



CHALMERS
UNIVERSITY OF TECHNOLOGY



UNIVERSITY OF GOTHENBURG

Enhancing Coordination, Traceability and Compliance Checking in Systems Development

Master's thesis in Computer science and engineering

MICHAEL OSEI ADUAMAH

MICHIALE HADGU ARAYA

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2024

MICHAEL OSEI ADUAMAH & MICHIALE HADGU ARAYA

© MICHAEL OSEI ADUAMAH & MICHIALE HADGU ARAYA, 2024.

Supervisor: Eric Knauss, Computer Science and Engineering
Examiner: Jennifer Horkoff, Computer Science and Engineering

Master's Thesis 2024
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Gothenburg, Sweden 2024

Michael Osei Aduamah & Michaile Hadgu Araya
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg

Abstract

Context : Delivering safety critical software solutions that are dependable, and of high quality has been shown to depend heavily on ensuring standards compliance in system development. Ensuring adherence to safety standards such as ISO 26262 demands precise documentation and traceability of all work products and roles responsible throughout the entire design and development phase. The manual verification of compliance against numerous requirements, including the scrutiny of work products within process entities, presents significant challenges, such as labor intensity and susceptibility to errors. This context emphasizes the critical need for tools that can maintain detailed documentation of design choices, safety analyses, verification and validation results, and changes over the product life cycle, ensuring that all work products are traceable to requirements and justified as part of a safety case.

Objective : This study aims to address these challenges by introducing an approach that integrates the coordination capabilities of the BOMI Model with the traceability and compliance checking functionalities of TReqs, focusing on the identification and exploitation of the appropriate boundary objects(work products), roles(responsible parties) required by a particular standard to achieve compliance. The objective is to automate the compliance checking process, thereby reducing the burden on process engineers and improving the accuracy of compliance verification.

Method : To tackle the inefficiencies of manual compliance checking, we introduce an enhanced version of TReqs, a traceability tool, and Boundary Objects between Methodological Islands(BOMI), a modeling approach designed to enhance coordination in large scale agile system development. These tools are pivotal in addressing the requirement for traceability as stipulated by ISO 26262, ensuring that all work products are accounted for and verifiable throughout the development lifecycle. The study employs a design science research approach, iterating through the development and evaluation of tool enhancements through workshops, surveys, focus groups, interviews, and literature reviews.

Results : This study successfully developed an enhanced version of the TReqs tool, incorporating automated compliance check capabilities. This enhancement significantly reduces the manual effort required for compliance verification, thus addressing one of the primary challenges identified in the initial problem statement. The automated compliance check functionality was designed to interpret and evaluate the compliance of work products against the requirements of ISO 26262, ensuring that all aspects of the system under development adhere to the standard's specifications. This study also implemented the BOMI model within T-Reqs to improve coordination within the system development process. By defining clear boundary objects, roles, and methodological islands (teams), the BOMI model facilitated a more struc-

tured and efficient approach to managing the complex relationships between different stakeholders and shared artifacts involved in the development process. This dual focus on compliance check automation and coordination addresses key challenges faced by large scale agile system development teams, enabling them to deliver high-quality, compliant systems more efficiently and effectively.

Conclusion : The findings of this study highlight the critical role of automation and improved coordination in addressing the challenges associated with ensuring compliance to safety standards like ISO 26262 in system development. The introduction of automated compliance check capabilities in the TReqs tool, along with the implementation of the BOMI model for enhanced coordination, represents significant advancements in the field of safety-critical software development.

These developments not only reduce the labor-intensive and error-prone aspects of manual compliance verification but also improve the overall efficiency and reliability of the development process. By automating compliance checks and streamlining coordination efforts, the study demonstrates the potential for substantial improvements in the delivery of dependable, high-quality safety-critical software solutions. Looking forward, the next steps in this research direction could involve further refining the automated compliance check algorithms to accommodate more nuanced interpretations of ISO 26262 requirements. Additionally, exploring the integration of these tools and processes with other safety standards could broaden their applicability and impact across different industries and domains. As the demand for safety-critical software continues to grow, the lessons learned from this study will undoubtedly play a crucial role in shaping the future of system development practices.

Keywords: TReqs, BOMI, Requirements Engineering (RE), ISO 26262, Agile,

Acknowledgements

We would first and foremost like to thank our academic supervisor, Eric Knauss, for his guidance and help throughout the project. His time and dedication to see the project come to fruition is greatly appreciated. We would also like to thank Jennifer Horkoff, Filip Lange, Abdullatif Alshriaf, Sicily Brannen, Alessia Knauss, Robert Nilsson, Anders Kvist, Jesper Thyssen for their valuable input to this study. Finally we would like to thank everyone that took time to participate in this study.

Michael Osei Aduamah & Michiale Hadgu Araya, Gothenburg, September 2024

Contents

1	Introduction	1
1.1	Purpose of Study	2
1.2	Problem Statement	2
1.3	Research Questions	3
1.4	Scope	3
1.5	Structure and Contributions	4
1.5.1	Structure	4
1.5.2	Contributions	4
1.5.2.1	Contributions Beneficial for Industry Practitioners: .	4
1.5.2.2	Contributions Beneficial for Researchers:	4
2	Background	7
2.1	Large Scale Agile Development	7
2.2	T-Reqs for Traceability and Compliance in Large scale agile develop- ment	8
2.2.1	The T-Reqs Type and Trace Information Model	8
2.2.2	Advantages of T-Reqs over other tools in Large Scale Agile Development	8
2.2.2.1	Lightweight Tooling:	8
2.2.2.2	Unmatched Flexibility and Adaptability	8
2.2.2.3	Scalability Without Compromise	9
2.2.2.4	Superior Traceability and Linkage	9
2.2.2.5	Cost-Effective Without Sacrificing Quality	9
2.2.2.6	Future-Proof Infrastructure	9
2.3	The BOMI Model	10
2.4	Regulatory Standards Compliance and ISO 26262	11
2.4.1	Benefits of Achieving ISO 26262 Compliance	12
2.4.2	ISO 26262 Part 6	13
2.4.3	Compliance To ISO 26262 Part 6 in T-Reqs enabled projects .	14
3	Related Work	15
3.1	Large Scale Agile, Boundary Objects and Coordination	15
3.2	Traceability And Compliance In Systems Development	16
3.2.1	Traceability Strategies in Industry	17
3.2.1.1	Challenges Faced During the Traceability Process . .	17
3.2.2	Compliance Strategies in Industry	18

3.2.2.1	Challenges Faced During the Compliance Process . . .	19
4	Research Methods	21
4.1	Design Science Research	21
4.1.1	Design Process	21
4.1.1.1	Problem Identification	21
4.1.1.2	Solution Suggestion	22
4.1.1.3	Solution Implementation	22
4.1.1.4	Solution Validation	23
4.1.1.5	Solution Evaluation	23
4.2	Data Collection	23
4.2.1	Participant Selection Criteria	23
4.2.2	Workshops	24
4.2.3	Surveys	25
4.2.4	Interviews	25
4.3	Data Analysis	26
4.4	Design Science Iterations	27
4.4.1	Iteration 1	28
4.4.2	Iteration 2	28
4.4.3	Iteration 3	28
5	Findings	29
5.1	The Artifact : The Improved T-Reqs Tool for automated compliance checking	29
5.1.1	The New TTIM	30
5.1.2	Enhanced Compliance Checking Capabilities	32
5.1.3	Export to PDF Functionality	34
5.2	Iteration I	34
5.2.1	Problem Identification	35
5.2.1.1	Problem 1: Documentation Management Conflict	35
5.2.1.2	Problem 2: Abstract and Evolving Standards	35
5.2.2	Solution Suggestion	36
5.2.3	Solution Implementation	36
5.2.4	Solution Validation	38
5.2.5	Solution Evaluation	40
5.3	Iteration II	40
5.3.1	Problem Investigation	41
5.3.2	Solution Suggestion	41
5.3.3	Solution Implementation	42
5.3.4	Solution Validation	43
5.3.5	Solution Evaluation	44
5.3.5.1	Survey Findings:	45
5.4	Iteration III	45
5.4.1	Problem Identification	46
5.4.2	Solution Suggestion	46
5.4.3	Solution Implementation	46
5.4.4	Solution Validation	47

5.4.5	Solution Evaluation	47
5.4.5.1	Evaluation Tasks Overview	48
6	Discussion	51
6.0.1	RQ1: How can the integration of the BOMI metamodel within T-Reqs enhance coordination and support compliance checking, and what role do boundary objects play in this process? .	51
6.0.2	RQ2: How can T-Reqs be optimized to automate compliance checking for ISO 26262 using BOMI, and what are the benefits of this approach?	51
6.0.3	RQ3: How effective is the use of T-Reqs and BOMI for automating compliance checking for ISO 26262 in system projects, and what are the key factors that contribute to their success or failure?	52
6.1	Threats to Validity	52
7	Conclusions and Future Work	53
7.0.1	Future Work	53
	References	55
A	Appendix 1	I
A.1	A sample TReqs Type and Traceability Information Model	I
A.2	Implementation	II
A.3	Survey for evaluation in Iteration 2	VII
A.4	Survey for evaluation in Iteration 3	XIII
A.5	Interview Guide	XVI

1

Introduction

Delivering safety critical software solutions that are dependable, and of high quality has been shown to depend heavily on ensuring standards compliance in system development initiatives.

Strict documentation and traceability of all work products throughout the design and development lifecycle are necessary to comply with safety requirements such as ISO 26262. The lifecycle's work products support the system's safety claims by acting as direct, instantaneous proof that is necessary for the safety evaluation procedure. To be more precise, ISO26262's V-model's left-side outputs such as requirement specifications offer direct evidence, and the right-side outputs—such as verification results offer immediate evidence [1]. Failure to properly manage documents or proof can unintentionally increase risks to the compliance process [2]. Part 10 of the standard emphasizes that documents should be accurate, succinct, logically structured, easily understood by intended users, and maintainable [3]. The manual process of verifying compliance against the standard's requirements, including the detailed examination of work products within the project scope, introduces substantial challenges such as high labor intensity and a higher risk of errors [4]. Given the complex dynamics of change, particularly in geographically distributed large-scale agile projects, as well as the need for effective documentation and cooperative teamwork, automated traceability between the standards-mandated requirements and the development artifacts generated becomes crucial [5]. With frequent changes and the requirement for rapid adaptation, presents major challenges in identifying and maintaining clear lines of accountability for developing and managing work products that meet these standards. The challenge surrounds who is responsible for specific work deliverables at any one time and what work deliverables necessary for compliance have been fulfilled. This ambiguity might impede the efficient management of documentation and evidence needed for safety assessments, potentially increasing the risk of compliance failure.

Moreover, agile methodology's emphasis on iterative development and continuous improvement can promote these challenges. Teams often operate in overlapping phases of the development life cycle, making it difficult to establish clear boundaries for what constitutes "fulfilled" work products that meet the standard's requirements. This overlap can complicate the tracking and verification of compliance, as work products may evolve or be redefined as new iterations progress.

In this study, we introduce an enhanced traceability tool [6] designed to address the complexities of adhering to safety standards like ISO 26262, particularly in the context of large-scale agile projects. This tool is focused on improving the management of required work products by the standard, aiming to streamline the

process of documenting and tracing compliance with ISO 26262.

Our approach focuses on improving traceability between the requirements dictated by standards, the development artifacts produced, the roles and teams involved in the development process. Despite the limitations of our investigation, our findings are generalizable to other parts of the ISO 26262 standard, making this work a significant contribution towards the provision of an ISO 26262-compliant methodological approach for representing and shaping the crucial work products required for safety case creation.

1.1 Purpose of Study

This study's main objective is to address a significant issue that development teams and process engineers encounter: the time-consuming compliance checking operations. We aim to identify the challenges, propose solutions and evaluate these solutions through workshops with industry experts. We seek to implement an artifact that will cater for the challenges faced during compliance check activities and coordination in system development projects across globally dispersed teams.

This study was grounded in the Design Science Research methodology (DSR) through which we explored the difficulties presented by compliance to ISO 26262. The findings are anticipated to provide beneficial insights to both industry professionals and researchers. By providing this tooling solution for automated compliance checking, practitioners can effectively manage and trace all required work products necessary to achieve compliance. Researchers can also learn about the challenges and proposed solutions in compliance check activities while navigating and collaborating with standards.

1.2 Problem Statement

The stakes are extremely high in the automotive business since even small mistakes can have severe consequences. Modern cars are complicated and interconnected, frequently incorporating sophisticated software and technology. As a result, any deviation from strict safety regulations or poor quality control can have disastrous results. The automotive industry needs flawless performance since mistakes might result in accidents that cause property and human casualties to be lost. This elevated risk environment emphasizes how vital it is to have thorough documentation, strict quality controls, and strong traceability systems in place to guarantee that every facet of vehicle development, manufacture, and design adheres to the strictest safety regulations.

There are several challenges when manually verifying compliance against numerous standards. The labor-intensive nature of this approach is among the most notable. The process of carefully reviewing process entities to make sure that work products exist that satisfy a set of requirements is always challenging because safety standards are so complex in nature. Additionally, the complexity of collaboration across large-scale agile teams introduces new hurdles. Coordinating efforts, sharing knowledge, and ensuring consistency across teams become increasingly difficult as projects grow

in size and scope. This lack of cohesive collaboration can lead to inconsistencies in compliance verification, potentially compromising safety standards.

Another significant challenge is the vulnerability to errors. Manual processes, by their very nature, are prone to human error. Even minor mistakes in documentation or oversight in the traceability of work products can have profound implications for compliance. These errors might cause gaps in the information supporting the system's safety claims, which could risk its regulatory status and end-user safety.

Incorporating automated processes can enhance the efficiency and accuracy of compliance checking, even in geographically dispersed teams. For instance, utilizing software solutions that support traceability between requirements, test cases, test results, code, and code reviews can streamline the process of ensuring that work products adhere to the specified standards.

1.3 Research Questions

Goal: How can the T-Reqs tool effectively facilitate tracing to standards and support automated compliance checking in the context of large-scale agile systems engineering?

RQ1. How can the integration of the BOMI metamodel within T-Reqs enhance coordination and support compliance checking, and what role do boundary objects play in this process?

RQ2. How can T-Reqs be optimized to automate compliance checking for ISO 26262 using BOMI, and what are the benefits of this approach?

RQ3. How effective is the use of T-Reqs and BOMI for automating compliance checking for ISO 26262 in system projects, and what are the key factors that contribute to their success or failure?

