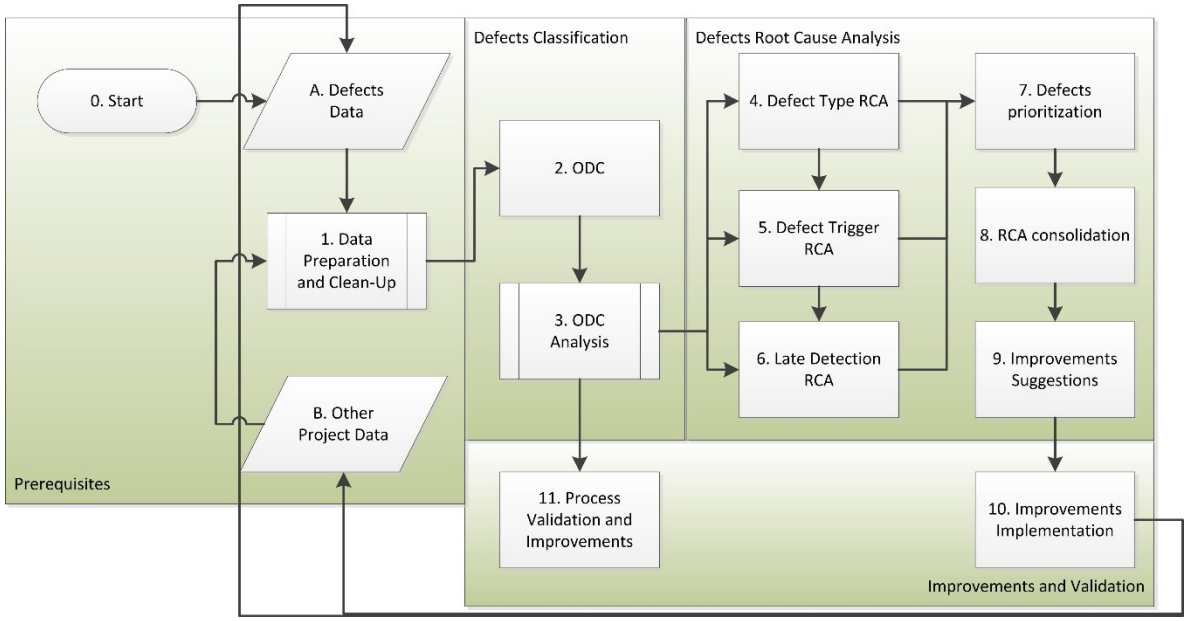


Figure 21: Defect Assessment Process[6]

Q10: What is the main strength you can point in such a defect analysis process?

[Point out and comment the main strength in the proposed tasks or the overall process]

R10:_____

_____

_____

Q11: What is the main weakness you identify in the proposed defect analysis process?

[Point out an important weakness in the proposed tasks or the overall process]

R11:_____

_____

_____

Q12: According to your experience, highlight up to three missing/important activities/steps that should be part for such a defect analysis process.

[You might highlight activities that you already perform and that bring added value to the process or you might propose the replacement of some activities in the defect analysis process]

[6] This figure is slightly different from the final process presented in Figure 10 since after the survey results have been collected we have operated modifications to the process presented in this figure.

R12.1:_____
_____

R12.2:_____
_____

R12.3:_____
_____

Q13: Would you recommend such a defect analysis process to be used in your organization?

[Leave blank if you have no opinion]

☐ 1 – Strongly Recommend

☐ 2 - Recommend

☐ 3 – Maybe Recommend

☐ 4 – Would Not Recommend

C. Defect Development Causes

Q14: From your experience and expert judgement, classify the following causes for defects introduced during "development", according to the frequency of occurrence.

[Please place an X in your classification: 1-Extremely Frequent; 2- Very Frequent; 3- Somehow Frequent; 4 – Not Frequent. If you don't understand the cause or have no opinion, leave the line blank]

Defect "Development" Cause	Classification			
	1	2	3	4
Inefficient/insufficient reviews				
Ambiguous/missing/incorrect artefacts (documentation, requirements, design, tests)				
Insufficient/Wrong tests (unit, integration, system, fault injection)				
Limitations of the tools or toolsets that deal with documentation				
Lack of Completeness and consistency of system level (or previous phases) documentation				
Oversimplified documentation planning procedures Lack of time to produce, review and accept documentation artefacts Lack of importance given to some documentation artefacts Simplification of the product assurance processes related to documentation artefacts				
Limited engineers domain knowledge – lack of appropriate skills				
Incomplete specifications in what concerns FDIR and erroneous situations				
Lack of reliability and safety culture				
Incomplete specifications in what concerns interfaces, environment and communications				
Limited definition of the operation, usability, maintainability requirements				
Lack of tools knowledge, programming languages, design languages				
Version and configuration management procedures inappropriately implemented				

Q15: Provide up to 3 examples of additional causes for defects introduced during "development", according to your experience and knowledge of critical systems.

[Please place an X in your classification: 1-Extremely Frequent; 2- Very Frequent; 3- Somehow Frequent; 4 – Not Frequent. If you don't understand the cause or have no opinion, leave the line blank]

Additional Defect "Development" Cause	Classification			
	1	2	3	4
R13.1:				
R13.2:				
R13.3:				

D. Defect Detection Causes

Q16: From your experience and expert judgement, classify the following causes for failing the detection of defects during development, according to the frequency of occurrence.

[Please place an X in your classification: 1-Extremely Frequent; 2- Very Frequent; 3-Somehow Frequent; 4 – Not Frequent. If you don't understand the cause or have no opinion, leave the line blank]

Defect "Detection" Cause	Classification			
	1	2	3	4
Lack of traceability verification culture				
Lack or inefficient usage of tools that support traceability across lifecycle phases				
Lack of appropriate test planning and test strategy definition				
Lack or inefficient testing tool and testing environment support				
Incomplete tests specification and execution				
Review process related root causes				
Documentation related root causes				
Deficient usage of tools and applicable processes				
Unclear or missing/incomplete specifications				
Ambiguous or unclear architecture definition				
Lack of usage of tools that support data and control flow analysis				
Inappropriate architecture support tools or tool usage				
Deficient specification or design artefacts				

Q17: Provide up to 3 examples of additional defect "detection" causes according to your experience and knowledge of critical systems.

[The examples can include techniques applied by you or simply known. Please place an X in your classification: 1-Extremely Frequent; 2- Very Frequent; 3-Somehow Frequent; 4 – Not Frequent. If you don't understand the cause or have no opinion, leave the line blank]

Additional Defect "Detection" Cause	Classification			
	1	2	3	4
R17.1:				
R17.2:				
R17.3:				

E. Defect Avoidance Measures

Q18: From a development perspective, classify the following measures that could avoid the introduction of defects, according to the perceived importance.

[Please place an X in your classification: 1-Extremely Frequent; 2- Very Frequent; 3-Somehow Frequent; 4 – Not Frequent. If you don't understand the cause or have no opinion, leave the line blank]

"Development" Measure	Classification			
	1	2	3	4
Define/redefine appropriate review methods, processes and tools and enforce their application at every stage of the SDP;				
Implement automated documentation generation processes and tools to avoid inconsistencies between artefacts/lifecycle phases;				
Use tools that integrate and manage all the phases of the lifecycle, such as concept specifications, requirements, architecture, source code, tests, etc.;				
Introduce/use tools with automatic validations (documentation completeness, design consistency, code analysis, control and data flow analysis);				
Provide training to the engineering teams, to improve the domain knowledge, the system or interfacing systems knowledge, standards knowledge and techniques and tools practice;				
Promote workshops or meetings to present the specifications/requirements, to discuss and clarify them before advancing to the following phase;				
Introduce additional guidelines or even specific requirements (e.g. by defining and specifying the reasoning behind the standards requirements and how to achieve them in full conformance) in the applicable standards (PA/QA, version and configuration control and development).				

Q19: Provide up to 3 examples of additional defect avoidance measures according to your experience and knowledge of critical systems.

[The examples can include measures applied by you or simply known. Please place an X in your classification: 1-Extremely Frequent; 2- Very Frequent; 3-Somehow Frequent; 4 – Not Frequent. If you don't understand the cause or have no opinion, leave the line blank]

Additional "Development" Measures	Classification			
	1	2	3	4
R19.1:				
R19.2:				
R19.3:				

F. V&V Measures

Q20: From the V&V perspective, classify the following measures according to the perceived importance in detecting defects.