1.4 Scope

The study is specifically focused on enhancing the T-Reqs tool, a text-based, lightweight requirements management solution, to better handle the complexities of coordination, compliance and traceability in software development projects [7] [8].

Integration with BOMI Metamodel: Our research aims to create a specialized TReqs Type and Traceability Information Model (TTIM), based on the BOMI metamodel. This model will enable the effortless creation of uml diagrams that depict the coordinational aspects of a system project from traceability data. [9].

Automated Compliance Checking: The study also explores the representation of ISO standards in T-Reqs using Markdown to enable automated compliance checking, aiming to streamline the process of ensuring adherence to critical standards like ISO 26262.

Large-Scale Agile Projects: The research is particularly relevant to large-scale agile projects, where maintaining traceability between changing requirements and related development artifacts is a significant challenge [10] [11].

1.5 Structure and Contributions

1.5.1 Structure

Chapter 2 provides the necessary background information to understand the context and significance of the study. This chapter is crucial for establishing the theoretical framework and the rationale behind the research questions. Chapter 3 of this study delves into the related work that has been conducted in the field. This chapter sets the stage by reviewing previous studies and findings that are relevant to the research questions being addressed. Chapters 4 and 5 are dedicated to discussing the research methods and findings. Chapter 4 outlines the methodological approach taken to investigate the research problem, describing the specific methods of data collection and analysis used. Chapter 5 presents the results obtained from the research, providing insights into the findings and their implications. Chapter 6 engages with the research questions through a discussion, offering a critical analysis of the findings in relation to the research questions. Finally, Chapter 7 concludes the study by addressing potential threats to the validity of the research findings.

1.5.2 Contributions

The contributions of this study aimed at addressing the complex challenges of ensuring compliance with safety standards like ISO 26262 in large-scale agile software development projects will be beneficial to both industry practitioners and researchers.

1.5.2.1 Contributions Beneficial for Industry Practitioners:

Our contribution to industry are centered on improving the T-Reqs tool, with an emphasis on automating its compliance check procedures. This development makes it possible to track necessary work artifacts and determine their completion status quickly and effectively. Furthermore, we used the BOMI metamodel to define roles and teams participating in the development process and to assign accountability for each work output. This strategy improves overall project efficiency, knowledge sharing, team coordination, and compliance while streamlining compliance management and fostering a clearer sense of accountability within agile projects.

1.5.2.2 Contributions Beneficial for Researchers:

Automated Compliance Check Processes: By introducing our method for automating compliance checks, this study provides academics with a useful foundation for researching how document management tools affects compliance management in agile projects. This paper adds to the body of knowledge in academia regarding the use of technology to enhance regulatory compliance.

Role Definition and responsibilities in Agile Teams: Our research offers an organized approach to investigating team dynamics and their impact on project outcomes by applying the BOMI metamodel to define roles and assign responsibilities within agile teams. This methodology presents a fresh perspective on how

well-defined roles and responsibility frameworks can improve team productivity and project outcomes.

Generalizable Insights Across Different Standards and Tools: The insights derived from our study extend beyond the specific context of T-Reqs and ISO 26262, offering a broader understanding of software development and compliance management. This generalizability enables researchers to apply our findings to other standards and tools, broadening the scope of their investigations.

Efficient and Effective Traceability and Compliance Management: Taken as a whole, our contributions seek to transform traceability and compliance management in large-scale agile projects, especially in situations involving critical safety. This significant development not only answers open research problems, but it also establishes a standard for further research on improving software development procedures under strict safety regulations.

2

Background

This chapter delves into the theoretical foundation and background information necessary for understanding the study, focusing on T-Reqs, a text-based requirements management solution designed to support agile teams in managing requirements within large-scale agile system development [6] [7] [8]. The Boundary Objects between Methodological Islands (BOMI) model, a comprehensive framework that addresses the coordination needs and boundary objects of an organization, especially useful for large-scale systems development, is also discussed [9] [12] [13]. Furthermore, the chapter touches upon ISO 26262 Part 6, outlining the requirements for product development at the software level, emphasizing the importance of safety in the development process [14].

2.1 Large Scale Agile Development

Large-scale agile has grown to be a critical approach to system development in the software industry, leveraging agile methodologies to tackle the complexities of developing large-scale applications and systems. Originating from the early 2000s, agile methodologies, as articulated in the "Agile Manifesto", prioritize human interaction, working software, customer collaboration, and adaptability to change. These principles have been foundational in shaping agile development practices, emphasizing the importance of cross-functional teams, iterative development cycles (sprints), and face-to-face communication. Large-scale agile development faces significant challenges in traceability, collaboration. These challenges arise from the flexible and iterative nature of agile methodologies, which can sometimes conflict with the rigid requirements of regulatory standards and the need for thorough documentation and traceability.

Inter-team coordination also becomes more complex as the number of teams involved increases, demanding specific ways to ensure efficient communication and knowledge sharing [15]. This includes utilizing backlogs, planning meetings, and specialist roles such as architects to address problems across team boundaries. Knowledge management, which is critical for coordinating and directing development efforts, distinguishes between tacit, implicit, and explicit knowledge, emphasizing the relevance of artifacts in managing knowledge in agile contexts. Traceability, a significant enabler of knowledge management, guarantees that linkages between artifacts are tracked, allowing for the rapid discovery of important knowledge affected by changes [15] [16].

2.2 T-Reqs for Traceability and Compliance in Large scale agile development

T-Reqs (Textual Requirements) is a text-based requirements management solution designed by researchers from Chalmers University [17] supported by Software Center and Ericsson AB to aid agile teams in managing requirements within large-scale agile system development. It integrates with Git, a widely-used distributed version control system, to provide a powerful, scalable solution for managing requirements alongside code and tests. This integration is particularly beneficial for agile teams, enabling them to keep requirements close to the development process and adapt them dynamically as the project evolves. T-Reqs was developed from research work done on traceability challenges with several industrial partners from Software center with Axis, Ericsson, Grundfos, Saab, Siemens, TetraPak, Volvo Cars, Volvo Trucks and Vinnova FFI (NGEA) [6] [7] [8].

2.2.1 The T-Reqs Type and Trace Information Model

To define and trace types within T-Reqs, an innovative approach is represented by the T-Reqs Type and Trace Information Model (TTIM). The goal of this model is to improve complexity, interoperability, and maintainability in the creation of complex systems by providing a framework for organizing and connecting different requirements engineering components. To demonstrate how types and traces can be efficiently defined and used within T-Reqs, a sample TTIM can be found in "Appendix A.1".

2.2.2 Advantages of T-Reqs over other tools in Large Scale Agile Development

Choosing T-Reqs for our study, especially in the context of traceability and compliance in large-scale agile development, was based on several key advantages that T-Reqs offers over other existing tools. Our decision was primarily influenced by T-Reqs' unique capabilities that align closely with the requirements of managing traceability and compliance within agile environments, particularly those operating at scale.

2.2.2.1 Lightweight Tooling:

T-Reqs offers lightweight tooling that does not require significant adjustments to an organization's existing processes. Instead, it aims to fit into the organization's agile way of working.

2.2.2.2 Unmatched Flexibility and Adaptability

Unlike rigid, proprietary tools that force you into their mold, TReqs offers unparalleled flexibility:

- Its text-based format allows for seamless integration into existing workflows without disruption.
- It works perfectly with standard agile development tools, eliminating the need for costly replacements.
- Its extensible and customizable architecture ensures it adapts to your evolving needs, not the other way around.

2.2.2.3 Scalability Without Compromise

While traditional tools falter under complexity, TReqs thrives:

- Designed specifically for managing requirements in large-scale agile system development.
- T-Reqs scales effortlessly to meet the growing demands of companies and supports multiple teams working simultaneously on requirements without bottlenecks.

2.2.2.4 Superior Traceability and Linkage

TReqs outperforms competitors like Word and Excel in tracing to requirements:

- Provides robust traceability between requirements and system properties
- Facilitates comprehensive understanding of relationships between different levels of requirements

2.2.2.5 Cost-Effective Without Sacrificing Quality

Unlike expensive enterprise solutions like Doors and RequisitePro, TReqs delivers high-quality features at a fraction of the cost:

- Eliminates licensing costs associated with traditional requirements tools.
- Utilizes existing Git infrastructure, reducing overall tooling expenses.
- Open-source nature allows for customization without vendor lock-in.

2.2.2.6 Future-Proof Infrastructure

While others struggle to keep pace, TReqs embraces modern development practices:

- Built on Git-based principles, ensuring compatibility with cutting-edge CI/CD pipelines
- Supports standard technology stacks used by agile teams, future-proofing your investment
- Infrastructure-as-code approach allows for easy deployment and management

In comparison to other tools evaluated, including Doorstop, word processors, spreadsheets, traditional RE tools, issue trackers, and Eclipse Capra, T-Reqs emerged as the superior choice due to its comprehensive feature set tailored for agile development environments. Its integration with Git, support for traceability, version control, and collaboration at scale, together with its advanced features and potential for regulatory compliance support, made T-Reqs an ideal solution for this study [18] [7] [8].

2.3 The BOMI Model

Boundary Objects between Methodological Islands (BOMI) model is a comprehensive framework that provides an understanding of the coordination needs and boundary objects [13] of an organization. This model is especially useful for large-scale systems development, where it can be difficult to manage knowledge across various organizational groups, especially in the rapidly changing context of agile development [9] [13].

According to the BOMI model, a group that employs development techniques distinct from those utilized by the neighboring organization is referred to as a Methodological Island (MI). These MIs work in an environment where other organizational divisions that employ different development techniques are present. They are usually found in larger organizational structures. This calls for the use of coordination mechanisms, which are backed by a range of artifacts such as written documents, models, backlogs, and code referred to as Boundary Objects. [9] [13]. Boundary Objects (BOs), are the key interfaces that connect these Methodological Islands. They act as the links between various development teams and methodologies, facilitating effective knowledge sharing and coordination [13].

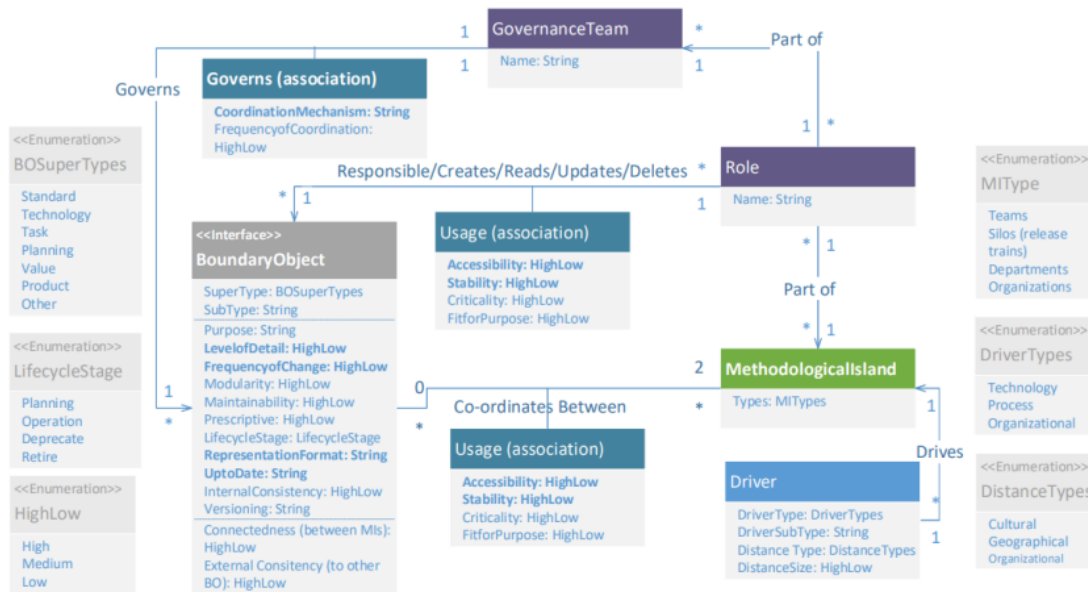


Figure 2.1: The BOMI metamodel [9]

Given that our goal of this thesis also aimed at improving coordination within large-scale agile projects, the BOMI model emerged as an optimal choice. It offers a clear understanding of how boundary objects facilitate communication and collaboration across different methodological islands, which is essential in agile environments characterized by rapid iterations and dynamic team interactions.

This choice was driven by several key considerations:

Unique Attributes of the BOMI Model: Our research did not identify any other model that offered the same level of detail and structure around shared artifacts

across dispersed teams(boundary objects), roles, and project components as the BOMI model. This uniqueness was a significant factor in our decision to adopt the BOMI model.

Enhanced Collaboration: By clearly defining and managing boundary objects, teams can better understand and communicate the roles and responsibilities of different stakeholders, leading to improved collaboration and coordination.

Increased Transparency: Clear documentation and management of boundary objects increase transparency within teams and across projects. Stakeholders can easily see where decisions are made and who is responsible for different aspects of the project.

Better Decision Making: By providing a structured approach to managing boundary objects, teams can make more informed decisions. Understanding the implications of changes to boundary objects can lead to more strategic decision-making.

Promotes Knowledge Sharing: By centralizing the management of boundary objects, teams can promote knowledge sharing. This can lead to a more cohesive understanding of the project and its goals among all participants.

Based on these foreseeable benefits in integrating BOMI in TREqs enabled projects, we decided to adopt the model for this study. The BOMI model can significantly improve coordination in T-Reqs enabled projects by providing a structured framework for managing boundary objects(work artifacts) shared between different methodological islands. This will ensure that all teams are aligned, that tacit information flows effectively between different parts of the project, and that the project can adapt and evolve as needed [9]. The attributes of the main parts of the BOMI model will also serve as a blueprint for managing artifacts, roles, and teams involved in the compliance checking functionality of our proposed tooling solution.

2.4 Regulatory Standards Compliance and ISO 26262

Regulatory standards compliance is a critical aspect of modern industries, ensuring products meet safety, performance, and environmental criteria [19]. Among these standards, ISO 26262 stands out as a globally recognized standard specifically tailored for functional safety in automotive electronic/electrical systems. Published by the International Organization for Standardization (ISO), ISO 26262 aims to reduce risks posed by hazards caused by malfunctioning behavior of electrical and/or electronic systems. The standard outlines a process for identifying, assessing, and mitigating risks associated with these systems throughout their life cycle, from concept to production and service. It covers activities such as hazard analysis, risk assessment, hardware and software fault modeling, failure mode and effects analysis (FMEA), and robustness analysis.

The standard ISO 26262 is composed of twelve parts, including ten normative parts and two guidelines, designed to guide the functional safety of automotive electronic/electrical systems throughout their lifecycle, from concept to decommissioning, ensuring the highest standards of safety and reliability.

Part 1: Vocabulary This part of the ISO defines terms used in the standard to ensure everyone understands what is meant by "functional safety."

Part 2: Management of Functional Safety This part explains how to organize and manage functional safety efforts within an organization.

Part 3: Concept Phase This part of the ISO guides the early stages of product development, where safety goals and requirements are established.

Part 4: Product Development at the System Level This part helps define safety features at the system level, ensuring the whole system works safely together.

Part 5: Product Development at the Hardware Level This part of the ISO focuses on designing and evaluating hardware to prevent failures that could harm people.

Part 6: Product Development at the Software Level This part of the ISO outlines how to design and verify software to minimize errors that could lead to accidents.

Part 7: Production and Operation This part of the ISO ensures that the manufacturing and use of the product don't introduce new safety risks.

Part 8: Supporting Processes This part of the ISO details the processes needed to keep track of changes and ensure safety remains intact.

Part 9: Automotive Safety Integrity Level (ASIL)-oriented and Safety-oriented Analysis This part of the ISO shows how to analyze safety risks and decide on the right safety measures.

Part 10: Guidelines on ISO 26262 This part of the ISO gives extra information to help understand and apply the standard better.

Part 11: Guidelines on Applying the Standard to Semiconductors This part of the ISO advises semiconductor makers on how to follow the standard in their work.

Part 12: Adaptation of ISO 26262 to Motorcycles This part of the ISO explains how the standard can be adapted for motorcycles, ensuring they're safe too.

Compliance with ISO 26262 is mandatory for automotive manufacturers and suppliers in many jurisdictions, reflecting the standard's recognition as a benchmark for functional safety in the automotive sector. Adherence to the standard helps organizations avoid legal penalties, enhance consumer trust, and ensure the reliability and safety of their products.

This standard plays a pivotal role in enhancing the safety of automotive systems by providing a comprehensive framework for managing functional safety risks. Its adoption shows the industry's commitment to innovation and safety, ensuring that advancements in technology do not compromise the fundamental principle of protecting human life. [20] [21] [3].

2.4.1 Benefits of Achieving ISO 26262 Compliance

Improved Safety: Adopting ISO 26262 helps ensure that the safety of car components is considered from the beginning of the development process. This can improve the safety of car electronic systems and show customers, regulators, and end users the company's commitment to safety [20] [19].

Reliable Product Production: The standard outlines organizational structures and management practices necessary for compliance with functional safety, ensuring that organizations have the necessary management structure to reliably produce safety-based products. This includes evidence of competence, quality management, and measures for assessing functional safety [20] [19].

Regulatory Compliance: While ISO 26262 is not required by law, many car makers and suppliers follow it to show their commitment to safety. Sometimes customers and regulators might require them to prove they follow the standard. Even if it's not required, following it can improve the safety of car electronic systems [20].

2.4.2 ISO 26262 Part 6

Part 6 of the ISO 26262 standard, emphasizes the importance of extensive traceable documentation as proof of compliance throughout the software development lifecycle [14]. This documentation, referred to as "work products" in the standard's annexes and at the end of subclauses, is crucial for demonstrating that all safety-related aspects of the software have been adequately addressed and verified. Work products encompass a wide range of documents and records generated during the software development process, including but not limited to:

Requirements Specifications: Detailed descriptions of the software's functional and non-functional requirements, including safety requirements.

Design Documents: Documentation of the software architecture, design choices, and interfaces with other systems.

Implementation Records: Evidence of the actual code written, including comments and annotations that relate to safety considerations.

Test Plans and Results: Comprehensive plans for testing the software against its requirements and the results of those tests, including any safety-specific tests.

Configuration Management Records: Information on the versions of the software and its dependencies, ensuring that the correct versions are being used and that changes are properly tracked.

Safety Analysis Reports: Summaries of the analysis conducted to identify and mitigate safety risks, including fault tree analysis, failure mode and effects analysis (FMEA), and others.

These work products serve as tangible evidence that the software development process has been carried out in accordance with the principles of ISO 26262, focusing on minimizing risks associated with product design and development to prevent hazards and potential human health and life-threatening failures. The requirement for extensive traceable documentation shows the standard's commitment to transparency, accountability, and the continuous improvement of safety practices in the development of safety-critical software [20] [14] [21].

2.4.3 Compliance To ISO 26262 Part 6 in T-Reqs enabled projects

We have chosen to focus specifically on ISO 26262 Part 6 for this thesis/study due to its direct relevance to the development of safety-critical systems at the software level [14]. Part 6 provides comprehensive guidance on the entire software development life cycle, from initial concept through to decommissioning, covering requirements specification, design, implementation, integration, verification, validation, and configuration management [14]. The standard requires extensive traceable documentation called "work products" as proof of compliance. These work products serve as a record of the development process, ensuring that all decisions regarding software design and functionality are well-documented and traceable. This is crucial for demonstrating compliance, conducting audits, and facilitating continuous improvement. Tools like T-Reqs significantly enhance the ability to manage and trace these requirements effectively within agile methodologies. By focusing on ISO 26262 Part 6, this study aims to explore how T-Reqs can be effectively integrated into the software development process of safety critical systems to ensure compliance with safety requirements particularly in large-scale projects [22] [23]. It is important to note that the findings and insights gained from this study on integrating T-Reqs into the software development process for ISO 26262 Part 6 compliance are likely to be applicable to other standards that similarly emphasize the importance of traceability in work artifacts as evidence of compliance. This suggests that the benefits of using T-Reqs could extend beyond the specific context of ISO 26262 Part 6, potentially offering valuable lessons for organizations working with various safety-critical standards across different industries.

Table A.1 — Overview of product development at the software level

Clause	Objectives	Prerequisites	Work products
5 General topics for the product development at the software level	The objectives of this Clause are: a) to ensure a suitable and consistent software development process; and b) to ensure a suitable software development environment.	(none)	5.5.1 Documentation of the software development environment
6 Specification of software safety requirements	The objectives of this sub-phase are: a) to specify or refine the software safety requirements which are derived from the technical safety concept and the system architectural design specification;	Technical safety requirements specification (see ISO 26262-4:2018, 6.5.1) Technical safety concept (see ISO 26262-4:2018, 6.5.2) System architectural design specification (see ISO 26262-4:2018, 6.5.3)	6.5.1 Software safety requirements specification 6.5.2 Hardware-software interface (HSI) specification (refined) 6.5.3 Software verification report

Figure 2.2: Annex A of ISO 26262 Part 6 showing a breakdown of required artifacts necessary for compliance [14]

3

Related Work

3.1 Large Scale Agile, Boundary Objects and Coordination

As agile practices and teams scale up, the challenge of coordinating between teams during development phases becomes increasingly complex. This section delves into the intricacies of Agile coordination at scale, drawing insights from recent studies that highlight the pivotal role of boundary objects in facilitating seamless inter-team collaboration. Studies on the lack of efficient mechanisms for inter team coordination affecting the overall efficiency and success of large-scale agile projects, reveal key areas of concern [24] [16] [15]. Recent studies suggest traceable boundary objects offer as a promising avenue for addressing these coordination challenges, by providing a tangible means to link parts of the project and ensure that all stakeholders are aligned with the project's objectives and compliance status [15] [13] [16].

From Wohlrab's study [15] the concept of using traceable boundary objects to enhance coordination among large scale agile development teams, was the main theme. Her work explores how boundary objects, which are entities that are shared among groups and carry meaning across social worlds, can be made "living" or dynamic to support agile methodologies' iterative nature.

Boundary objects are particularly valuable in agile environments because they facilitate communication and understanding between teams that may be working on different aspects of a project but need to coordinate their efforts effectively. However, traditional boundary objects can become static and less effective as projects scale and evolve rapidly. Wohlrab proposes the idea of "living boundary objects" that are adaptable and can evolve over time alongside the project's needs, thus providing a flexible yet reliable means of coordination [15].

Traceability in these living boundary objects is crucial for several reasons:

Visibility Across Teams: Traceability allows all team members to see the current state of the project, the status of tasks, and how different parts of the project relate to each other. This visibility is essential for teams to understand the broader context of their work and how it fits into the larger project goals.

Efficient Communication: By being able to trace the origin, history, and evolution of a boundary object, teams can communicate more effectively. This reduces the likelihood of miscommunication and misunderstandings, which are common pitfalls in agile development.

Facilitating Collaboration: Traceable boundary objects enable teams to collaborate more seamlessly. They provide a common ground for discussion and decision-

making, ensuring that all stakeholders have access to the same information and can make informed decisions based on up-to-date data.

Improving Agility: The ability to trace changes and adaptations in boundary objects enhances agility. As projects evolve, teams can adjust their plans and priorities more dynamically, knowing that their coordination tools are flexible enough to accommodate these changes.

3.2 Traceability And Compliance In Systems Development

Traceability and compliance are pivotal in systems development, especially in sectors prioritizing safety, quality, and regulatory adherence. The literature review by Castellanos et al. emphasizes the critical role of traceability in enhancing software maintenance quality and efficiency, highlighting the importance of automated traceability for real-time, comprehensive information on any requirement [25]. This aligns with the findings that traceability can significantly improve the quality of software maintenance while saving time and effort, as evidenced by studies showing developers completing tasks 24% faster and producing 50% more correct solutions when using enhanced traceability [25].

Further insights come from a conceptual study by Ghobadi, which explores the challenges faced by cross-functional software development teams [24]. Ghobadi's work sheds light on the complexities involved in coordinating between different teams, emphasizing the need for effective traceability mechanisms to facilitate smoother inter-team collaboration and decision-making processes [24]. In our work, we leverage the coordinational aspects of BOMI and the traceability of T-Reqs to facilitate collaborative traceability across boundary objects, roles, teams and drivers.

A multiple case study by Wohlrab et al. investigates collaborative traceability management from the perspectives of organization, process, and culture. Their findings highlight the necessity of a comprehensive approach to traceability, considering both technological and cultural factors to ensure successful implementation and utilization of traceability systems across different organizational contexts [16].

Building upon these insights, TReqs and BOMI can significantly contribute to collaborative traceability management in software and systems engineering. By incorporating TReqs into the development workflow, developers and stakeholders participating in the development phase can more effectively evaluate compliance risks by establishing and maintaining trace links between work products and the standard's requirements [2]. This integration facilitates a deeper understanding of how each component of the system aligns with the standard, enabling proactive identification and mitigation of potential compliance issues. Additionally, BOMI can support the management of shared boundary objects that facilitate communication and knowledge sharing among distributed teams. Both TReqs and BOMI can leverage the findings from the study to enhance traceability management, making it easier for practitioners to navigate the complexities of distributed development environments and to harness the full potential of traceability in improving collaboration and compliance.

3.2.1 Traceability Strategies in Industry

Requirements-Centered Approach: This approach focuses on tracing requirements throughout the development lifecycle, ensuring that all work products are aligned with the defined requirements. It emphasizes the importance of having a clear and unambiguous specification of requirements and their relationships with the developed artifacts [16]. Our approach to using TReqs and BOMI aligns closely with this approach by ensuring trace links between work products and standard's requirements throughout the development process. This method enhances the clarity and unambiguity of fulfilled standards, thus aiding in the evaluation of compliance risks.

Developer-Driven Approach: In this approach, developers are tasked with creating and maintaining links between their work and relevant requirements or standards, presenting significant challenges. Manual checks are prone to human error, can be time-consuming, and may not capture the full scope of traceability needs, especially in complex projects with numerous work artifacts and evolving requirements. Furthermore, relying solely on individual developers' discipline and attention to detail can be risky, potentially leading to inconsistencies in traceability coverage and completeness. This can make it difficult to accurately assess compliance and identify areas of concern.

Mixed Approach: This approach combines elements of both the requirements-centered and developer-driven methods, aiming to balance the need for rigorous traceability with the practicalities of software development. It may involve a mix of automated tools and manual processes to ensure that traceability is maintained across the project lifecycle.

3.2.1.1 Challenges Faced During the Traceability Process

Several studies have explored the importance of establishing traceability between development artifacts and how it aids in complying with ISO 26262, particularly in the automotive industry [26] [27] [28] [29] [16] .

Based on the study by Maro, several key challenges emerge in the realm of traceability management, which, if not addressed, can hinder the effective implementation and utilization of traceability in software and systems engineering projects [29]. These challenges serve as critical problem statements that highlight the need for innovative solutions and methodologies to enhance traceability practices:

Purposeful Traceability: Ensuring that traceability is established for a reason, meaning it is aligned with the needs of various stakeholders both internally and externally. Without a clear purpose, traceability efforts may lack focus and fail to deliver meaningful value.

Cost-effectiveness: The absence of methods for measuring the return on investment (ROI) for establishing traceability makes it difficult for development organizations to determine which traceability practices and tools are most cost-effective and beneficial under specific circumstances.

Value Recognition: For traceability to be effectively utilized within a development organization, all stakeholders involved in its planning, creation, maintenance, and use must recognize its value. If traceability is seen as optional or of low priority, it

is likely to be poorly established and utilized.

Portability: In the context of embedded systems development, where systems are developed across multiple departments and sometimes across different organizations, traceability must be portable and efficiently exchangeable between these entities to ensure seamless collaboration.

Trustworthiness: Users need to trust the traceability links to utilize them for activities such as impact analysis or change management. Ensuring the correctness and completeness of these links is crucial for building this trust.

Configurability: Different development organizations and projects may have varying needs for traceability. Traceability tools must be configurable to accommodate these differing needs, allowing for flexibility in how traceability is implemented.

Scalability: As projects grow larger, so does the network of traceability links. Traceability solutions must be scalable to handle increasing complexity and volume without compromising performance or usability.

Ubiquity: Achieving seamless traceability that is automatically established as work progresses is considered the ultimate goal. However, realizing this "grand traceability challenge" requires overcoming significant hurdles, including the development of advanced automation tools capable of establishing traceability links in the background.

These challenges reveal the complexity of implementing effective traceability management in software and systems engineering projects. Addressing these issues requires a sophisticated approach that considers the specific needs and constraints of each project, leveraging technology and best practices to enhance traceability practices and outcomes.