[Please place an X in your classification: 1-Extremely Frequent; 2- Very Frequent; 3-Somehow Frequent; 4 – Not Frequent. If you don't understand the cause or have no opinion, leave the line blank]

"V&V" Measure	Classification			
Define appropriate test plans and strategies, especially unit and integration tests. The soundness of the test plans and strategies will reflect in the success of the validation;	1	2	3	4
Ensure appropriate (or automated) traceability analysis at every stage of the development lifecycle;				
Improve the testing completeness, coverage and reviews;				
Implement non-functional tests (fault detection, fault injection, redundancy, etc.);				
Apply or develop tools to verify and validate the implementation and design compliance.				

Q21: Provide up to 3 examples of additional effective V&V measures according to your experience and knowledge of critical systems.

[The examples can include measures applied by you or simply known. Please place an X in your classification: 1-Extremely Frequent; 2- Very Frequent; 3-Somehow Frequent; 4 – Not Frequent. If you don't understand the cause or have no opinion, leave the line blank]

Additional "V&V" Measures	Classification			
	1	2	3	4
R21.1:				
R21.2:				
R21.3:				

Q22:

Thank you for your time answering this questionnaire. If you have any additional suggestions or any relevant observations please feel free to expose them. In case you would like to receive the results of this questionnaire by email when they become available please indicate so.

R22:_____

_____

_____

_____

_____

References

[1] Silva, N.; Lopes, R., Overview of 10 Years of ISVV Findings in Safety-Critical Systems, Software Reliability Engineering Work-shops (ISSREW), 2012 IEEE 23rd International Symposium on , vol., no., pp.83.83, 27-30 Nov. 2012.

[2] Orthogonal Defect Classification v 5.2 for Software Design and Code, IBM, September 12, 2013.

Acknowledgements

This work is being partially supported by the European Project FP7-2012-324334-CECRIS (CErtification of CRItical Systems): http://www.cecris-project.eu/.

PEOPLE
IAPP MARIE CURIE ACTION

Annex 1. Defects Analysis Process Description

This annex provides a general approach for root cause analysis of critical software issues, enabling the continuous improvement of implementation and V&V at all levels (processes, techniques, tools, personnel, application of standards, organization, and so on). Figure 21 shows the general approach of a defects assessment procedure, which includes a root cause analysis and a continuous improvement procedure, described hereafter.

1. Procedure Prerequisites

The approach is based on data analysis and software engineering knowledge that require some prerequisites to be fulfilled for the correct application of the process:

0. Start:

In order to successfully perform the defects analysis it is necessary that the collected data (A. Defects Data and B. Other Project Data) contain the necessary information. This includes basic requirements such as: a) detailed information about each defect and its fix; b) knowledge of defect environment conditions, such as tools, personnel, constraints; c) engineers assessment of the defect causes; and d) phase where the defect was introduced and where it was detected.

1. Data preparation and clean-up:

Once we have the necessary data it is important to organize it and perform some anonymization if required. Data organization is essential for the next steps, since it is important to have the data in a searchable and manageable manner.

2. Defects Classification

In order to efficiently and concretely tackle the important problems of critical software engineering, the first set of activities shall focus on an orthogonal classification of the sets of defects:

2. ODC:

Perform the ODC classification on the organized dataset. Enhancements and adaptations to the ODC taxonomy can be useful depending on the nature of the defects and the domain, however, these enhancements should be quite precise.

3. ODC Analysis:

Provide a summary of the ODC analysis (results analysis and distribution). This information gives the first hints about the quality of the dataset, which can provide some feedback to the implementation and V&V teams.

3. Defects Root cause analysis

The root cause analysis is composed by several steps that include analysis of the defects types, the triggers allowing defect detection, the defects that could have been detected

earlier, and then prioritization and consolidation of these root causes leading to concrete proposed improvements:

4. Defect Type RCA:

Identify what caused the specific defect types, and try to aggregate them.

5. Defect Trigger RCA:

Identify the causes and V&V techniques or triggers that allowed the defects detection at the defect detection stage.

6. Late Detection RCA:

Identify the causes of the failures in the V&V and ISVV techniques that allowed the defects to propagate until a later stage in the development lifecycle.

7. Defects prioritization:

If required (for example to tackle the defects with the highest impact on the system, or due to the large amount of defects and respective causes) prioritize the list of defect types and triggers.

8. RCA consolidation:

From the list of defects and the corresponding root cause analysis obtained in the previous steps, consolidate the root causes into a prioritized list.

9. Improvements Suggestions:

For all the root causes, define solutions or modifications to the processes, techniques, tools, training, resources, environment or application of standards.

4. Improvements and Validation

The suggested improvements might be difficult to implement, and their efficacy can vary from team to team. They shall contribute to improve the software quality and reduce the amount of defects, different defects can then surface, and this is why this process shall have a consistent process improvement in place:

10. Improvements Implementation:

The development and V&V teams must be informed about the required changes or adjustments (**9. Improvements Suggestions**), and the organization, management and quality planning shall decide on the improvements to implement for future projects.

11. Process Validation and Improvements:

At every step, it is possible to derive improvements to the process. Such improvements can be set to adjust to the company culture, to the project environment, to the customer requirements, etc. However, it is essential to measure the effectiveness of the implementation of the results (**9. Improvements Suggestions** and **10. Improvements Implementation**) once the suggestions have been implemented and new defects (or no defects) have been collected.

Annex B. Defects Analysis Textual Responses

Collection of the textual answers and recommendations provided by the experts as a response to the Defects Assessment Questionnaire. The responses are presented as provided by the experts with no modifications to their text, excluding some typos corrected.

Process Strength, Weakness and Additional Experts Recommendations.

Q2: Technical Domain	**Q10: Process Strength**	**Q11: Process Weakness**	**Q12.1: Suggestion 1**	**Q12.2: Suggestion 2**	**Q12.3: Suggestion 3**
Space	Helps improve organizational processes based on actual feedback in a structured way.	Grouping of defects could be cumbersome, especially if not all relevant information is logged.	N/A	N/A	N/A
Fault Tolerant/ High Availability/Resilient commercial computing	Process promotes identification of weak areas of design. verification, and test.	Perhaps implicit but unclear how process feeds back to the point of defect initiation (e.g., design, documentation, etc.)	current defects vs past defects (root cause, location, escape from process, etc)	design process change recommended and implemented as a result of process	N/A
Space	It is a very thorough process that could help identifying the root causes of the defects in order to correct them and improve the overall quality of the developed software.	The process relies on the availability of good quality data about the defects which is not available very often.	Ensure management / project team commitment with the activity	Discuss findings with management / project team before presenting the improvement suggestions	N/A
Automotive, telecomm	Steps 10 resp. 11 are most important (and difficult to implement): current	The improvement process may need more attention: are there classifications	Analysis of impacts of recommended improvements on	Selection process of participants and managing the defects analysis	Monitoring the improvement implementation (again

unications, IT software	operations/processes/software must be reviewed and changed resp. updated. This takes time, which may not be available in the course of a project. However, only through improvements can the number of defect be reduced	possible, any relationship between defect and type of improvement – most important: I am missing a feedback from the defects analysis to the development process (the arrows only go back to the defects analysis (ODC)	project/schedule/cost; optimization of different improvements	operational aspects (sort of orthogonal to described process)	somewhat orthogonal): how to assure that the learnings are effectively applied to development and analysis processes.
Space	The use of the ODC (an existing methodology) associated to improvements; the correlation among the defects classification (type, trigger, detection)	It is not clear how historical data may be used in the process: to define improvements to classify defects, etc.	In step 9, I suggest you to classify the list of causes according to the person who will receive it). Each stakeholder have their own interest or position (V&V team, development team, manager, etc..)	N/A	N/A
Manufacturing automation, Automotive, Energy – nuclear	Within the context of high quality high performance organizations producing high quality low-volume products, using high quality processes, a statistical approach with aggregation of de-fect data is not as useful as a forensic analysis of each defect: • Each incident is a learning opportunity. Why wait for more data? • Too much calendar-time would be lost waiting for statistically significant data to ac-cumulate. Meanwhile the	Same as cell before	In the case of field data, as mentioned in R10, the quality and completeness of the information collected is a very significant factor.	(In the context of a product family where successive products are similar) Information about what changed when a new defect or change in defect occurred. Forensics focused on the change and its relationship to the defect would	(In the context of one of a kind products or very low volume products) a forensic interview with the people involved in the production processes.

	root cause would remain hidden and is not addressed. • In an application domain of low volume systems or devices and rapid innovation, meaningful aggregation of historic data is very difficult. • Even in organizations with high volume products, e.g., automobiles, data from the field is not collected well – take a look at their warranty claim records. Typically, a module, called an electronic control unit (ECU), is replaced. The servicing record does not include much context information, e.g., the operating history or profile of the vehicle, the specific conditions when the malfunction was first noticed. The diagnostic test codes (DTCs) are primitive.				
Academia	It seems that this process is rather complete. It is able to analyse, to classify, and to identify root-causes. Thus the completeness seems to	First, the quality of the process depends on the quality of defect Data. You should clarify how defect data should be characterized. Second, it	Quantification of the effort (cost) needed to apply this process.	Definition of Validation metric.	Think about a potential extension of ODC in order to maximize the number of defects covered (classified) by your process.