3.2.2 Compliance Strategies in Industry

Compliance Checking from Standards Concepts Modeling: This approach involves modeling the process elements required by specific standards directly. Studies in this category focus on creating a knowledge base of process concepts defined within the standard itself, often using models of standards (MoS) and user-defined processes (UdP). Compliance checks are performed by comparing these models against the standard's requirements during both process design and runtime. This method emphasizes fidelity to the standard's concepts, aiming to ensure that the software process closely matches the standard's specifications.

Compliance Checking from Process Modeling Languages: Here, process modeling languages are utilized to create process elements and their interactions. This approach leverages existing process modeling languages like SPEM 2.0 and transforms them into ontologies or other formal representations to apply constraints derived from various standards. Compliance checking is facilitated through the formalization of processes in these languages, allowing for a structured and standardized way to assess compliance with specific standards.

Compliance Checking from Documents Workflow: The study explored document-centered approaches as a promising model for assessing software process-related normative compliance. Two notable studies in this category focus on checking document compliance to quality constraints during the development process, utilizing different

states of documents (e.g., planned, in use, submitted, ready for QA, accepted, and present, missing) to derive compliant activities. Another method employs policies that trigger appropriate checks upon document events (open, close, update), acting in different modes (error, warning, guideline) depending on the check's mandatory nature

Compliance Checking from Role-Based Access Controls: Role-based approaches assess compliance by evaluating the permissions and access rights granted to different roles within the software development process. These methods often utilize formal logic to define and enforce access controls based on the roles involved in accessing certain information or performing specific tasks. Compliance is ensured by verifying that access rights are correctly assigned according to the standard's requirements, thereby safeguarding the integrity and confidentiality of the software process.

Each of these strategies offers unique advantages and considerations, tailored to different aspects of the software development lifecycle and varying degrees of standardization and formalization. The choice of strategy depends on the specific needs of the project, the complexity of the software being developed, and the standards to which compliance is required.

For projects requiring compliance with standards like ISO 26262, especially in the automotive industry, a specific Traceability Matrix meant for compliance, known as a compliance matrix, can be used. This matrix tracks requirements from the specified regulations to make it easier to comprehend development and testing. However, creating and maintaining a compliance matrix requires manual effort and does not come with built-in compliance checkers [16] [30].

3.2.2.1 Challenges Faced During the Compliance Process

The literature review by Ardila et al. identifies several key challenges in establishing and maintaining traceability and compliance in software processes [25], including the complexity of traceability across multiple domains, the difficulty of integrating traceability tools with existing systems, the challenge of automating traceability processes, and the scalability of traceability systems. These challenges resonate with the experiences shared by Maro et al., who identified 13 traceability challenges in a large automotive supplier, highlighting the universal applicability of these issues [29].

Complexity of Standards: ISO 26262 is a comprehensive standard with intricate requirements that can be difficult to interpret and implement consistently across different projects and teams ISO 26262.

Integration with Existing Processes: Integrating compliance checks into existing development workflows without disrupting productivity is a significant challenge. Automated tools can help streamline this process but require careful setup and maintenance Automated Compliance Checking.

Cross-Functional Collaboration: Ensuring that all stakeholders, including developers, testers, and managers, are aligned on compliance goals and methodologies can be challenging, especially in large or distributed teams Cross-Functional Software Development Teams.

Resource Allocation: Allocating sufficient resources for compliance activities,

including training and tool procurement, is often a struggle, particularly in fast-paced development environments Resource Management in Software Projects.

Traceability Maintenance: Maintaining up-to-date traceability matrices and documentation is crucial for compliance but can be labor-intensive and prone to errors Traceability in Software Development.

Continuous Compliance Verification: Ensuring ongoing compliance as the software evolves requires robust processes and tools capable of detecting deviations from standards in real-time Continuous Compliance in Agile Development.

Our implementation addresses these challenges by providing an enhanced traceability platform that integrates seamlessly with existing development tools and workflows. It automates compliance checks, reduces the burden of manual traceability maintenance, and facilitates cross-functional collaboration. By leveraging advanced traceability reports our solution can predict compliance risks early in the development cycle, allowing teams to proactively address issues before they become critical.

4

Research Methods

4.1 Design Science Research

Software and systems engineering require collaboration among professionals from various backgrounds during the innovative process of creating complex products. The research methodology adopted in this thesis is rooted in design science principles. We adopted Knauss' guidelines for the Design Science Research (DSR) cycle to structure our study [31]. Our approach began by identifying and addressing real-world challenges and needs within the industry. Following this, we gathered insights into current practices and developed potential solutions. These solutions were then assessed collaboratively with industry partners and academic supervisor and refined further. The design science research activities used in conducting this study involved problem identification, solution suggestion, development(implementation), solution validation and evaluation. The design science research methodology was selected as an appropriate methodology because its objective is to develop new technology-based solutions to important and relevant industry problems or to improve existing artifacts. This makes it inline with the objective of this study. Because the main goal of this study is to improve the basic version of the T-reqs tool in such a way that it improves coordination and compliance checking large-scale agile requirements engineering challenges. This study has been conducted in three different iterations. Each iteration involves the application of all design science research activities. The following section provides a brief description of the design science activities and a summary of what is done during those activities.

4.1.1 Design Process

4.1.1.1 Problem Identification

In our first cycle , the study concentrated on finding current limitations in the traceability and compliance support of the T-Req tool (RQ1), ISO 26262 compliance and investigated the current metamodel utilized to Model Boundary Objects between Methodological Islands (BOMI) (RQ1). This included a thorough analysis of related literature on T-Req, BOMI, compliance, traceability and coordination. We also delved into current traceability and compliance methodologies currently being used in industrial settings and thier challenges. Our second cycle presented new literature and areas to look at from discussions and feedback obtained from the evaluation phase of our first cycle with industry experts. Our 3rd and final cycle identified problems from an interview held with an industry expert and feedback

from the previous cycle.

4.1.1.2 Solution Suggestion

Following the problem identification stage, the plan of action was to develop a T-Reqs type-and-traceability-information-model (TTIM) based on the current BOMI metamodel (RQ1) during cycle 1. By developing a data model that details the parts of the metamodel and their linkages, this strategy sought to improve the coordination aspects of T-Reqs enabled projects. Additionally, the plan included establishing standardized procedures for describing standards in markdown/T-Reqs format (RQ2). This step was crucial for ensuring that standards are clearly defined and easily accessible, facilitating a more efficient and effective traceability process within T-Reqs. Systematic procedures were developed for tracing information from standards to development artifacts (RQ2) in cycle 2 and 3. This involves creating a traceability report that visually maps the connections between standards' requirements, development artifacts and related roles making it easier to identify and manage traceability.

4.1.1.3 Solution Implementation

In the solution implementation stage, the focus was on putting the proposed solutions into action to enhance traceability and compliance support within T-Reqs enabled projects. The following are the phases of our implementation,

Development of the Type-and-Traceability Information Model (TTIM)- (RQ1): Based on the current BOMI metamodel, a TTIM was developed during Cycle 1. This model served as a foundational structure, detailing the components of the metamodel and their interconnections. The TTIM was specifically designed to improve the coordination aspects of T-Reqs-enabled projects by providing a clear and structured representation of the relationships between different elements of the project.

Sample Implementation of ISO 26262 Part 2(RQ3): Leveraging the TTIM and the newly established procedures, a sample implementation of ISO 26262 Part 2 was created using an already modeled version of that standard in Cycle 1. This practical application demonstrated the applicability and effectiveness of the new TTIM . The sample implementation served as a tangible example of how the proposed solutions could be applied in real-world scenarios, offering insights into their potential impact on improving traceability, coordination and compliance support.

Standardized Procedures for Describing Standards (RQ2): To ensure that standards are clearly defined and easily accessible, we utilized markdown for standards representation in T-Reqs. We defined the ISO 26262 Part 6 in markdown and tagged all the required work products for each subclause in Cycle 2. This step was crucial for streamlining the process of tracing the standard easily to all stakeholders and work artifacts required for compliance.

Systematic Procedures for Tracing Information and Reporting Risks(RQ2): Building on the previous steps, systematic procedures were developed for tracing information from standards to development artifacts using the new TTIM in Cycle 2 and 3. This involved creating a traceability report that visually maps the connections between standards' requirements, development artifacts, and related roles. Such mapping

simplifies the identification and management of traceability, making it easier for project participants to understand how each part of the project aligns with the standards, which ultimately enhances coordination and compliance.

4.1.1.4 Solution Validation

Making sure the proposed solutions successfully address the issues found in the traceability and compliance support of the T-Reqs tool is the main goal of the solution validation step. After each implementation phase, the proposed solutions were shown to our supervisor and industry participants from the workshop for feedback and checked for feasibility.

4.1.1.5 Solution Evaluation

During this phase, the solutions were tested in real-world scenarios to measure their impact on T-Reqs enabled projects' coordination, traceability and compliance processes. This involves collecting data on the effectiveness of the solutions in improving traceability and compliance support in the T-Reqs tool and analyzing this data to determine the extent to which the solutions meet the needs of the project and its stakeholders. Our study needed continuous feedback throughout each iterative cycle to refine our exploration of the research questions and assess our proposed solutions. The solutions were presented to industry experts for feedback during 2 workshops held in Cycle 1 and 2. Our participant pool was drawn from a workshop named "Managing Knowledge Flows for Large Scale Agile System Development." This included academic researchers from the University of Gothenburg and Chalmers University with industry experts from various companies.

Company 1: Specializes in automotive software development.

Company 2: Specializes in automotive manufacturing.

Company 3: Specializes in telecommunications.

Company 4: Specializes in manufacturing.

It was beneficial to have diverse views from different fields of work

4.2 Data Collection

During the data collection phase, surveys, interviews and workshop discussions were employed to gather insights and information relevant to the problem investigation and evaluation stages. These methods were chosen for their ability to provide rich data that can offer deep insights into the complexities of the study.

4.2.1 Participant Selection Criteria

A primary consideration in participant selection was the depth of knowledge and experience in T-Reqs and compliance to standards. Participants with direct involvement in software development, such as systems architects and engineers, were prioritized for their extensive experience and deep understanding of the issues faced

when complying to standards. Also, researchers who were actively involved in developing various aspects of T-Reqs and the BOMI model, provided invaluable insight. Their practical knowledge and thorough understanding of software development processes and compliance standards helped us navigate some practical challenges and opportunities associated with implementing and managing T-Reqs and standards compliance in real-world scenarios.

The table below lists the participants involved in the data collection workshops and interviews, detailing their roles, years of experience, and participation in workshops and interviews.

Participant	Role	Experience (years)	Workshops Present	Interviews
Participant 1	Systems Architect	10+	1& 2	
Participant 2	Systems Engineer	14+	1& 2	YES
Participant 3	Systems Engineer	10+	1& 2	
Participant 4	Researcher	5+	1& 2	
Participant 5	Researcher	5+	1& 2	
Participant 6	Researcher	5+	1& 2	
Participant 7	Research Assistant / Masters Student	5+	1& 2	
Participant 8	Masters Student	0	1& 2	
Participant 9	Masters Student	0	1& 2	
Participant 10	Masters Student	0	1& 2	
Participant 11	Masters Student	0	1& 2	
Participant 12	Bachelor's Student	0	1& 2	

Table 4.1: List of participants involved in data collection workshops and interview

4.2.2 Workshops

Workshops are particularly effective in the early stages of problem investigation to generate ideas and solutions, fostering a collaborative environment where participants can contribute their expertise [32]. We engaged experts in two workshops, corresponding to cycles 1 and 2 of our research, to collect their feedback on the identified problems and proposed solutions. Our presentations and discussion during each workshop lasted approximately 45 minutes and the rest of the 3 hour long session was focused on topics revolving around requirements engineering, making the discussions highly relevant to our study.

To facilitate participation and accommodate attendees' preferences, we adopted a hybrid format for the workshops, combining physical attendance with virtual participation via Microsoft Teams. This approach allowed us to reach a broader audience, including those who might not have been able to attend in person.

At the beginning of each workshop, we presented the problems we had identified along with our proposed solutions. Following this presentation, we opened the floor for a discussion, encouraging participants to share their thoughts, experiences, and suggestions. This interactive session fostered a collaborative environment where participants could contribute their expertise and insights.

Participants willingly gave their consent for us to record the proceedings and use the collected data for our study. The recordings served as a valuable resource, allowing us to revisit the discussions and extract detailed insights that might have been overlooked during the live sessions. The workshops proved to be an effective means of engaging with experts in the field, providing us with rich, qualitative data

that significantly contributed to the depth and breadth of our research findings.

4.2.3 Surveys

During Cycle 2 and 3, surveys were distributed to workshop participants and other selected experts knowledgeable about compliance and safety standards. This survey aimed to gather insights pertinent to Research Questions 1 and 2 (RQ1 and RQ2), specifically exploring the representation of BOMI in T-Reqs, the efficiency of tracing to boundary objects for automated compliance checking, and the evaluation of the compliance report generated.

The surveys utilized Likert scales, a psychometric scale commonly employed in questionnaires to measure attitudes or opinions. A Likert scale typically presents respondents with statements and asks them to rate their agreement or disagreement on a symmetric agree-disagree scale. In this case, the survey included both 3-point and 5-point Likert scale questions, providing respondents with options ranging from strong disagreement to strong agreement, with neutral options available. This scale allows for quantifiable analysis of subjective responses, facilitating statistical analysis of the collected data.

Additionally, the surveys incorporated open-ended questions to capture qualitative feedback. This combination of quantitative Likert scale questions and qualitative open-ended questions enabled a comprehensive analysis of the participants' perceptions and experiences, offering both numerical data for statistical analysis and rich textual data for thematic analysis. This approach ensured a thorough investigation into the effectiveness of BOMI representation in T-Reqs, the utility of tracing for compliance checking, and the perceived value of the compliance reports, addressing the core objectives of RQ1 and RQ2.

4.2.4 Interviews

Interviews, especially semi-structured ones, allow for a detailed exploration of individual perspectives and experiences, providing a direct line of communication between the researcher and the participants. This method is particularly useful for understanding the nuances of the problems and the potential solutions from the participants' viewpoints [33].

In the second cycle of the study, a semi-structured interview was conducted to delve deeper into the insights gathered from the survey results and previous workshop. This interview served as a complementary method to further enrich the data collected, aiming to uncover any overlooked details or to explore new avenues that were not previously considered. The choice of a semi-structured interview format allowed for flexibility in the discussion, enabling us to probe deeper into the participant's experiences and opinions while still maintaining a structured approach to ensure coverage of key areas of interest.

The interview lasted approximately an hour, during which the participant provided consent for recording. This consent facilitated the capture of detailed responses, which could be reviewed and analyzed later for patterns, themes, and insights that might not have emerged from the survey or workshop. This approach not only

supplemented the quantitative data obtained from surveys and workshops but also contributed to a more holistic view of the issue being studied, enriching the overall research output.

The data collected will be transcribed and coded for analysis, providing a solid foundation for the subsequent stages of the research process.

4.3 Data Analysis

The data analysis section of this study encompasses a comprehensive examination of the qualitative and quantitative data collected through interviews, workshops, and surveys.

Thematic Analysis of Interviews Thematic analysis was applied to the transcriptions of semi-structured interview conducted in the second cycle of the study. This method allowed for the identification of recurring themes and patterns within the participant's responses, providing a nuanced understanding of their perceptions and experiences regarding the challenges and potential solutions in traceability and compliance management. The interpretation of these themes was guided by the research questions and objectives, ensuring that the analysis remained aligned with the study's goals.

Workshop Observations and Survey Responses Observations made during the workshops were summarized and interpreted to gauge the participants' reactions to the proposed solutions and to identify areas of consensus or divergence among the experts. This qualitative data was complemented by the analysis of survey responses from 5 respondees, which included both open-ended and closed questions. Open-ended responses were interpreted qualitatively to capture detailed feedback and suggestions from the participants. Closed questions, formatted as Likert scales, were analyzed quantitatively to assess the overall sentiment towards the functionality and usability of the T-Reqs tool towards compliance checking. The responses were converted into numerical values and visualized using bar charts, facilitating a comparison of satisfaction levels across different aspects of the tool.

Usability Assessment Using the System Usability Scale (SUS) The System Usability Scale (SUS) was employed to evaluate the usability of the T-Reqs tool. The SUS scores were calculated by converting the Likert-scale responses from five respondees into numerical values, summing these values for each respondent, and averaging the results across all respondents. The average SUS score was then interpreted according to established benchmarks to determine the tool's usability rating. This quantitative analysis provided a standardized measure of usability, offering insights into the tool's user-friendliness and areas for improvement.

4.4 Design Science Iterations

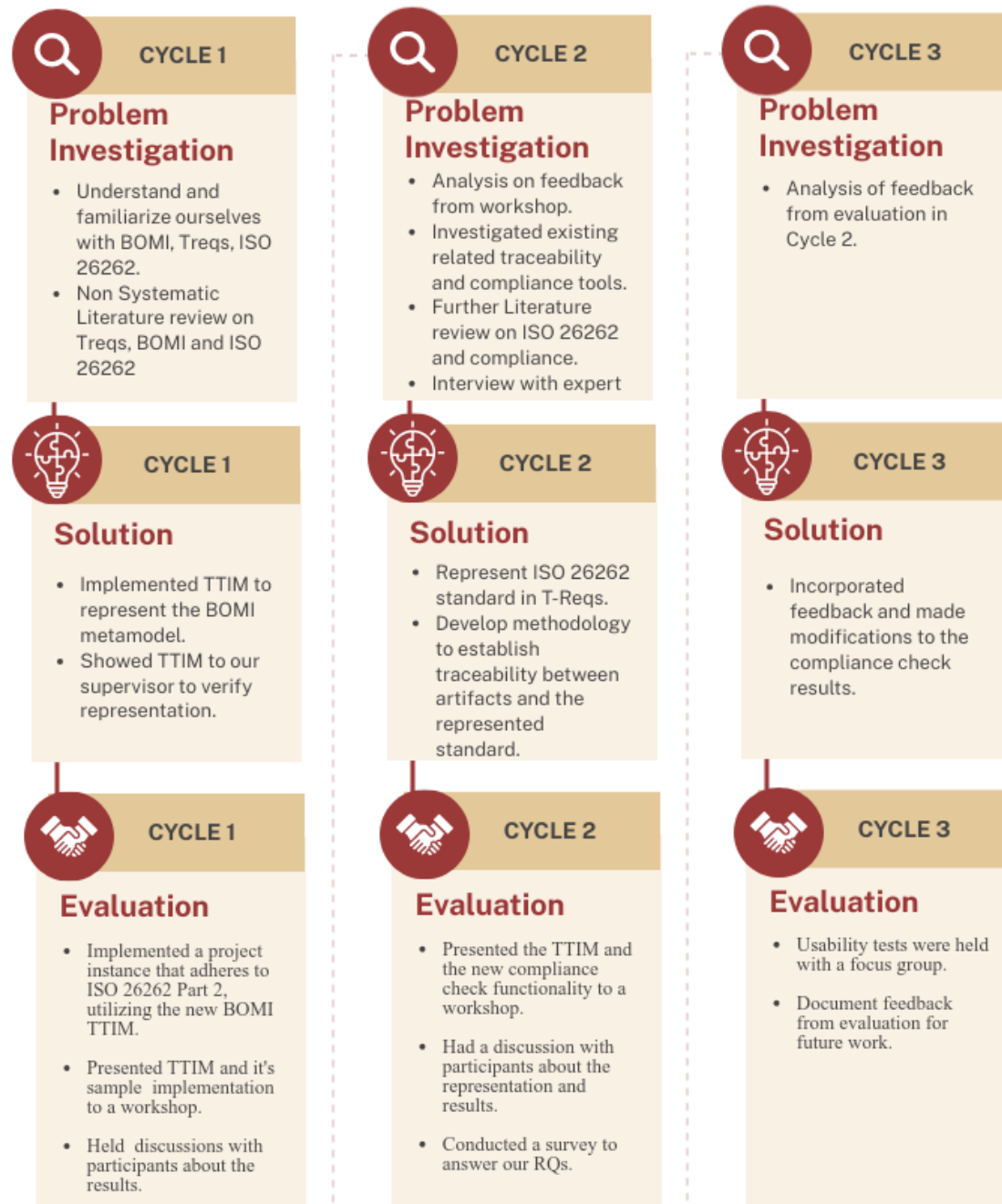


Figure 4.1: An overview of design science cycles used for this study

Figure 4.2 below gives an overview of how T-Reqs interacts with BOMI's concepts and aligns itself with ISO 26262 requirements across the three iterations of our design science research.

4.4.1 Iteration 1

Iteration 1 focused on gaining a deep understanding of the concepts surrounding BOMI, TReqs, and ISO 26262. Through related literature review and discussions with industry experts in a workshop, we familiarized ourselves with these areas. We identified a number of significant problems throughout this iteration that keep hindering large-scale projects' ability to handle requirements and compliance effectively. We then developed the new Treqs Type and Traceability Information Model(TTIM) based on the BOMI model and implemented an instance of an already made BOMI model representing ISO 26262 Part 2. The results of this implementation was presented and discussed in a workshop setting, providing us with initial insights into the practical application and reception of our TTIM.

4.4.2 Iteration 2

Iteration 2 built upon the feedback received from the first iteration's workshop. We refined our approach by developing a methodology to establish traceability between work artifacts(Boundary Objects) and the required artifacts(Boundary Objects) of ISO 26262 Part 6. This phase involved further investigation into existing traceability and compliance tools, along with an expanded literature review on ISO 26262 and compliance. The solutions developed were again presented and evaluated in a workshop, fostering a dialogue with participants about the representation of the compliance results and its applicability.

4.4.3 Iteration 3

Iteration 3, the final phase, concentrated on addressing the limitations and problems identified in the previous iterations, specifically targeting improvements in the TTIM, the compliance check results to include "Roles" involved and UML representations. Instead of seeking new challenges, this iteration was dedicated to refining and evaluating the tool based on the feedback gathered. Evaluation sessions featured a focus group consisting of master's students with extensive knowledge in T-Reqs and software engineers with significant experience. These evaluations provided valuable insights into the tool's functionality and areas for improvement, contributing significantly to the refinement of our approach.

5

Findings

We go deep into our study in this chapter with the results of three thorough iterations of the design science cycle. We successfully navigated the difficulties of creating a novel artifact intended to address our problem statement through careful preparation, execution, and evaluation. With every iteration, we made progress in perfecting our solution by incorporating insightful knowledge from both industry experts and literature.

The final artifact and the new features that were implemented are presented in chapter 5.1. Additionally, the findings from each iteration in the design science cycle are documented in chapters 5.2, 5.3, and 5.4.

5.1 The Artifact : The Improved T-Reqs Tool for automated compliance checking

The T-Reqs tool has undergone significant improvements to enhance its functionality and usability, particularly in facilitating compliance checking for ISO26262 standards [34]. These advancements are centered around the introduction of a new TReq Type and Traceability Information Model (TTIM) based on the Boundary Objects between Methodological Islands (BOMI) model. This aims to streamline the process of defining and tracing relationships between various work artifacts required by a particular defined standard within a project, thereby improving traceability, clarity, and ease of updates in the compliance check process, audits, and overall project management [34].

T-Reqs Feature	BOMI related parts	ISO 26262 related parts
Creation of new TTIM	All main parts were defined(BO, MI, Role, Driver, Governance team	Not involved with this feature
Compliance Check (Tracing to required work artifacts by a particular standard)	Boundary Objects(using the "boundaryObject" type of the new TTIM	"Work products" clauses defined by the standard as deliverables for evidence to compliance
Compliance Check (Tracing to roles responsible for the required artifacts by a particular standard)	Roles (using the "responsibleFor" attribute of the TTIM "role" type and "relatesTo" attribute of boundary objects	Roles not explicitly defined.

Table 5.1: Relationship between the main parts of the artifact (T-Reqs, BOMI and ISO26262 Part 6

5.1.1 The New TTIM

The newly introduced TTIM is a pivotal component of the enhanced T-Reqs tool . It provides a structured framework for defining types and traces in accordance with the BOMI metamodel.

```

name: Sample T-Reqs Type and Trace Information Model (TTIM) for Boundary Objects between Methodological Islands
version: 0.0.1
description: An example how types and traces in the BOMI Metamodel can be defined in T-Reqs.
author: Michael Osei Aduamah, Michiale Hadgu
types:
####
# Boundary Objects (BO): Tools, deliverables or artefacts that
# cross boundaries between methodological islands ('groups')
- name: boundaryObject
  links:
  - type: coordinatesBetween
    target: methodologicalIsland
  - type: relatesTo
  - type: hasParent
    target: boundaryObject

####
# Methodological Islands (MI): Groups whose way of working
# differs from the rest (e.g. teams in
# different locations)
- name: methodologicalIsland
  links: []

####
# Governance Team : Roles are part of a Governance Team that
# governs a Boundary Object
- name: governanceTeam
  links:
  - type: governs
    target: boundaryObject
    required: 'true'

# Roles: The function fulfilled by a person, team, or group.
- name: role
  links:
  - type: partOf
    target: governanceTeam
  - type: partOf
    target: methodologicalIsland
  - type: responsibleFor
    target: boundaryObject
  - type: creates
    target: boundaryObject
  - type: reads
    target: boundaryObject
  - type: updates
    target: boundaryObject
  - type: deletes
    target: boundaryObject
  - type: impacts
    target: boundaryObject

####
# Driver: A type of Technology, Process or Business that drives an MI
- name: driver
  links:
  - type: drives
    target: methodologicalIsland
    required: 'true'

```

Figure 5.1: The new BOMI based TTIM

This model categorizes T-Reqs elements into several key types, including:

Boundary Objects (BO): Artefacts that traverse boundaries between methodological islands, serving as tools, deliverables, or other essential documentation can be tagged as boundary objects within T-Reqs.

Methodological Islands (MI): Distinct groups with differing ways of working, such as teams located in various geographical locations can also be defined within a T-reqs enabled project scope.

Governance Team: A collective responsible for governing boundary objects, emphasizing the importance of oversight and management.

Roles: Functions performed by individuals, teams, or groups, encompassing responsibilities ranging from creation to deletion of boundary objects.

Driver: Technologies, processes, or business aspects driving a methodological island, highlighting the forces shaping project development.

This TTIM facilitates the labeling of work artifacts as boundary objects and allows for the dynamic definition and updating of roles, governance teams, and drivers within a dedicated document in the project directory. This flexibility ensures that the project structure remains adaptable to changes and evolving requirements.

We identified several advantages that the BOMI-based TTIM offers compared to the direct standard-based approach. Specifically, these included:

Enhanced scalability: The BOMI model's ability to handle complex project structures makes it suitable for large-scale agile projects, ensuring the compliance checking methodology remains effective as the project grows and evolves.

Improved role and team definitions: The explicit definition of roles and their interactions with project components provides a structured approach to managing complex project dynamics, especially beneficial in large organizations.

Better alignment with agile principles: The BOMI model's focus on flexible, adaptive structures aligns well with agile methodologies, making it an ideal choice for projects that require constant adaptation and improvement.

More comprehensive project context: By considering roles, team affiliations, and governance structures, the BOMI-based TTIM provides a more holistic view of the project, reducing the likelihood of overlooking critical aspects during compliance checks.

Improved collaboration across dispersed teams: The clear definition and management of boundary objects facilitate better communication and coordination among team members, crucial in large-scale agile environments.

5.1.2 Enhanced Compliance Checking Capabilities

A significant addition to T-Reqs is the introduction of a dedicated compliance folder designed to house all defined standards in markdown format.

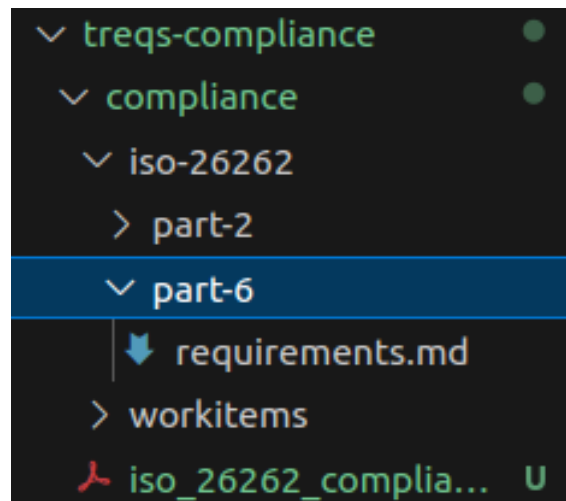


Figure 5.2: The compliance folder with defined standards

Initially populated with ISO26262 Part 6 definitions, this folder enables easy modification and tagging of required work products and roles, enhancing traceability and clarity for stakeholders. The usage of an editable markdown file for definition of standards allows for straightforward updates to the standard, ensuring that compliance checks remain current and accurate.