	be the main strength of the proposed pro-cess.	seems that the process you propose is rather complex. I don't know how the overall analysis will be effective. You are trying to identify defect types and trigger for all the defect data. It is a hard task, especially for the trigger identification. Third, how to validate this process. In other words, can we trust this process? Your should define (or adopt) some metrics to measure the quality of the results.			
Railway	The defects are systematically considered. The experiences from previous projects are re-used.	N/A	Identifying new defect types in the case of new technologies are used.	N/A	N/A
Space and Air Traffic Control Systems	Being able to characterise main issues in sw supplied by a particular development team sup-plier. This enables focusing the system testing prior to operational usage of particular sw.	Ensuring consistency in defect classification. Find that usually even though a definition is provided for the meaning of urgency, it is difficult to come up with common understandings across projects. Different teams use specific naming conventions e.g. vlaunch anomaly means it has to be fixed prior to launch.	Define rules within team for defect types	Define rules within the team for defects trigger	Ensure that consolidation is not impacted by timing or budget constraints. Often, the classification is well done but, for cost reasons directives such has "no more than 50 defects can be raised" impact the classification quality. As a consequence, defects are collated together and one looses the traceability with testing due to an increase in complexity regarding the defect description.

Academia, Space, Defence, Railway	Detailed classification of defects	Too much steps to achieve the final result, namely in the RCA	Impact analysis of the defect	(Safety) Classification of the defect	Defect correction validation
Aeronautics, Space	The process has a well-define	The process describes generic top level tasks such as “Data Preparation” but provides no guidance neither guidelines on how to implement those tasks. Without further guidance and/or guidelines different users may implement the same process in very different ways resulting in all from a very effective to a rather poor one.	I am not sure about what is done in “defect type”, “defect trigger” and “late detection” RCAs. To properly assess the proposed method, more information is required.	The mind-set when we are developing a new system or modifying and existing one, or even when fixing a problem in a long duration project is not the same. The team or person doing the modification may not be the same that has originally developed the system. Team dynamics, organisation culture and psychology have huge impact in the introduction of errors and the ability to detect them. I am not sure this is addressed within the activities of the pro-posed process	N/A
Defense, Telecommunications	Prerequisites steps, about quality of data	classification group could be omitted and still have a positive outcome	N/A	N/A	N/A
Aeronautics, Automotive, Railway	it is a process, i.e. if carried out properly, it provides a structured path to improvements	it is only a process, hence the real result will depend mostly on the people	N/A	N/A	N/A
Automotive, Others	N/A	possible feedback to design	N/A	N/A	N/A

Space, Railway, Energy, Finance	The main strengths are that there really is a process and that there are criteria for classification of defects (ODC).	I think the process and tasks are very dependent on having a large system where there are a considerable number of defects (whatever that means) and a lot of experience with analysing and correcting defects. In the last process, Improvements and validation, there is not mentioned explicitly the topic of retesting, i.e. how much shall be retested to make sure the defect has been really corrected (not the symptom), and that the correction has not led to unexpected problems somewhere else.	As mentioned above, strategy for retesting should be addressed	Maybe also the ITIL-processes change management, release management and configuration management should be addressed (for a system in live operation)	N/A
Space, Aeronautics, Railway	Comprehensive and detailed process. Low effort associated with the "Prerequisites" and "Defects Classification" phases (supported by the projects). Possibility to break-down the process by its phases and assign them to different teams ("Prerequisites" and "Defects Classification" to the project team, remaining to an R&D/Process	Requires all team members to master the different ODC classifications. May require one full time resource to periodically perform a sanity check to the submitted defects and their classifications in order to ensure the correctness of the defects classification.	In order to start the third phase, one needs to gather a considerable amount of information on the second phase. Given that when the second phase finishes, project team members may no longer be available for clarifications of the raised defects, an additional optional activity may be added to the second phase of the process, encompassing a periodic	N/A	N/A

	Improvement team). Usage of a well-known classification system.		sanity check of the classifications of the submitted defects in order to ensure the quality of the data to be used for the subsequent phases.		
Railway, Academia, Others	The mean strength is the application of root cause analysis itself. It helps understand the functionality of the system even better.	It is a long process, while maybe important information could get lost	Analysing whether multiple defects could have a different effect	N/A	N/A
Space	N/A	N/A	N/A	N/A	N/A
Avionics, Medical Devices, Automotive	Nice approach – the feedback to truly analyze a defect is very important, so the Improvement process is important (as usually this is missing in most organizations)	Perhaps it misses remedial training of the person responsible for the defect then assessing the adequacy of that improvement. Same for the process improvement: how we assess/know the improvement worked?	manual re-review to assess adequacy of correction, by an independent person	Check for Unwarranted Changes, e.g. additional changes made during the defect correction process which were not warranted and in fact caused unintended side effects (problems)	Missing the process improvement stage, or assessing the adequacy of the process improvement.
Aeronautics, Space, Defense, Automotive, Railway, Real-time Embedded Systems	In any domain it is always difficult to collect previous defect data base when performing RAMS analysis. A process for classifying the defects, identification of their causes and then taking action to improve the development process can be an enormous added value, in terms of reliability, cost and schedule.	Do to the budget and schedule pressure, most organizations are not willing to invest in improvement processes.	Defects data collection process could be further detailed, indicating the common or potential source of information, for example: input data from previous RAMS analysis performed in the same type or similar systems, existing test data from similar systems, existing defects data-bases per domain.	An activity for creation and updating of a defects database could be included, i.e., each project should benefit and contribute for a domain specific defect data base.	A preliminary risk analysis could be performed early in the process to identify what are the most common types of defects expected for that type of system and to prepare a set of mitigation actions that could be taken to prevent those defects from occurring

Aerospace, Defense	ODC Analysis (4 or 5 or 6)	A and B (although at first sounds easy, the data collection can be one of the items difficult to carry out completely	in some part of the process a model/snapshot view of the system could be help	N/A	N/A
Commercial Software. Consulting Clients have been in Networking, Operating Systems, Retail, Aerospace, Nuclear, Insurance.	Real data from the process yield measurements. So, we see what it is as opposed to an opinion by someone without actual insight.	Takes time and effort. It is much easier to do a shoddy job after spending a day talking to people, and forming an opinion that can be biased or influenced by the people at task.	In our process, we also have a technology assessment that goes with this.	A skills inventory is also useful.	N/A
Academia	N/A	N/A	N/A	N/A	N/A
Railway	Understanding the root causes are very important to be able not only to improve our pro-cesses, but to understand why are these processes so important. My experience is that development projects apply standards and processes only because it is mandatory, and not because they understood its usefulness. This process could help the recognition.	Currently in our project, since it is the first version of our product, we have to face with really a lot of bugs. This procedure does not talk about how to define the relevant bugs, if we would like to cut the effort required for this process, before we start the ODC classification. I feel that it is not possible to lower somehow the bugs taken into account for	Some kind of pre-analysis of bugs to decrease the effort needed for the process	N/A	N/A

	And also based on this process the best methods could be used and implemented which fit the best to the given team, since this can help also to analyse the weak points of a given team.	ODC, no one will give green light for doing this process, since it seems to be really a lot of work.			
Aeronautics, Space	The structured approach facilitates its usage	Length and amount of activities you need to perform before you have useful data	N/A	N/A	N/A
Railway, Air Traffic Control	The main strength is to combine ODC and RCA analysis; indeed they have different purposes, advantages and drawbacks, as pointed out by Chillarege itself in his original ODC paper. This allows combining a more "quantitative" analysis, as enabled by ODC, and a more qualitative one, like RCA, thus allowing capturing both more general process-level flaws and trends, and issues more related to a specific product development/verification team.	The problem in defect analysis is the manual classification it is required. ODC tends to minimize this issue, by a more "systematic" characterization, but it remains challenging to get to fast and efficient classification without an automatic or semi-automatic classification procedure.	I think that what is missing is a step for validation of the classification process itself, in terms of reliability of the classification (e.g., different people classifies in the same way), effectiveness (i.e., the classification actually serves the purpose of detecting process flaws and identify potential improvements), efficiency (i.e., classification is effective but also requires an acceptable time)	An initial tuning could help tailoring the ODC classification for the purpose of a company (see for instance this paper that applies a lightweight classification: "An Industrial Case Study of Implementing and Validating Defect Classification for Process Improvement and Quality Management"	N/A
Railway	The improvement that you gain by this analysis, in theory it should help you to build a better software	The process is time consuming, since it requires a proper classification of the issue when it came out.	N/A	N/A	N/A