A new T-Reqs command, "treqs compliance", was also introduced to activate compliance check operations.

```
Usage: treqs [OPTIONS] COMMAND [ARGS]...

Options:
  --help  Show this message and exit.

Commands:
  check          Checks for consistency of treqs elements.
  compliance    Check compliance against a standard.
  create        Creates a treqs element and prints it on the command line.
  createlink    Creates a link to a treqs element.
  generateid    Generate a new id used for treqs element.
  list          List treqs elements in this folder
  process       Process a treqs-controlled file, i.e.
```

Figure 5.3: The compliance command with other default treqs commands

Once "treqs compliance" is run, it scans recursively through the "compliance" folder for all defined standards and shows a list of available definitions to choose from.

This command generates log outputs detailing the compliance status against the defined and selected standard, including completion percentages, required work items, locations, fulfillment statuses, and roles responsible.

5. Findings

```
-----
COMPLIANCE CHECK RESULTS TO ISO 26262- 14 % COMPLETE.
-----
Standard | Part | Required Work Item | Location | Fulfilled? | Work Item Location | Role Responsible
-----
ISO 26262 | 6 | ### 5.5.1 Documentation of the Software Development Environment | compliance/iso-26262/part-6/requirements.md:145 | YES | compliance/workitems/sw-dev-en
v-doc.md:1 | Development Team Lead
ISO 26262 | 6 | ### 6.5.1 Software Safety Requirements Specification | compliance/iso-26262/part-6/requirements.md:236 | NO | Not found | Mr. John Doe- Senior QA Engin
eer
ISO 26262 | 6 | ### 6.5.2 Hardware-Software Interface (HSI) Specification (Refined) | compliance/iso-26262/part-6/requirements.md:246 | NO | Not found | Not found
ISO 26262 | 6 | ### 7.5.1 Software architectural design specification | compliance/iso-26262/part-6/requirements.md:485 | NO | Not found | Not found
ISO 26262 | 6 | ### 7.5.2 Safety analysis report | compliance/iso-26262/part-6/requirements.md:418 | NO | Not found | Not found
ISO 26262 | 6 | ### 7.5.3 Dependent failures analysis report | compliance/iso-26262/part-6/requirements.md:414 | NO | Not found | Not found
ISO 26262 | 6 | ### 8.5.1 Software Unit Design Specification | compliance/iso-26262/part-6/requirements.md:526 | NO | Not found | Not found
-----
Choose an option:
1) Export Compliance Results
2) Exit
Enter the corresponding number: █
```

Figure 5.4: A sample log output after compliance check has been run

This feature provides a clear and concise overview of compliance progress, which helps to improve auditing and collaboration among stakeholders.

5.1.3 Export to PDF Functionality

To further support auditing and collaborative efforts, an export to PDF functionality has been added.

```
-----
Choose an option:
1) Export Compliance Results
2) Exit
Enter the corresponding number: 1
Exporting data...

-----
COMPLIANCE CHECK RESULTS EXPORTED TO compliance/iso_26262_compliance_report.pdf.
-----
```

Figure 5.5: A sample log output after compliance check has been run

This allows developers and stakeholders to easily generate comprehensive reports on compliance check results. These PDF documents serve as valuable resources for review meetings, audits, and sharing insights across teams, enhancing transparency and accountability in the compliance checking process.

5.2 Iteration I

Our first iteration aimed at addressing the coordination needs in large scale agile system development when working with standards. We represented the main parts of the BOMI metamodel in TReq subject to further improvements to enable the definition of boundary objects, roles, teams, methodological islands, drivers and governance teams in system projects. We did not focus on the implementation of compliance checking to ISO 26262 in this phase but still gathered some information during the problem identification stage that could be of help during our next iteration.

5.2.1 Problem Identification

In this phase, we highlight the findings from identified problems related to standards compliance and coordination in large scale agile system development. These problems were identified using related literature, and a workshop. From our literature review some notable challenges were identified that affected Large-scale agile when attempting to comply with standards such as ISO 26262, particularly in terms of documentation management and traceability to work artifacts. These challenges are notable in cultures that prioritize agility and informality, reconciling ISO 26262-compliant documentation management with agile methodologies seemed to be a significant hurdle

5.2.1.1 Problem 1: Documentation Management Conflict

The conflict between ISO 26262's requirements and agile methodologies stems from the former's need for precise, concise, and structured documents that are straightforward to trace and comprehend by the target audience. This necessity stands in stark contrast to agile's inclination towards informal and adaptable documentation [1]. As highlighted by Participant 7 during the first workshop, "I think there's one of the standards I don't remember. If it was 26262 or Aspice, it actually requires you to have the responsible people or like an overview of stakeholders that work on the process to be documented somewhere." This statement highlighted the challenge of translating abstract standards into tangible, actionable documentation within an agile framework.

On the other hand, Participant 4 shared her experience working with ISO 26262, "So I work with ISO 26262 and the challenges usually that you have like it's a long list of documents or like things that you need to comply with and it's not usually the same Participant that is creating the documents and working on the documents and so on." She pointed out that it's uncommon for the same individual to both create and manage these documents, complicating coordination and traceability efforts. "So it makes things very difficult in terms of coordination like do we, how do we fulfill it in what way and trace it to the documents as well," she said, illustrating the practical difficulties encountered in navigating the documentation landscape under ISO 26262.

5.2.1.2 Problem 2: Abstract and Evolving Standards

The challenges posed by abstract and evolving standards, such as ISO 26262, were extensively discussed during the workshop, revealing key concerns around role identification, requirement consistency, and the complexity of managing multiple stakeholders. Participant 7 noted, "the ISO standards are kind of known for having this abstract language where you might have to derive a little bit to know who are the roles involved," highlighting the difficulty in deciphering the implications of these standards without a deep understanding. Participant 8 further elaborated on the potential for inconsistency in requirement definitions, stating, "Developers could lead to inconsistency in their requirement definition because there will be several people working on the same set of requirements, and also because developers often are not

requirement experts." This pointed out the risks associated with multiple contributors, especially those lacking expertise in requirement engineering. Additionally, Participant 8 emphasized the critical importance of high-quality requirements for ensuring clarity across cross-functional teams, noting, "And moreover, it is important to have good quality requirements so that everyone is able to understand the current state of the project when there are cross functional teams working on it."

Participant 4 also commented on the process complexity due to the involvement of many people, "And that is a very lengthy and and complex process because many people are involved", this reflects broader coordination challenges inherent in large-scale agile projects. These insights, derived from both the literature and the workshop discussions, place emphasis on the challenges of integrating ISO 26262 compliance with agile methodologies, particularly in terms of managing required work products.

5.2.2 Solution Suggestion

To enhance documentation management, coordination, and representation of roles within agile environments, we proposed integrating BOMI in T-Reqs. This approach draws on insights from the work of Wohlrab et al., emphasizing the importance of shared work artifacts, also known as boundary objects, in fostering collaboration across different agile teams [16].

By incorporating BOMI into T-Reqs, we can effectively trace, visualize and manage these shared boundary objects that are being shared by different teams. This integration will also facilitate the clear representation of roles, team affiliations (referred to as Methodological Islands), governance structures, and driving factors.

This integration aims to eliminate ambiguity regarding the ownership and responsibilities of individual artifacts, thereby promoting better understanding and cooperation among developers and stakeholders.

5.2.3 Solution Implementation

During the implementation phase of the BOMI TTIM (Type and Trace Information Model) for T-Reqs, we delved into the YAML syntax and its application in defining the model's structure and relationships. YAML, standing for "YAML Ain't Markup Language," is a human-readable data serialization language designed for configuration files and data exchange between languages with different data structures. Its simplicity and readability, coupled with its flexibility in representing complex data structures, made it an ideal choice for defining the TTIM.

The diagram below illustrates the BOMI TTIM and the primary elements of the metamodel.

```

name: Sample T-Reqs Type and Trace Information Model (TTIM) for Boundary Objects between Methodological Islands
version: 0.0.1
description: An example how types and traces in the BOMI Metamodel can be defined in T-Reqs.
author: Michael Osei Aduamah, Michiale Hadgu
types:
####
# Boundary Objects (BO): Tools, deliverables or artefacts that
# cross boundaries between methodological islands ('groups')
- name: boundaryObject
  links:
  - type: coordinatesBetween
    target: methodologicalIsland
  - type: relatesTo
  - type: hasParent
    target: boundaryObject

####
# Methodological Islands (MI): Groups whose way of working
# differs from the rest (e.g. teams in
# different locations)
- name: methodologicalIsland
  links: []

####
# Governance Team : Roles are part of a Governance Team that
# governs a Boundary Object
- name: governanceTeam
  links:
  - type: governs
    target: boundaryObject
    required: 'true'

# Roles: The function fulfilled by a person, team, or group.
- name: role
  links:
  - type: partOf
    target: governanceTeam
  - type: partOf
    target: methodologicalIsland
  - type: responsibleFor
    target: boundaryObject
  - type: creates
    target: boundaryObject
  - type: reads
    target: boundaryObject
  - type: updates
    target: boundaryObject
  - type: deletes
    target: boundaryObject
  - type: impacts
    target: boundaryObject

####
# Driver: A type of Technology, Process or Business that drives an MI
- name: driver
  links:
  - type: drives
    target: methodologicalIsland
    required: 'true'

```

Figure 5.6: The new TTIM based on the BOMI metamodel

The TTIM file begins with metadata specifying the name, version, description, and authors of the model. This metadata is crucial for tracking the evolution of the model and attributing contributions. Following the metadata, the core of the TTIM is defined through a series of type definitions, each encapsulating a specific entity within the BOMI Metamodel. These entities include Boundary Objects (BOs), Methodological Islands (MIs), Governance Teams, Roles, and Drivers, each with its own set of attributes and relationships.

Boundary Objects are defined with links to Methodological Islands, indicating their role in bridging gaps between different working methodologies. Methodological Islands themselves are defined without explicit links, signifying their autonomy but also their potential connections through shared boundary objects. Governance Teams are linked to Boundary Objects, highlighting their governing role over these artifacts. Roles are further detailed, linking them to both Governance Teams and Methodological Islands, reflecting their multifaceted responsibilities within the system. Finally, Drivers are linked to Methodological Islands, emphasizing the driving forces behind different working methodologies.

Following the successful creation of the BOMI TTIM (Type and Trace Information Model), our next objective was to specify certain work artifacts as 'boundary objects' within a sample T-Reqs project.

The figure below shows an instantiation of a T-Reqs element, specifically a Boundary Object.

```
<treqs-element id="be0d7065d63211ee9267145afc7dd63e" type="boundaryObject">
# Impact Analysis Report
## Requested Change:
```

Figure 5.7: Instantiating a T-Reqs element with the new TTIM

In the depicted scenario, an Impact Analysis Report is labeled as a Boundary Object. This designation signifies its role as a shared resource that can be utilized by various teams, thereby promoting collaboration and knowledge sharing.

The figure also demonstrates the establishment of a link to a Methodological Island (MI) that shares the artifact.

The `<treqs-link>` tag specifies a relationship between the Boundary Object and a Methodological Island, utilizing the `type="coordinatesBetween"` attribute to indicate the nature of their interaction. The `target` attribute then provides the unique identification (UID) of the Methodological Island, establishing a direct connection between the two entities.

5.2.4 Solution Validation

Building upon the foundational work of the BOMI creators on ISO 26262 Part 2 [35], we embarked on a detailed implementation of a sample project with workitems and defined roles as required by ISO 26262 Part 2. This involved defining the roles involved in a sample project, specifying the boundary objects required by the

standard (such as the Impact Analysis Report, Item Level Modification Report, etc.), and defining sample methodological islands (representing different teams). The figure below shows the already existing BOMI model of ISO 26262 Part 2 that was used to help validate our implementation in this stage.

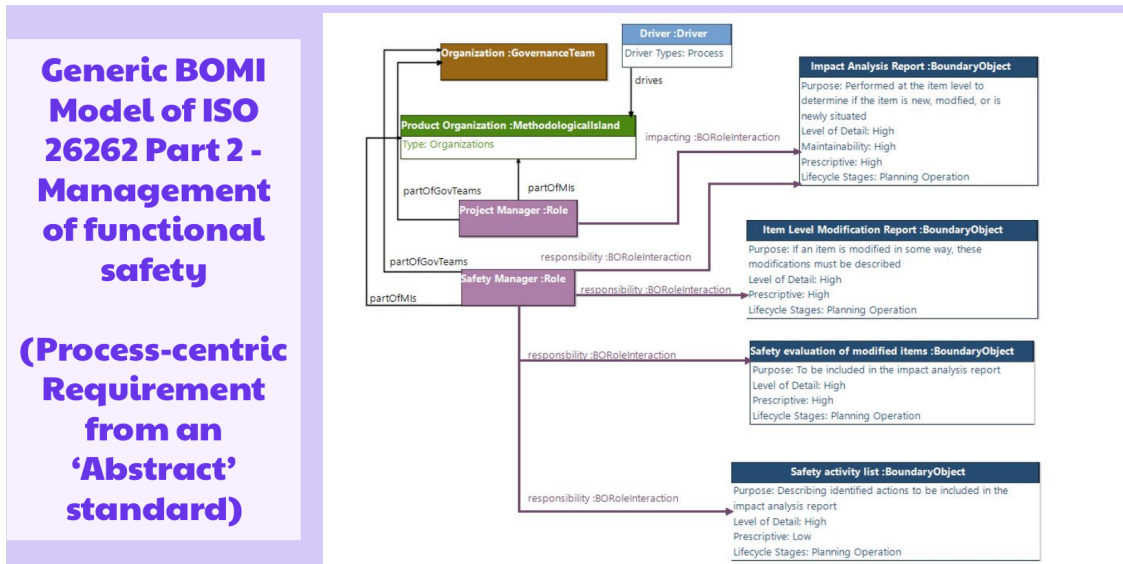


Figure 5.8: BOMI Model of ISO 26262 Part 2

Once the project’s components were defined as T-Reqs elements, we proceeded to autogenerate a UML (Unified Modeling Language) diagram from Treqs based in this new TTIM.

This autogenerated UML was used to mainly assess the effectiveness of the new BOMI TTIM. Specifically, it aimed to validate the tool’s capability to capture all defined roles, drivers, governance teams, methodological islands, boundary objects, and most importantly, establish the correct links between these elements.

The figure below shows the relationships and dependencies within our project, providing a clear and accessible overview within T-Reqs. This visualization not only aids in the internal comprehension of the project but also serves as a valuable reference for external stakeholders, ensuring transparency and alignment with industry standards.

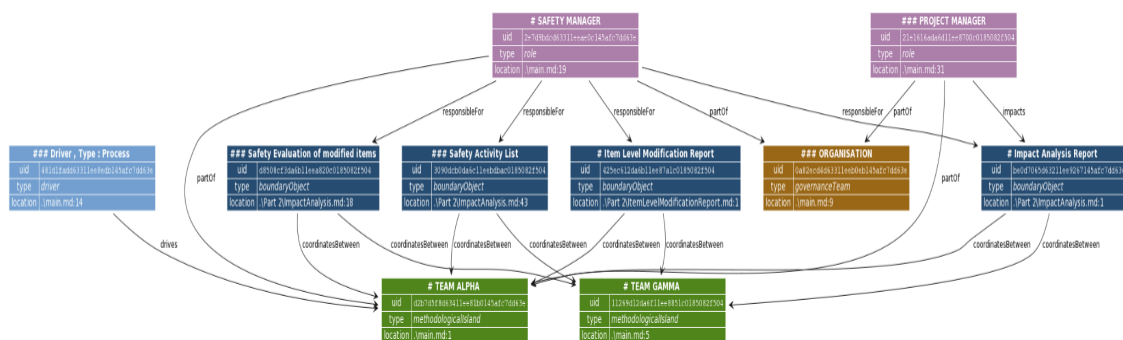


Figure 5.9: A T-Reqs autogenerated uml diagram showing traceability between artifacts using the BOMI TTIM

5.2.5 Solution Evaluation

During the evaluation stage of this iteration, our main goal was to ensure that the new TTIM based on BOMI and its application in T-Reqs accurately reflected the complexities and requirements of system development projects, especially those subject to ISO 26262.

We began by presenting the BOMI TTIM to our participants in Workshop 1, detailing its structure and the rationale behind its design. This included an explanation of how the TTIM was structured and how Treqs elements can be created with the new TTIM types, "boundaryObject", "role", "methodologicalIsland", "governanceTeam" and "driver".

Following this introduction, we moved on to demonstrate the instantiation of the BOMI model for ISO 26262 Part 2 within a hypothetical project. This involved showing the autogenerated UML diagram from Treqs, which depicted the boundary objects, roles, teams, and governance structures involved in the sample project.

The UML diagram also served as a visual aid, helping participants understand the relationships and dependencies between different parts of the project. It highlighted how the BOMI model could be applied in real world T-Reqs enabled projects to manage and coordinate activities in line with ISO 26262 standards, ensuring that all aspects of the project were aligned with functional safety requirements.

Post-presentation, we facilitated a dialogue session, encouraging participants to offer feedback on the UML diagram and the methodology for implementing the BOMI TTIM within T-Reqs. This interactive engagement permitted us to collect firsthand insights from those most affected by our initiatives, ensuring that our proposals were both pragmatic and efficacious.

A noteworthy observation from Participant 1 highlighted the utility of mapping multiple roles and boundary objects, suggesting it could serve as a valuable resource for auditing purposes or even prior to audits. They also acknowledged the challenges faced in adapting the BOMI meta-model to Eclipse, indicating a need for comparison and consideration of alternative representations. While concerns were raised about the graphical models' limitations, it's important to note that they were primarily attributed to the current version of T-Reqs and its constraints. Our future work will address these issues by incorporating association classes, which were identified as essential for expressing complex relationships and dependencies more effectively.

5.3 Iteration II

Our primary objective for this iteration was to explore methods for optimizing T-Reqs to enable automated compliance checking for ISO 26262, leveraging the newly introduced BOMI TTIM.

During this phase, our concentration was on the representation, tracing, and compliance verification of boundary objects defined in the project scope from the BOMI TTIM. This focus was guided by the principal requirements outlined in the "work products" sections of ISO 26262 Part 6, which emphasize the importance of tracing to these deliverables as evidence in achieving compliance.

5.3.1 Problem Investigation

We embarked on an investigation phase following the feedback gathered from the first workshop and the analysis of existing traceability and compliance tools. Firstly, we analyzed the feedback from the workshops, focusing on the areas where participants expressed confusion or requested further clarification. This involved reviewing the UML diagrams, the BOMI TTIM implementation, and the approach to integrating ISO 26262 standards into T-Reqs. The insights gained from this analysis helped us identify specific areas for improvement and refinement.

Next, we investigated existing traceability and compliance tools available in the market. This investigation aimed to understand the current state of technology and identify any tools that could complement our solution or offer alternative approaches to achieving our goals. By comparing these tools with our proposed solution, we were able to highlight the unique advantages of our approach and identify potential synergies.

Our last step for problem investigation phase for this iteration involved an interview with a requirements engineer at Ericsson. The interview lasted for about an hour and provided valuable insights into the challenges faced in ensuring compliance with standards, particularly in the context of managing and updating documentation across large projects. Below are key highlights from the interview:

On the Complexity of Compliance Documentation: "We are compliant with the entire text here, except we have kind of like one exception and it's something in here we don't support or something like that and we have to do that for all their paragraphs in all of that specification like in the entire standard. So there are a few 100 documents. Or a few 100 pages each. Uh, but that we have to provide the specific statements."

Regarding the Manual Nature of Compliance Checks: "Yeah, well, uh, it's a lot pretty much, uh. So it becomes very complicated and it's easy to like miss something and so on. So that's an issue and it requires a lot of work."

"There is standard and it's always an issue when a standard is updated, you have to go through and then add changes and so on and we have to analyze that changes again. So it's quite a lot of work also to do that."

"And in in addition, all our standards are like in PDF or word format and so on. So it's a bit like annoying to like see the changes and so on. It's not very like friendly format to like automate and extract information automatically."

Impact on Team Performance and Dynamics: "Um, yeah, absolutely. because it's the the responsibility of our agile teams as well to update if they make a change like affecting that specific functionality and we have some problem. We have some general system managers that are the main response. So the teams can can always ask the general system managers have for guidance and input as well. But on the other hand, the general system managers are not often available. So they can be quite hard to get hold of. So there is another challenge."

5.3.2 Solution Suggestion

Based on the findings from the investigation phase, we proposed a revised solution that aimed to enhance the representation of ISO 26262 standards within T-Reqs

5. Findings

and establish a clearer and automated traceability between the work artifacts and required work products of the standard.

We decided to use the BOMI TTIM as a blueprint for establishing links between the various work artifacts involved using the "boundaryObject" type defined.

Due to the current version of TReqs limitations, assessing external files wasn't possible. So we planned to describe the standard in markdown within TReqs to facilitate direct trace links.

This markdown document would serve as a central repository for all relevant information about the standard, including its requirements, applicable artifacts, and the roles and responsibilities of different teams. By structuring this information in a such format, we hoped to improve clarity and ease of access for all stakeholders involved in the project.

5.3.3 Solution Implementation

In this implementation phase, we developed a revised solution that aimed to enhance the representation of ISO 26262 standards within T-Reqs and establish a clearer traceability between artifacts and the represented standard using the types defined in the new BOMI TTIM.

```
## Table 1 – Topics to be Covered by Modelling and Coding Guidelines
| Topics | ASIL A | ASIL B | ASIL C | ASIL D |
|-----|-----|-----|-----|-----|
| 1a Enforcement of low complexity | ++ | ++ | ++ | ++ |
| 1b Use of language subsets | ++ | ++ | ++ | ++ |
| 1c Enforcement of strong typing | ++ | ++ | ++ | ++ |
| 1d Use of defensive implementation techniques | + | + | ++ | ++ |
| 1e Use of well-trusted design principles | + | + | ++ | ++ |
| 1f Use of unambiguous graphical representation | + | ++ | ++ | ++ |
| 1g Use of style guides | + | ++ | ++ | ++ |
| 1h Use of naming conventions | ++ | ++ | ++ | ++ |
| 1i Concurrency aspects | + | + | + | + |

**NOTE**: An appropriate compromise of this topic with other requirements of this document may be required.

## 5.5 Work Products

<trreqs-element id="ed9ef676061411ef8c6c0bd417f19d39" type="boundaryObject">
## 5.5.1 Documentation of the Software Development Environment
Resulting from requirements 5.4.1 to 5.4.3 and C.4.1 to C.4.11.
</trreqs-element>
```

Figure 5.10: Representation of ISO 26262 Part 6 in markdown

Figure 5.11 below illustrates the specification of a trace link for a sample software development environment documentation in a sample project. This figure provides a visual representation of how specific artifacts(boundary objects) are shared between teams. This document, required by the ISO 26262 Part 6 Clause 5 is linked to the exact section in the standard. This linkage not only facilitates bi-directional traceability and tracking of compliance but also supports the ongoing assessment and improvement of the project's adherence to the standard.

```

treqs-compliance > compliance > workitems > sw-dev-env-doc.md > ## Conclusion
34
35 ## Collaboration and Documentation
36
37 - Documentation: Use Markdown for documentation and keep it updated in the project's repository.
38 - Communication: Utilize Slack for real-time communication and Zoom for meetings.
39
40 ## Conclusion
41
42 This document serves as a guide to setting up and maintaining a consistent software development environment. It aims to
43 streamline the development process and ensure high-quality code.
44
45 <treqs-link type="relatesTo" target="ed9ef676061411ef8c6c0bd417f19d39" />
46
47 </treqs-element>

```

Figure 5.11: Specifying trace link of software development environment documentation in sample project.

5.3.4 Solution Validation

To assess the validity of our solution, we conducted a series of compliance checks in a sample project using the newly implemented methodology. The results of these checks were analyzed through log outputs and exported as PDF documents for thorough examination. The log outputs provided developers with clickable links to the fulfilled artifacts and the specific clauses of the ISO 26262 standard that they corresponded to. This feature was designed to streamline the process of identifying unfulfilled requirements, allowing developers to quickly navigate to the relevant sections of the standard's Markdown document to determine the actions needed to achieve compliance.

The clickable links embedded in the log outputs served as a direct pathway to the standard's documentation, significantly enhancing the usability and effectiveness of the traceability mechanism. By providing immediate access to the necessary information, developers could efficiently identify and rectify any discrepancies between the project's artifacts and the ISO 26262 standard's requirements. This approach not only facilitated a more accurate and timely compliance check but also empowered developers with the tools needed to actively contribute to the project's adherence to the standard.

COMPLIANCE CHECK RESULTS TO ISO 26262- 11 % COMPLETE.

Standard	Part	Required Work Item	Fulfilled?	Work Item Location
ISO 26262	6	### 5.5.1 Documentation of the Software Development Environment	YES	compliance/workitems/sw-dev-env-doc.md:1
ISO 26262	6	### 6.5.1 Software Safety Requirements Specification	NO	MISSING
ISO 26262	6	### 6.5.2 Hardware-Software Interface (HSI) Specification (Refined)	NO	MISSING
ISO 26262	6	### 7.5.1 Software architectural design specification	NO	MISSING
ISO 26262	6	### 7.5.2 Safety analysis report	NO	MISSING
ISO 26262	6	### 7.5.3 Dependent failures analysis report	NO	MISSING
ISO 26262	6	### 8.5.1 Software Unit Design Specification	NO	MISSING
ISO 26262	6	### 9.5.1 Software Verification Specification	NO	MISSING
ISO 26262	6	### 9.5.2 Software Verification Report (refined)	NO	MISSING

Figure 5.12: Compliance check results exported to pdf

5.3.5 Solution Evaluation

In the final phase of Iteration II, we presented the compliance checking implementation made possible with the BOMI TTIM to a workshop, similar to the one conducted in the first iteration. This presentation included an overview of the changes made based on the feedback and investigation, along with demonstrations of the new methodology for representing ISO 26262 standards and establishing bi-directional traceability between the work artifacts and the standard's requirements.

During the workshop, several participants provided valuable feedback regarding the compliance checking implementation facilitated by the BOMI TTIM. Participant 6 highlighted the tool's potential for improving coordination and traceability, stating, "I think this tool that you created is really nice because then people could go back and forth... Especially if you add like stakeholders or roles responsible, then you also know like whom to contact when looking in the standard itself and also the products themselves." This feedback highlighted the importance of clear responsibility assignment in compliance processes.

Participant 2 raised concerns about the manual interpretation of standards, noting, "It depends on like a manual like interpretation of the standard... So it's probably a bit harder for those higher standards" This comment emphasizes the challenge of accurately interpreting and applying high-level standards across different contexts and teams.

Participant 4 suggested exploring the integration of Git history to enhance the governance aspect of the compliance checking process, mentioning, "Since T Rex is implicitly git connected, you get a lot of history and process information potentially... There could be merge requests that could be reversed that could give you some indication of the governance activities." This idea opened up possibilities for leveraging git version control systems to support compliance efforts.

Overall, the feedback received during the workshop highlighted the potential benefits of the proposed solution for enhancing compliance checking and coordination among teams. However, it also pointed towards areas for further exploration, such as the integration of Git history for governance insights, the challenge of interpreting high-level standards and the representation of roles responsible for required artifacts. These insights will guide the next steps in refining and expanding the capabilities of the compliance checking implementation.