	(e.g., avoid inflating the same type of bugs, improve test detection efficiency)	You can still classify the issue "later on", let's say, when you have some more time to devote to it, but it would not be effective.			
Academia. Aeronautics	Systematic defect management; Use of a defect classification; Inclusion of RCA	Need for a feedback from step 11 (the feedback from improvements comes only from step 10, but steps from 4 to 10 are not always performed in companies – process improvements may be decided even without RCA);	Defect prioritization (Step 7) may be required for process improvements even if RCA is not performed	N/A	N/A
Space, Academia	Logic + Functional Organisation	Some sub-processes might be lost or forgotten If not stated explicitly (some extra information on these steps needs to be provided)	Defect Data Interface Check (is it the root or secondary defect under study)	Defect Data related algorithm check	Defect data related configuration check
Railway, Automotive	It seems a strongly structured approach to perform defect analysis, that cover all the activities that are considered as necessary. Using a structured approach like this one will give you a guideline and greater evidence that no important information are lost in the analysis.	It seems missing a clear trace of the defects with respect to the components and versions affected by the defect. Clarifying this part will enhance the capabilities of the process.	IDENTIFICATION OF COMPONENT / SUBSYSTEM AND VERSION THAT IS AFFECTED BY THE DEFECT	N/A	N/A
Academia	Enhancements and adaptations to the ODC taxonomy should be quite precise	Bugs that could not have been found through static and dynamic analysis	Prevent defects from recurring	Select defects for further analysis	Determine if defect analysis is necessary

Space, Defense, Automotive	Continuous improvement, feedback	getting relevant measurement, getting measurement early enough, implementing the improvements (overcoming resistance)	how do you prioritize?	How do your formulate improvement suggestions?	How do you gain commitment?
presucion measuring systems, railways infrastructure and railways rolling stock.	N/A	N/A	intermediate decisions involving several stake holders (e.g. financial, technical, operational, safety, customer focus). Most of the time a defects impact and/or it's resolution is different depending on the main goals set. E.g. delivery product quickly, make it very user friendly, make it very cheap,	N/A	N/A
Space	its a looped system (should lead to improvement of the product and the process) AND it con-siders also other projects data for the analysis.	1/ No safety assessment (when relevant) 2/ I do not see any explicit preliminary analysis of the effects of the defect (e.g. impact on the system being developed such as functional errors, performance etc… and on the development operations such as delay and cost).	Unless causes and effects are so obvious so the process can be executed very fast, it is to me very important to do impact assessment at early stage because you never know how long will the analysis last. Then eventually, defect prioritization may be changed at step 7 once both effects and root causes are known.	Part of the impact assessment is the Safety assessment (mandatory for processes with safe-ty issue): It is necessary to check the impact of the defect on safety (of the development team). Important for type of process such assembly and integration, fuelling, physical testing with dangerous material or physical conditions (vacuum, vibrations)etc etc…	Short term countermeasures (work-around solution) when the defect has a significant impact on safety or project development

Railway and Space	structure and completeness	how to enforce it for real?	N/A	N/A	N/A
TeleGeoInformatics, Statistical Processing (Applications including Data Mining, Machine / Network Intelligence), Software Engineering, Academia	Defect Analysis process is built on the foundations of ODC methodology with a strong emphasis on disciplined systematic process.	The approach is trying to solidify on disciplined process on a particular methodology rather than on a process discipline. Flexibility of approaches (and methods) that can be obtained through process discipline is a key to building reliable software systems.	ODC is good for defect classification for procedural development with a waterfall process. This method does not bode well for object/ service/ aspect oriented development and also for agile methods. It is important to realize that ODC is a classification approach to group defects and has limitations in quantification of quality.	This approach may not align well to bring holistic picture for program or operations management unless the organization is developing just one product. If you have a product line, it is difficult to get a holistic grasp considering the multiple business priorities.	N/A

Q15: Additional Development Root Causes

The following text also contains the frequency of the proposed root cause as evaluated by the expert: 1-Extremely Frequent; 2- Very Frequent; 3-Somehow Frequent; 4 – Not Frequent, 0 – No Opinion.

Q2: Technical Domain	Q15.1: Suggestion 1	Q15.2: Suggestion 2	Q15.3: Suggestion 3
Space	Unmanageable software complexity (due to standards not correctly applied) (2)	Software design poorly documented, affecting software test effectiveness (3)	-
Fault Tolerant/High Availability/Resilient commercial computing	interaction with elements beyond system boundaries (2)	multiple failures (3)	-
Space	-	-	-
Automotive, telecommunications, IT software	insufficient staffing – limited resources, leading to insufficient backup know-how, review peers (2)	insufficient (management) planning to have resources available when required (3)	too high focus on functionality instead of overall system thinking (1)
Space	overconfidence from the developer team (mainly in case of re-use) (2)	manager's pressure (3)	-
Manufacturing automation, Automotive, Energy – nuclear	Weak or lacking so-called “non-functional” requirements, i.e., requirements for quality attributes (see ISO 25000 family of standards) and their transformation into explicit system constraints. (1)	Weak architectural design – it is typically a result of R13.1. Often, it is also a result of organizational culture. (1)	The way development work and affecting information is divided or scattered across organizations. Organizational division of work is not aligned with a sound architecture of the system (1)
Academia	-	-	-
Railway	Human carelessness (1)	Changing of the (railway) environment in which the system is installed. (Especially SIL0 systems) (3)	-

Annex B

Space and Air Traffic Control Systems	Team with mixed backgrounds e.g. C developer and a Java developer programming in ADA. (1)	Teams with biased backgrounds. People tend to import practices from previous working environment, and this might lead to defect introduction. (3)	Excess of interruptions at work (1)
Academia, Space, Defence, Railway	-	-	-
Aeronautics, Space	Lack of knowledge of common defects and their causes, not only the technical causes but also the organisational and even psychological causes that lead developers to inject defects. (2)	Lack of dissemination of the FDIR approach – what is the approach and why that particular approach was selected. (1)	-
Defense, Telecommunications	-	-	-
Aeronautics, Automotive, Railway	Ambiguous requirements and specifications	-	-
Automotive, Others	I believe one of the, if not the, main cause(s) for bad software (and consequently bad embedded systems) to be that companies and their (software) engineers are developing increasingly complex products whose workings they have increasing difficulty to understand. (1)	-	-
Space, Railway, Energy, Finance	Too tight schedule in combination with too many developers ("The Chinese Army approach to programming"…) (1)	No lean programming approach or wrongly implemented lean approach, i.e. the product released after a sprint can't be used for anything (1)	-
Space, Aeronautics, Railway	Unstable baselines (1)	Lack of focus on SW aspects when defining the System Software Specification (2)	-
Railway, Academia, Others	R13.1:Not well defined documentation framework (i.e, it is not clear, which document should contain certain information) (2)	Not well defined development responsibilities (1)	Permanently changing instructions from the management (1)
Space	-	-	-
Avionics, Medical Devices, Automotive	Requirements change process whereby requirements changed after implementation and incomplete regression analysis leading to incorrect operation (2)	Incorrect interface documentation, due to interface change and versioning problem between organizations (1)	code update by person less experienced with complex code and the original code was too complex, e.g.

			McCabe above 20 and complex C++ constructs (1)
Aeronautics, Space, Defense, Automotive, Railway, Real-time Embedded Systems	Lack or unefficient communication between development and V&V teams (e.g. when frequent changes of requirements exist) (1)	Schedule issues related to the development process (due to the fact that V&V phase occurs more to the end of the process it gets "squeezed" in terms of schedule, which may lead to specific or more complex scenarios (e.g. FDIR scenarios) not being tested enough to achieve a reasonable level of risk. For example, in the case of FDIR testing, it is impossible to test all the combinations of events that may trigger a reaction, but currently only basic FDIR scenarios are tested on host machines. Validation the FDIR scenarios in more representative hardware scenarios is usually not possible. (1)	Limitations of the tools and processes that deal with system configuration data. Defects originated in misconfigured system configuration data can be a source of many problems. (1)
Aerospace, Defense	Waiver solutions in the middle of the project (mainly hardware to software) (3)	-	-
Commercial Software. Consulting Clients have been in Networking, Operating Systems, Retail, Aerospace, Nuclear, Insurance.	Changed operating conditions – platform, network traffic, change of backend DB, etc.	Calibration changes that arise from changing suppliers, parts, manufacturing processes, etc.	-
Academia	-	-	-
Railway	Lack of project management knowledge, culture: tasks are not done in the right order, unclear tasks, unclear deadlines for the subtasks (2)	Lack of definition of responsibilities, and responsibility scope: who is responsible for what, who will decide go/no-go for each artefacts, who have the right to say "no". (2)	Lack of working according to the chosen development model/wrong model chosen for development (2)
Aeronautics, Space	Schedule pressure (2)	Budget constraints (1)	-

Annex B

Railway, Air Traffic Control	Programmers mistakes (3)	-	-
Railway	Insufficient or incomplete requirement elicitation. Not to confuse with requirements documentation. Here in the pure sense of elicitation, I mean that customers and software developers have few chance to meet and discuss software functionality. In many cases the customer leaves the choice to the software developers which might miss a proper vision of the software product or of the domain (4)	Customer with confused ideas on the mission of the software or its use (3)	-
Academia. Aeronautics	Pressure for release	Limited budget / Engineers overload	
Space, Academia	Lack of staff motivation (1)	Lack of interest on a given technology required to be used (2)	Short term planning (2)
Railway, Automotive	-	-	-
Academia	Coding flaws: lack of time to address the problem (2)	The design documentation could lead to incorrect source code (3)	Requirements Flaws: The requirements provided to the developer were incorrect (3)
Space, Defense, Automotive	Documents not reflecting "As build" (2)	Document not suited as means of communication (2)	V&V of models hard to assess (2)
presucion measuring systems, railways infrastructure and railways rolling stock.	Changes in scope after specification freeze (1)	Reduction of test on real system in order to reduce time/cost for developers (1)	Late availability of testing equipment for developers (2)
Space	Too much management pressure on delay disregarding the actual excessive workload on development teams (1)	Too much confidence of management in "reuse" part of an already flying system (e.g. an equipment off the shelve) conducting to deletion of some tests although operational conditions are different (rare but with catastrophic effect as for example first launch of Ariane 5) (4)	Missing detailed justification of requirements (from system level standpoint) together with lack of (inter-teams) communication resulting in a poor understanding of

			the actual system by developers. (2)
Railway and Space	-	-	-
TeleGeoInformatics, Statistical Processing (Applications including Data Mining, Machine / Network Intelligence), Software Engineering, Academia	-	-	-

Q17: Additional Defect Detection Root Causes

The following text also contains the frequency of the proposed root cause as evaluated by the expert: 1-Extremely Frequent; 2- Very Frequent; 3-Somehow Frequent; 4 – Not Frequent, 0 – No Opinion.

Q2: Technical Domain	Q17.1: Suggestion 1	Q17.2: Suggestion 2	Q17.3: Suggestion 3
Space	-	-	-
Fault Tolerant/High Availability/Resilient commercial computing	-	-	-
Space	Lack of awareness and experienced test team (2)	-	-
Automotive, telecommunications, IT software	lack of integrated specification – development – test tool system (2)	-	-
Space	-	-	-
Manufacturing automation, Automotive, Energy – nuclear	Ambiguity in requirements, including quality requirements (see ISO 25000 family of standards). (1)	Lack of information about what changed when. (2)	-
Academia	-	-	-
Railway	Human carelessness (1)	-	-
Space and Air Traffic Control Systems	Lack of communication (1)	Lack of domain knowledge in teams (1)	Management pressure to keep in budget (3)
Academia, Space, Defence, Railway	-	-	-
Aeronautics, Space	Insufficient incremental testing, e.g. jumping from unit tests to system tests without	Long execution time of test procedures, which increases the costs of –non-regression verification and therefore limits	Lack of effective validation facilities for system testing – either lack of automation, lack of numeric (i.e. simulated) benches,

	sufficient test campaigns in between. (2)	the frequency at which non-regression verification is performed) (2)	bogus validation facilities, complex test languages or test libraries, etc. (2)
Defense, Telecommunications	-	-	-
Aeronautics, Automotive, Railway	-	-	-
Automotive, Others	-	-	-
Space, Railway, Energy, Finance	Little knowledge of OS constraints / inherent faults (maybe this has to do with testing environment?) (1)	-	-
Space, Aeronautics, Railway	Defects on the Validation Environment that hide defects on the system (3)	-	-
Railway, Academia, Others	Not well defined and not clearly understood verification processes (2)	Lack of verification culture overall (2)	Lack of support from the management (3)
Space	-	-	-
Avionics, Medical Devices, Automotive	Usually weak LLR's implemented by person different than System or HLR writer (1)	Safety assessment missing hence missing derived requirements (2)	-
Aeronautics, Space, Defense, Automotive, Railway, Real-time Embedded Systems	Lack of a standard for performing independent verification and validation (most domains don't have it) (1)	Excessive number of "untestable" requirements due to complexity of some system features, which indicates architectural problems, i.e., the system was not conceived with a testability mindset. (1)	Lack of participation of V&V experts during the early design phase of the system (e.g. very few systems have been designed from scratch with built-in fault-injection capabilities). (1)
Aerospace, Defense	-	-	-
Commercial Software. Consulting Clients have been in Networking, Operating Systems, Retail, Aerospace, Nuclear, Insurance.	Lack of a reference model – especially for incremental releases.	-	-
Academia	-	-	-

Annex B

Railway	Lack of project management knowledge, culture: tasks are not done in the right order, unclear tasks, unclear deadlines for the subtasks (2)	Lack of working according to the chosen development model/wrong model chosen for development (2)	Evaluation of verification results too late / communication problems between development and verification teams (2)
Aeronautics, Space	-	-	-
Railway, Air Traffic Control	Inappropriate choice of (the mix of) testing and (automated) analysis techniques (1)	Unclear separation of the role of tester with respect to developers/designers/analysts (lack of independence) (1)	-
Railway	-	-	-
Academia. Aeronautics	-	-	-
Space, Academia	Inexperienced staff (1)	Tight deadlines (schedule) (1)	Unfamiliar project (1)
Railway, Automotive	-	-	-
Academia	Requirement Related root causes (3)	PM may not be analysing all possible risks (3)	Project control not exercised properly / Monitoring milestones not done (2)
Space, Defense, Automotive	Configuration management of run-time data (2)	-	-
presucion measuring systems, railways infrastructure and railways rolling stock.	Test in real environment (2)	Test with end- user (2)	-
Space	-	-	-
Railway and Space	-	-	-
TeleGeoInformatics, Statistical Processing (Applications including Data Mining, Machine / Network Intelligence), Software Engineering, Academia	-	-	-

Q19: Additional Development Measures

The following text also contains the relevance of the proposed measures as evaluated by the expert: 1-Extremely Relevant; 2- Very Relevant; 3-Somehow Relevant; 4 – Not Relevant, 0 – No Opinion.

Q2: Technical Domain	Q19.1: Suggestion 1	Q19.2: Suggestion 2	Q19.3: Suggestion 3
Space	Allocate more effort (money) to early verification! (1)	-	-
Fault Tolerant/High Availability/Resilient commercial computing	-	-	-
Space	-	-	-
Automotive, telecommunications, IT software	Get Management involvement and commitment for strict review processes (as required in "Define/redefine appropriate review methods, processes and tools and enforce their application at every stage of the SDP;" (1)	work with customers and regulator to enforce strict review processes/development processes/documentation processes (2)	work with relevant industry to enforce tool/documentation/review/development/specification standards (3)
Space	-	-	-
Manufacturing automation, Automotive, Energy – nuclear	Get the right requirements, esp. quality requirements. (1)	Transform quality requirements into system architectural constraints. (1)	Design the development process such that requirements-related questions are answered early in the development cycle (e.g., iterative evolutionary development process) (1)
Academia	-	-	-
Railway	Cross code reviews between two programmers, or strong code inspection. (1)	-	-
Space and Air Traffic Control Systems	Lack of leadership to communicate uniform verification approach (people with different backgrounds are not willing to accept other peoples approaches...) (1)	Simplification of traceability processes (3)	-

Academia, Space, Defence, Railway	-	-	-
Aeronautics, Space	Provide practical training on common error causes and on the mechanisms available to avoid or compensate them (2)	Clearly define and disseminate the FDIR approach not only describing it but explaining why that particular approach has been selected. (1)	Develop a modular architectural concept that clearly maps the concepts of Detection, Isolation and Recovery – often FDIR is, misleadingly, taken as a "magic" component that one adds to an architecture. (2)
Defense, Telecommunications	-	-	-
Aeronautics, Automotive, Railway	-	-	-
Automotive, Others	The most important measure would probably be to aim at producing a system that is simple enough to be well understood. (1)	The next-important measure would be to make full formal requirements specification mandatory for critical systems. This would throw the vast majority of companies out of business, eventually leading to better products. (1)	-
Space, Railway, Energy, Finance	Test based approach to software development (3)	"Buddy" programming / unit testing (1)	-
Space, Aeronautics, Railway	Review/contribute to the definition of the System Software requirements (2)	-	-
Railway, Academia, Others	Promote meetings not just to present the requirements, but for the members of different development teams / developers and testers, on different development stages, in order to ensure that the original goals are achieved / or if not, could they perhaps be modified (iterative development) (3)	-	-
Space	-	-	-
Avionics, Medical Devices, Automotive	Linking all code constructs to tests, e.g. DO-178C DAL A, B, C (1)	Reducing code complexity, using automated tools like LDRA, PRQA, (2)	Mandating MISRA C/CC++ automated static analysis test before independent code peer review. Having ONE reviewer, not one hundred. "One great reviewer is

			better than 100 good reviewers" – Vance Hilderman quote (1)
Aeronautics, Space, Defense, Automotive, Railway, Real-time Embedded Systems	Most safety-related industrial standards are typically not freely available. (1)	Lack of harmonization between the standards of the several safety-related domains (each domain defines its own processes), which makes it difficult to re-use process and tools across the different domains. (1)	Most of the standards have gaps. It is not always clear for the organizations the way to apply certain process and rules that are not defined in details by the standards, which requires support from certification authority representatives or certification agency. (1)
Aerospace, Defense	People in charge of this subject with correct (and better) skills (2)	-	-
Commercial Software. Consulting Clients have been in Networking, Operating Systems, Retail, Aerospace, Nuclear, Insurance.	Regularly review the ODC defect profiles with the teams (1)	Create a clear plan on what tests needs to be automated, versus kept manual (1)	-
Academia	-	-	-
Railway	-	-	-
Aeronautics, Space	-	-	-
Railway, Air Traffic Control	Improve requirements specification and validation (2)	Enforce design partitioning, modularity and reuse (2)	Training about basic software engineering principles (3)
Railway	Analysis of the post- delivery issues (4)	-	-
Academia. Aeronautics	-	-	-
Space, Academia	Groups of small engineers shall be trained (1)	Small groups of workshops or meeting shall be preferred. (1)	Tailor and/or develop project specific measures whenever feasible. (1)
Railway, Automotive	-	-	-
Academia	PM should have the overall control of the project (3)	Skilled programmers are to be employed for tasks in the critical path (3)	PM / PL should always have some buffer while planning for external dependencies (4)
Space, Defense, Automotive	Order feature development according to risk analysis (2)	Iterative development (short cycles)	-

presucion measuring systems, railways infrastructure and railways rolling stock.	Introduce and keep quality milestones (2)	-	-
Space	-	-	-
Railway and Space	-	-	-
TeleGeoInformatics, Statistical Processing (Applications including Data Mining, Machine / Network Intelligence), Software Engineering, Academia	-	-	-

Q21: Additional Verification and Validation Measures

The following text also contains the relevance of the proposed measures as evaluated by the expert: 1-Extremely Relevant; 2- Very Relevant; 3-Somehow Relevant; 4 – Not Relevant, 0 – No Opinion.

Q2: Technical Domain	Q21.1: Suggestion 1	Q21.2: Suggestion 2	Q21.3: Suggestion 3
Space	Thorough unit testing, specified against an actually documented detailed design (not against the code itself) (2)	-	-
Fault Tolerant/High Availability/Resilient commercial computing	-	-	-
Space	-	-	-
Automotive, telecommunications, IT software	Improve completeness of reviews (extension of above, as I feel reviews are more valuable than testing completeness or coverage) (1)	involve customer/requirements team in review and testing efforts (2)	-
Space	apply ISVV - independence is quite important (1)	have a trained and motivated team with skilled for finding errors (1)	-
Manufacturing automation, Automotive, Energy – nuclear	Validate requirements through interaction with experts (1)	Validate the decomposition and derivation of requirements and their allocation to various elements in the architecture. Apply rules of composition to ensure nothing is lost in the flow-down. (1)	Review, inspection and analysis for unwanted behaviour, e.g., through the application of advanced hazard analysis techniques such as STAMP/STPA (1)
Academia	-	-	-
Railway	Cross code reviews between two programmers, or strong code inspection. (1)	-	-
Space and Air Traffic Control Systems	Simple but strict configuration control policies (4)	Reviews where the developer asks the reviewer to explain the design (4)	Use of code linting tools (2)

Academia, Space, Defence, Railway	-	-	-
Aeronautics, Space	-	-	-
Defense, Telecommunications	-	-	-
Aeronautics, Automotive, Railway	-	-	-
Automotive, Others	The entire development process of safety-critical systems, including all certification-relevant artefacts, documentation, source code etc, should by law be required to be public and accessible to scrutiny by anyone via internet. (1)	-	-
Space, Railway, Energy, Finance	Informal "rainy day" testing in addition to the formal tests (in case something has been over-looked). (2)	-	-
Space, Aeronautics, Railway	Clear definition of the of Unit/Integration tests and Functional tests, including their goals and place in the overall SDP (2)	Introduce formal or semi-formal verification of the SW specification in the Verification process (2)	Use tools that integrate and manage all the phases of the lifecycle, such as concept specifications, requirements, architecture, source code, tests, etc.; (2)
Railway, Academia, Others	Use "creative", informal methods (i.e., analyse what could go wrong in the system) beside of formal ones. (3)	Consult regularly (but not too deeply, in order to ensure the independence) with the validator and / or assessor about requirements coming from standards. (3)	-
Space	-	-	-
Avionics, Medical Devices, Automotive	Check LLR to code robustness (2)	Decision Condition coverage tracing to LLR's (1)	-
Aeronautics, Space, Defense, Automotive, Railway, Real-time Embedded Systems	Evaluation of the testability of the architecture and requirements early in the process through the involvement of the V&V experts. During the V&V phase some requirements are simply not possible to test and verified through code	Lack of planning and definition of processes for performing non-functional tests. Most standards simply mention that these tests must be performed, but do not detail them, e.g. process for performing robustness testing. These tests are normally	-

	inspections (this is especially applicable to the space domain). (1)	performed ad-hoc in parallel to the functional tests. (1)	
Aerospace, Defense	the whole system team must be involved in the V&V process, at least to testimony of its part of the system "passed" in the tests (2)	-	-
Commercial Software. Consulting Clients have been in Networking, Operating Systems, Retail, Aerospace, Nuclear, Insurance.	Compare release to release ODC metrics to identify trends (1)	Establish with ODC analysis that current release is better than previous release!!!! Very Important (1)	-
Academia	-	-	-
Railway	Appropriate trainings for the test team about the system under test and the related domain. My experience is if they clearly understand the system, they can use the verification techniques more accurate and they have more motivation. (1)	Trainings for the test team about the defined verification and validation methods and the related standards, justification of the used methods. My experience is if they clearly understand what is the goal, they can use the techniques more accurate and they have more motivation. (1)	Develop a good communication with the development team, considering independency (2)
Aeronautics, Space	-	-	-
Railway, Air Traffic Control	Define “quantitative” test planning strategies to best allocate efforts (i.e., prioritize functions/components) (4)	Improve testing accounting for operational phase expected usage (i.e., operational/reliability testing) – Exploit historical data to assess usage profiles and corresponding tests (3)	Usage of ASA (automated static analysis) for code sanitization (2)
Railway	-	-	-
Academia. Aeronautics	Apply model-driven techniques for early defect detection	-	-
Space, Academia	Implement functional tests for validation (1)	Record and save the results automatically (1)	PA review of V&V Measures (1)
Railway, Automotive	-	-	-
Academia	Do adequate testing (2)	Eliminating escaping defects (3)	Preventing the occurrence of an individual defect or group of defects (3)
Space, Defense, Automotive	Continuous integration and test (1)	Tests driven development (1)	-

presucion measuring systems, railways infrastructure and railways rolling stock.	Clarify roles in order to have one person at least focussing on V&V (1)	-	-
Space	-	-	-
Railway and Space	-	-	-
TeleGeoInformatics, Statistical Processing (Applications including Data Mining, Machine / Network Intelligence), Software Engineering, Academia	-	-	-

Annex C. Summary of the Results of the Survey

This annex presents the summary of the survey results as provided by the experts. The data have been simplified and harmonized for data processing and analysis. Data from the experts experience is presented separately for anonymity reasons.

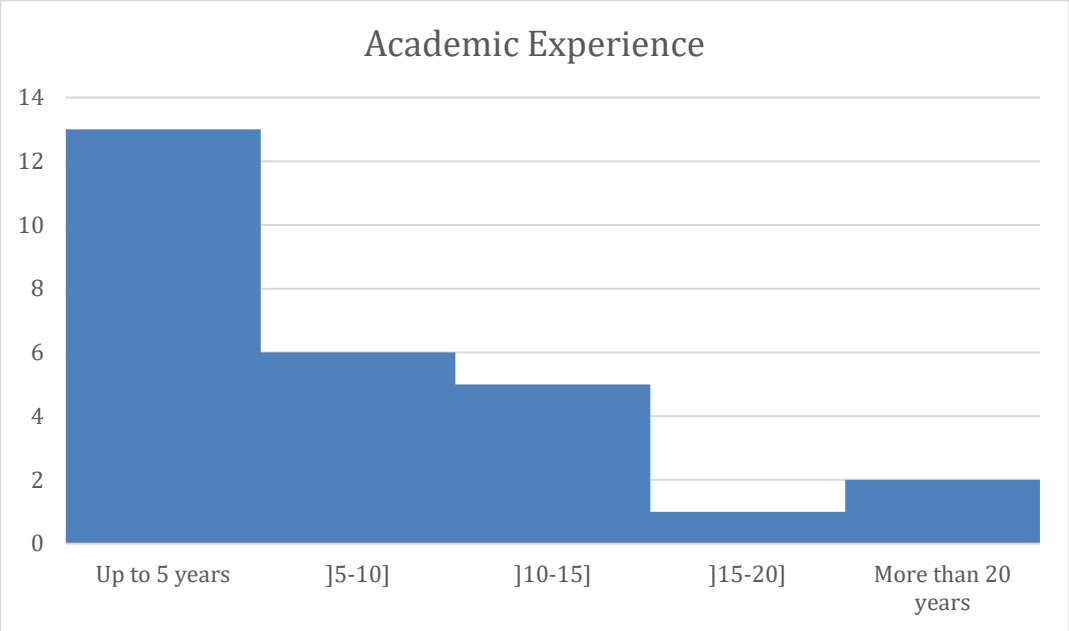
The data presented in the table can be understood as follows:

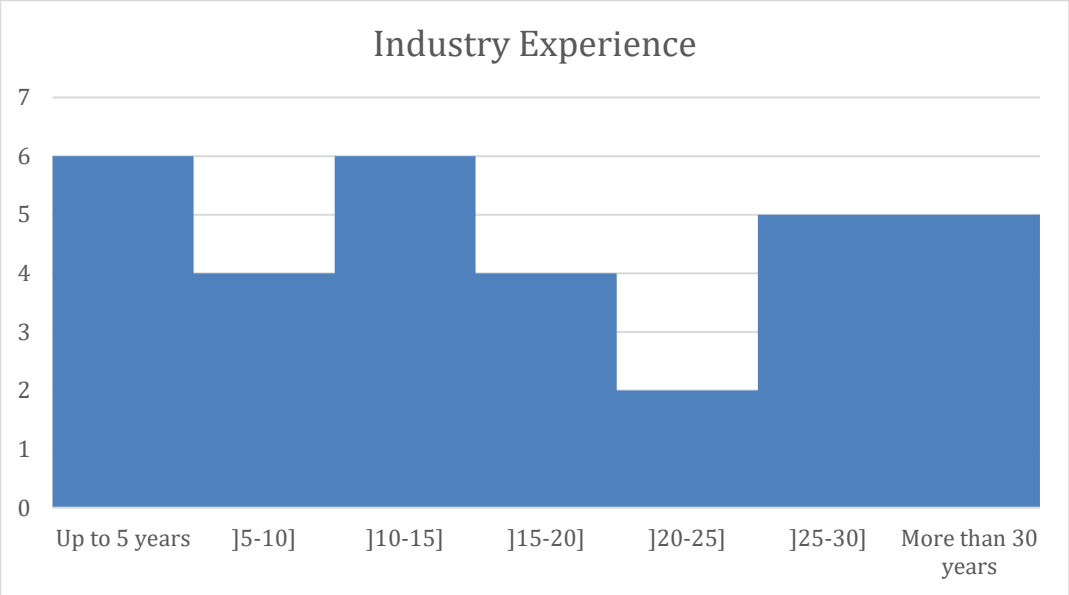
- For questions Q2, Q8 and Q8, a 1 represents “Yes”, a 0 represents “No”;
- For questions Q3 to Q7:
    - 1 - Extremely Important
    - 2 - Very Important
    - 3 - Somehow Important
    - 4 – Not relevant
    - Empty – No opinion
- For the remaining questions:
    - 1 - Extremely Frequent/Relevant
    - 2 - Very Frequent/Relevant
    - 3 - Somehow Frequent/Relevant
    - 4 – Not Frequent/Relevant
    - Empty – No Opinion.

Response ID	Q2: Academia Domain	Q2: Transportation Domain	Q2: Aerospace Domain	Q2: Other Domain	Q3: Standards Importance	Q4: Budget Importance	Q5: Schedule Importance	Q6: External Assessment Importance	Q7: RCA Importance	Q8: Used ODC?	Q9: Used Defect Analysis?	Q13: Recommend Process	Q14a: Inefficient Reviews	Q14b: Artefacts Problems	Q14c: Incorrect Tests	Q14d: Docu Tools Limitations	Q14e: Sys Docu Inconsistencies	Q14f: Docu Processes Overlooked	Q14g: Limited Domain Knowledge	Q14h: FDIR Specs Missing	Q14i: Lack Reliab. Safety Culture	Q14j: Incomplete Interface Specs	Q14k: Insufficient Maint, Oper, Usab Specs	Q14l: Lack Tools and Languages Knowl	Q14m: CM and Versioning Problems	Q16a: No Traceability Culture	Q16b: No Traceability Tools	Q16c: No Test Planning	Q16d: No Testing Tool/Env.	Q16e: Incomplete Test Spec	Q16f: Defective Review Process	Q16g: Defective Docu	Q16h: Bad Tools Usage	Q16i: Incomplete/Bad Specs	Q16j: Defective Architecture	Q16k: No Tools for Data Flow Analysis	Q16l: Inappropriate arch supp tools	Q16m: Deficient Design Specs	Q18a: Better Processes/Review	Q18b: Auto Docu Generation	Q18c: Tools for full lifecycle	Q18d: Auto Validation Tools	Q18e: Engineering Training	Q18f: Specifications Meetings	Q18g: Standards Guidelines	Q20a: Test Plans	Q20b: Traceability	Q20c: Test Coverage	Q20d: Non-functional Tests	Q20e: Tools for design compliance
1	1	0	1	0	1	3	3	2	1	0	0	4	4	3	2	4	2	3	3	2	3	1	2	4	2	4	4	2	3	2	4	3	4	2	4	4	4	2	2	3	4	4	2	4	3	2	3	2	3	4
2	0	0	0	1		1	1		2	1	0	3	4	1	2		2		4	4	4	2	3	4	1	4	3	1	2	2	3	3	3	2	2	3	3	2	3	1	1	1	3	3	4	1	3	3	3	3
3	0	0	1	0	1	1	2	1	2	0	0	2	2	1	2	3	2	2	2	2	2	3	4	4	4	2	3	2	3	2		2	3	2	3	3	3	2	1	1	2	2	2	2	3	2	1	2	3	2
4	0	1	0	1	2	2	1	2	1	0	0	1	1	2	3	2	3	3	3	2	2	3	2	4	4	3	3	3	3	3	2	2	4	3	3	2	2	3	1	2	2	2	3	3	3	2	2	3	3	2
5	1	0	1	0	2	1	1	1	1	0	0		2	1	2	4	2	1	3	2		2	1	4	2	2	2	3	3	1	3	3	3	1	3	4		2	1	2	3	3	2	2	3	3	3	1	1	2
6	1	1	0	1	3	1	1	1	1	1	1	4	2	1	2	3	1	4	1	1	1	1	1	4	3	2								1	1		2	1	4	4	4	4	2	1	3						
7	1	0	0	0	2	2		1	2	1	1	3	2	1	2	1	2	2	3	4	3	3	2	1	3							2		2	3		4	4								1	1	1	2	2
8	0	1	0	0	1	2	2	1	1	0	0	1	3	2	3	1	3	4	4	3	2	2	4	4	1	4	4	4	4	4	2	4	4	4	4	4	4	4	1	4	3	3	1	1	1	3	4	4	3	4
9	0	0	1	1	2	1	2	1	1	0	1	2	2	1	3	3	1	2	3	3		2	1	4	3	1	2	3	4	4	1	2	4	2	2	2	4	2	1	3	3	3	1	1	3	1	1	2	1	3
10	1	1	1	1	1	2	2	1	3	0	1	3	2	2	3	1	2	3	4	2	4	2	3	4	4	2	3	3	3	3	2	2	2	3	3	2	2	3	1	3	3	2	3	2	2	3	2	2	2	2
11	0	0	1	0	1	2	2	3	1	1	0	3	2	2	2	4	3	2	2	3	1	3	3	4	2	2	2	2	3	3	3	2	2	2	4	4	2	2	4	3	2	1	2	2	2	2	2	2	3
12	0	0	0	1	1	2	3	3	1	0	1	3	2	2	3	3	1	2	1	1	4	3	3	4	4	3	4	3	3	3	3	3	4	1	2	3	3	2	3	4	2	2	2	1	2	1	1	2	1	2
13	1	1	1	0	3	2	2	1	1	0	1		3	2	1	3	1	3	3	2	2	1	2	3	4	2	3	1	2	3	3	2	4	1	3	2	4	2	2	2	2	2	2	1	3	2	3	1	2	1
14	1	1	0	1	3	1	1	2	1	0	0		2	1	2	3	2	3	2		1	1	2	3	3						1				1				1												
15	0	1	1	1	2	2	1	2	2	0	1	3	1	1	1	2	1	1	1	1	2	1	1	2	2	2	2	1	2	1	2	2	2	1	1	2	2	2	2	2	2	2	1	1	1	2	1	2	1	2

16	0	1	1	0	2	2	3	1	2	0	0	1	3	1	3	2	2	4	4	3	4	3	3	4	3	4	2	4	3	4	3	4	3	3	3	3	3	3	3	2	1	2	3	2	4	1	1	1	2	2
17	1	1	0	1	2	2	1	2	2	0	0	2	1	1	2	3	1	2	3	2	2	1	2	3	2	2	3	2	3	2	2	2	3	1	1	3	3	2	2	3	3	3	3	3	2	3	2	2	3	3
18	1	0	1	0	2	2	1	3	2	0	0			1	1	2	1	2	3	3	2	1	1	3	3	1	2	2	2	1	2	3	3	3	1	3	3	2	2	1	2	2	1	1	1	1	1	1	1	2
19	0	1	1	1	1	2	2	1	1	0	1	1	2	3	3	4	2	2	4	2	3	1	2	4	4	1	2	2	3	1	1	2	3	1	2	3	2	1	1	3	3	4	1	2	2	2	1	1	3	3
20	0	1	1	1	1	1	1	1	1	0	1	1	1	1	1	4	4	1	4	4	1	2	4	4	1	1	1	4	3	1	1	1	3	1	1	1	3	1	1	1	1	1	1	1	1	1	1	1	1	1
21	1	0	1	0	1	1	1	2	1	0	0	2	2	1	2	2	1	2	3	3	2	3	2	3	2	2	1	3	3	3	2	2	3	2	2	3	3	3	1	2	2	3	3	3	2	2	2	2	2	3
22	0	0	1	1	2	2	1	2	1	1	1	2														2	3	1	1	1	2	1	3	3	3	2	3	2	3	3	3	2	1	1	1	1	1	1	2	3
23	1	0	0	0	2			3	2	0	0	3	3	2	2	2	2	2	3	3	3	4	3	4	4	2	3	3	4	1			3	3	2	3	3	3	2	2	2	2	4	3	3	2	2	2	4	3
24	0	1	0	0	1	2	1	1	1	0	0	1	2	1	3	4	2	1	3	2	2	2	3	3	2	1	2	2	3	2	3	1	2	1	1	2		2	1	2	2	2	1	1	1	1	1	2	1	2
25	0	0	1	0	1	2	2	1	2	1	1	3	2	2	2	3	3	2	3	2	2	3	3	3	2	3	3	2	3	2	3	3	3	2	3	3	3	2	3	3	3	3	2	2	2	2	2	2	2	2
26	1	1	0	1	1	2	2	2	1	1	1	2	3	1	2	4	3	3	4	1	1	2	2	3	4	1	2	1	3	1	2	3	3	2	3	2	4	2	2	2	3	2	4	4	3		3	2	3	2
27	0	1	0	0	2	2	2	1	2	0	0	3	3	4	3	2		4	3		4	4	4	1	2	3	2	4	3	4	4	4	4	4	3		2	3	3	3	3	2	4	3	3		3	3	4	3
28	1	0	1	0	1	2	3	1	1	1	1								3		2	2	3	3		3	2	1	4	1	1			1	3				1		3	1	2		1		1	1	1	
29	1	0	1	0	1	3	2	1	1	0	0	2	1	2	2	3	3	1	1	2	1	1	2	3	2	2	3	1	3	3	1	2	2	1	1	3	3	2	1	3	3	3	1	1	1	1	1	1	1	3
30	0	1	0	0	1	3	2	2	1	0	1	2	2	3	2	4	2	1	3	4	2	3	3	4	3	2	4	2	3	2	4	3	3	1	4	3	3	2	1	4	3	3	2	2	2	1	3	2	2	3
31	1	0	0	0	2	1	2	2	2	0	0	1	1	3	2	2	3	2		1	2	3	2	3	1	2	3	2	1	3	2	2	3	2	1	2	2	3	1	3	3	2	2	3	3		1	2	2	1
32	1	1	1	1	1	1	1	2	2	0	0	2	1	3	3	4	1	2	2	1	3	3	3	3	3	3	1	3	2	2			3	4	4	3	1	4	2	4	4	4	3	3	2	3	3	1	1	3
33	1	1	0	0	1	3	2	1	1	0	1	1	3	2	2	2	2	2	4	1	3	2	1	4	2	3	3	4	3	2	3	2	4	2	3	4	4	4	1	2	2	2	2	3	3	1	2	3	2	3
34	0	0	1	0	1	3	2	1	1	0	1	3	3	2	2	3	2	3	3	2	4	2	2	4	4	4	4	3	4	4	4	3	4	2	2	2	3	3	4	3	2	3	1	1	1	2	3		1	3
35	1	1	1	0	1	2	1	1	1	1	1	2	2	1	1	2	1	2	2	1	2	1	1	4	3	2	2	2	2	2	2	2	3	1	1	2	3	2	1	2	2	2	1	1	2	1	1	1	2	3
36	1	0	0	1	3	1	1	1	2	1	1		3	2	2	1	1	1	3		2	2	2	2	2	2	1	1	2	2	3	3	3	1	1	3	1	2	2	3	3	3	1	1	2	3	3	1	1	3

These charts show the amount of experts per experience range.





Annex D. Example of Data Collection Template

This annex presents the used data collection template and explains each of the data fields considered in the template.

Item	Description	Example
Number	Unique identifier for the issue	001
Project	Identifier of the project where the issue comes from (Mission 1, Mission 2, etc.)	SYS02
Subsystem	Subsystem or component within the project (star tracker, GPS, Laser system, etc.)	SS-02
Domain	Business domain applicable to the project of the issue (space, automotive, aeronautics, etc.)	Space
System Type	Definition of a system type applicable to the domain (Data Processing, User Interface, Database, Communications System, etc.)	OB ASW
Issue Title	Short issue title	Conflict between ASW and SA
Desc	Detailed description of the issue. Must contain enough details to be able to be analyzed or, eventually, to be processed in an automated way in the future.	The Application Software (ASW) contains an implementation (source code) that differs from the defined Software Architecture (SA). The differences are the following: Listing of the differences and inconsistencies.
Classification	Classification of the Severity of the issue. (Minor, Major, Catastrophic, Comment, etc.)	Minor
Problem Type	Originally classified issue type. This field has been used to contain the ISVV classification made by the team that raised the issue.	External Consistency
Phase Detected	Lifecycle phase where the issue has been detected. (Requirements, Design, Implementation, Testing, Operations, etc.)	Design
Phase Applicable	Lifecycle phase where the issue has been actually introduced. (Requirements, Design, Implementation, Testing, Operations, etc.)	Requirements
Defect Type	ODC Classification according to the defect type taxonomy defined in section 5.3.2	Function/Class/Object
Defect Trigger	ODC Classification according to the defect trigger taxonomy defined in section 5.3.3	Concurrency
Defect Impact	ODC Classification according to the defect impact taxonomy defined in section 5.3.4	Reliability
Comment	Field used to add comments related to the defect. This might be useful to complement the defect information or to justify some of the classification of the other fields.	The architecture should be updated to reflect the source code (or vice-versa).
Notes	Notes about the defects classification. This field has been used to enhance the defects classification taxonomy, to document the classification doubts and to propose new taxonomy items or taxonomy items merges.	ODC Classification reviewed and confirmed.
Activity	V&V activity that lead to uncover the issue. In our case this was the ISVV task that was applied and that lead to the raising of the issues (requirements verification, code inspections, testing, etc.)	Requirements verification