At the end of the presentation a survey was sent out to participants with some questions regarding their perceptions and experiences with the proposed compliance checking solution facilitated by the BOMI TTIM within TReqs. The survey aimed to gather quantitative and qualitative feedback to assess the tool's effectiveness, identify areas for improvement, and gauge its potential impact on systems development processes. Results from the survey are shown in Figure 5.13 below,

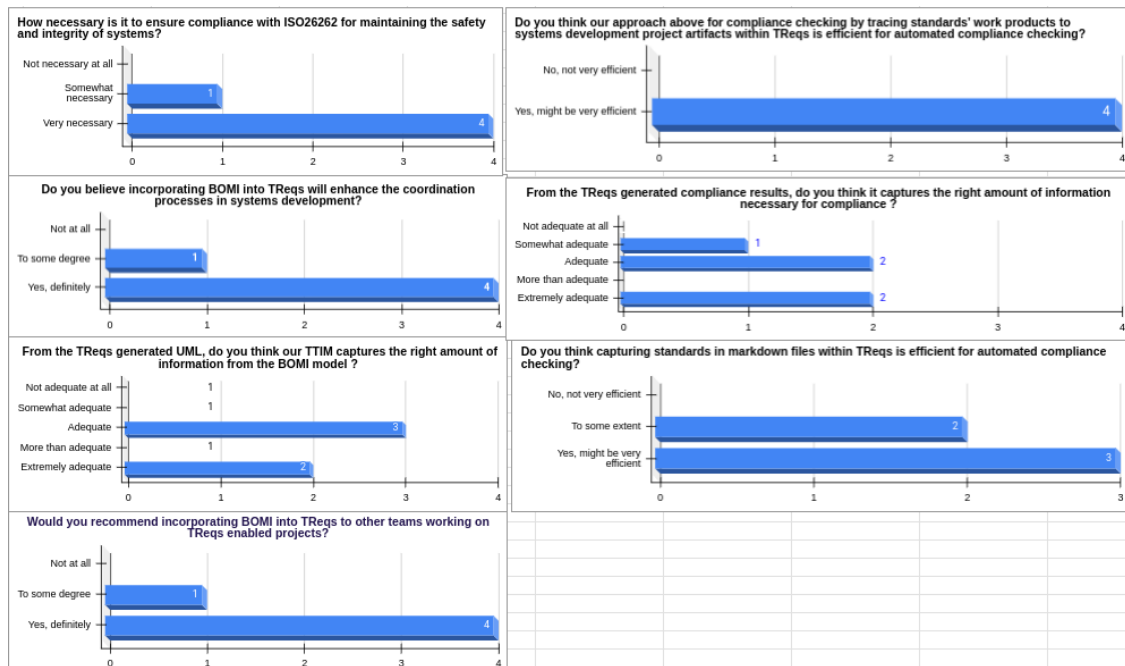


Figure 5.13: Survey Results from Iteration 2

5.3.5.1 Survey Findings:

Perceived Benefits of Incorporating BOMI into TReqs: A significant majority (80%) of respondents saw definite benefits in incorporating BOMI into TReqs for enhancing coordination processes in systems development. This was further supported by an equal percentage recommending this integration to other teams working on TReqs-enabled projects, highlighting the perceived value and potential widespread applicability of the solution.

Adequacy of Information Captured: When assessing the adequacy of information captured from the BOMI model by TTIM, a combined 60% of respondents found it to be adequate. This suggests that the current implementation effectively captures the necessary details for compliance checking, although there is room for improvement as indicated by the 40% who found it somewhat adequate.

Efficiency of Compliance Checking Methods: 60% of respondents found the proposed approach for compliance checking by tracing standards' work products to systems development project artifacts within TReqs very efficient.

Feedback on Compliance Results: The majority (80%) of respondents were satisfied with the adequacy of information captured in the compliance results, ranging from adequate to extremely adequate. This positive reception validates the effectiveness of the compliance checking process.

5.4 Iteration III

During Iteration 3, the focus was primarily on enhancing the tool's effectiveness in representing roles and responsibilities within compliance results. To this effect we utilized the "role" of the BOMI metamodel with its attributes defined in the TTIM

and still focused on Part 6 of ISO 26262.

5.4.1 Problem Identification

Through analyzing the outcomes of Iteration II, it became evident that while the tool successfully facilitated compliance checking and coordination among teams, there was room for improvement in how roles responsible for specific work artifacts were communicated within the compliance documentation. The challenge lay in ensuring that stakeholders could easily identify who was accountable for fulfilling certain standards, thereby streamlining the process of addressing non-compliance issues.

5.4.2 Solution Suggestion

To address this identified problem, a solution was proposed that involved integrating explicit indications of roles responsible for work artifacts directly within the defined standard documents. Specifically, under the section detailing required artifacts, a TRreqs element with the BOMI based TTIM type "role" would be added to denote which roles are accountable for those artifacts. This approach was designed to offer ease of access and enhanced clarity to stakeholders involved in compliance checking processes. By doing so, in instances where a standard was not fulfilled, stakeholders could swiftly navigate to the specific clause from the log output. Right under the work products section, they would find a TRreqs element indicating the responsible roles. If these roles had been previously defined within the system, they would automatically appear in both the log and PDF reports generated by the tool, facilitating immediate action towards compliance. The ISO 26262 standard, particularly Part 6, does not explicitly define roles responsible for compliance because it leaves this task to compliance officers or project managers to denote roles. This approach allows for flexibility in implementation across different companies and project types, recognizing that organizations may have varying structures and roles.

5.4.3 Solution Implementation

In the implementation phase of this iteration, we developed methodologies to track roles associated with each required work product in the selected standard directory.

To facilitate this, roles were defined within the standard documents according to the BOMI TTIM type "role". This approach leveraged the existing framework of the BOMI TTIM to categorize and manage roles systematically. By utilizing the "role" type, we ensured consistency and compatibility with the broader BOMI model, allowing for seamless integration of role definitions into the compliance checking tool.

```

## 5.5 Work Products

<treqs-element id="ed9ef676061411ef8c6c0bd417f19d39" type="boundaryObject">
### 5.5.1 Documentation of the Software Development Environment
Resulting from requirements 5.4.1 to 5.4.3 and C.4.1 to C.4.11.
</treqs-element>
<treqs-element id="4d971572481411ef9fb373c5ff3bd930" type="role">
Development Team Lead
<treqs-link type="responsibleFor" target="ed9ef676061411ef8c6c0bd417f19d39" />

</treqs-element>

```

Figure 5.14: Adding roles to required work products

Subsequently, our implementation was modified to recognize and record the roles defined in the standard documents. Through this modification, the tool became capable of identifying and associating roles with their respective responsibilities for work products, thereby enriching the compliance check results with critical accountability information.

With the enhanced tracking capability in place, the next step was to append this role information to the compliance check results. This was achieved by integrating the role data into both the log outputs and PDF exports generated by the tool. In the log outputs and pdf exports, roles associated with each work product were listed alongside the compliance status, providing immediate visibility into accountability.

5.4.4 Solution Validation

To assess the validity of our improved solution, we added some sample roles to our hypothetical project and run a series of compliance checks. The results of these checks were analyzed through log outputs and exported as PDF documents for thorough examination. The log outputs provided developers with clickable links to the fulfilled artifacts and the specific clauses of the ISO 26262 standard that they corresponded to including the newly defined "role responsible" column. Figure 5.9 below shows a sample exported compliance check result.

COMPLIANCE CHECK RESULTS TO ISO 26262- 14 % COMPLETE.

Standard	Part	Required Work Item	Fulfilled?	Work Item Location	Role Responsible
ISO 26262	6	### 5.5.1 Documentation of the Software Development Environment	YES	compliance/workitems/sw-dev-env-doc.md:1	Development Team Lead
ISO 26262	6	### 6.5.1 Software Safety Requirements Specification	NO	Not found	Mr. John Doe- Senior QA Engineer
ISO 26262	6	### 6.5.2 Hardware-Software Interface (HSI) Specification (Refined)	NO	Not found	Not found
ISO 26262	6	### 7.5.1 Software architectural design specification	NO	Not found	Not found
ISO 26262	6	### 7.5.2 Safety analysis report	NO	Not found	Not found
ISO 26262	6	### 7.5.3 Dependent failures analysis report	NO	Not found	Not found
ISO 26262	6	### 8.5.1 Software Unit Design Specification	NO	Not found	Not found

Figure 5.15: Results from new compliance check methodology

5.4.5 Solution Evaluation

The final evaluation stage of this thesis focused on assessing the usability, functionality, and effectiveness of the compliance checking feature of T-Reqs for ISO26262 standards. A focus group for this evaluation consisted of two software engineers, one software tester who were introduced to TReqs for the first time. It also included

one software engineering master's student, and a DevOps engineer, who were very familiar with TReqs. The participants were tasked with setting up TReqs, performing specific operations, and answering follow-up questions to gauge their experience and understanding of the tool.

5.4.5.1 Evaluation Tasks Overview

Setup and Installation: Participants were required to install Python3, set up a GitLab account, clone the TReqs repository, and install TReqs along with the necessary dependencies. They needed to verify the installation by running `treqs` and confirming the presence of the "compliance" option.

Creation of New Elements: Participants were instructed to create new TReqs elements of types "boundaryObject" and "role", linking them appropriately to work products under the predefined ISO26262 standard in T-Reqs.

Linking Roles to Work Items: Participants added "treqs link" elements with type "responsibleFor" to newly created roles, targeting specific work items.

Evaluation of Fulfillment: Participants evaluated the fulfillment of required standard work products by tagging additional files as boundary objects and establishing links to relevant work artifacts.

Compliance Checking: After completing the tasks, participants ran the "treqs compliance" command, selected ISO26262, and verified the log output/PDF export for the compliance check results showing the fulfilled/non-fulfilled work items and roles responsible.

Following the completion of the tasks, participants were asked to answer a series of evaluation questions designed to assess their overall experience with TReqs, including ease of setup, clarity of instructions, usefulness of the tool in identifying compliance issues, and suggestions for improvements. The results of these evaluations provided valuable insights into the strengths and weaknesses of the compliance checking feature, highlighting areas where the tool excelled and others where enhancements could be made.

Figure 5.16 below shows the results from the evaluation.

Describe briefly your role if you are an employed or your course of study if you are a student: Your answer	Software Developer	Software Tester	Software Engineering	DevOps Engineer	Software Engineer
Have you worked with TReqs or familiar with how it works?	Yes	Yes	Yes	Yes	Yes
To what extent do you think TReqs' compliance check functionality will help improve coordination and compliance check activities in system development projects? Not likely (1) to Extremely Likely (5)	5	4	5	4	4
To what extent do you think this implementation enhance the effectiveness and efficiency of the traditional manual compliance checking process?" Not so effective (1) to Very Effective (5)	5	4	5	3	4
To what extent do you think the representation of roles and boundary objects will enhance team performance and project success? Not likely (1) to Very Likely (5)	5	4	5	3	4
How straightforward and intuitive was the process of tagging boundary objects and assigning roles within the project directory? Not clear (1) to Very clear (5)	4	3	4	3	3
How clear and understandable are the compliance check results presented in the exported PDF format ? Not clear (1) to Very clear (5)	5	5	3	3	5
How clear and understandable are the defined standards presented in markdown ? Not clear (1) to Very clear (5)	5	5	4	4	4
How likely are you to recommend this TReqs feature to development teams that require automated compliance check activities ? Not likely (1) to Extremely Likely (5)	5	5	4	4	4
Rate your overall satisfaction using this TReqs automated compliance check feature. Not satisfied (1) to Very satisfied (5)	5	4	4	3	4

Figure 5.16: Results from artifact evaluation with focus group

6

Discussion

In this section we discuss our outcomes from this study in relation to our research questions.

6.0.1 RQ1: How can the integration of the BOMI meta-model within T-Reqs enhance coordination and support compliance checking, and what role do boundary objects play in this process?

The integration of the BOMI metamodel within T-Reqs has shown promising results in enhancing coordination and supporting compliance checking. The use of boundary objects as shared work artifacts fosters collaboration across different agile teams, addressing the challenge of managing documentation and traceability in large-scale agile projects. The BOMI TTIM also allows for the clear representation of roles, team affiliations, governance structures, and driving factors, which are crucial for understanding and managing the complexities of large-scale projects. This integration has tackled the need for visibility across teams, as boundary objects provide a shared platform for all team members to see the current project state, task statuses, and how different components relate to each other. This increased visibility helps teams understand their work's broader context and how it fits into larger project goals, resolving coordination challenges highlighted by Ghobadi's study on cross-functional software development teams [24]. The BOMI TTIM has also demonstrated its effectiveness in supporting compliance checking by serving as a blueprint to define roles responsible and tag required work items as boundary objects to enable the traceability and compliance check processes. The results reduce ambiguity regarding the ownership and responsibilities of individual artifacts, thereby promoting better understanding and coordination among developers and stakeholders.

6.0.2 RQ2: How can T-Reqs be optimized to automate compliance checking for ISO 26262 using BOMI, and what are the benefits of this approach?

Optimizing T-Reqs for automated compliance checking for ISO 26262 using BOMI has been successful in enhancing traceability and compliance management. The introduction of a dedicated compliance folder with standards defined in markdown format simplifies the process of updating and tagging required work products and roles. This approach not only improves traceability but also ensures that compli-

ance checks remain current and accurate. The benefits of this approach include a more simplified automated compliance checking processes, which addresses challenges identified by Maro et al., such as purposeful traceability, cost-effectiveness, value recognition, portability, trustworthiness, configurability, scalability, ubiquity [29].

6.0.3 RQ3: How effective is the use of T-Reqs and BOMI for automating compliance checking for ISO 26262 in system projects, and what are the key factors that contribute to their success or failure?

The effectiveness of using T-Reqs and BOMI for automating compliance checking for ISO 26262 in system projects has been validated through iterative testing and evaluation. The introduction of explicit indications of roles responsible for work artifacts directly within the defined standard documents has significantly improved the clarity and accountability in compliance documentation. The compliance checking feature, facilitated by the BOMI TTIM, has proven to be efficient and user-friendly, as evidenced by the positive feedback from workshop participants and survey results. The new artifact addresses identified challenges like integration with existing processes, cross-functional collaboration, resource allocation, traceability maintenance, and continuous compliance verification [25]. It also aligns Wohlrab's findings on the importance of traceability visibility across teams and efficient communication facilitated by traceable boundary objects [15]. However, areas for further exploration, such as the integration of Git history for governance insights highlight ongoing opportunities for refinement and expansion of the tool's capabilities.

6.1 Threats to Validity

Compliance Variabilities Across Standards: The effectiveness of the enhancements to T-Reqs may vary depending on the specific standards being applied. While the implementation should work well for standards that require detailed documentation of workproducts as evidence for compliance and traceability, it may face challenges with standards that do not mandate such extensive documentation practices.

Generalizability: The findings and enhancements to the T-Reqs tool may not be universally applicable to all software development projects, as the effectiveness can depend on the specific context, including the nature of the project, the team's familiarity with the tool, and the complexity of the system being developed.

Technical Dependencies: The study assumes that the integration of the BOMI metamodel and the use of Markdown for representing ISO standards in T-Reqs will be technically feasible and will not introduce new limitations or challenges.

Tool Limitations: The research is based on the current capabilities of T-Reqs and the BOMI metamodel. Future updates or changes to these tools could affect the applicability or effectiveness of the enhancements proposed.

7

Conclusions and Future Work

This study has effectively identified the challenges encountered in large-scale agile development that impede coordination and adherence to standards. Insights were gained through a comprehensive review of existing literature, alongside inputs from workshops and an interview with industry professional. The study's findings aim to mitigate the labor-intensive manual compliance checks typically associated with verifying fulfilled artifacts necessary for compliance. The improvements made to the traceability tool T-Reqs has proven to help reduce the labor intensive by automating checks and traceability to these required work artifacts thereby improving coordination, clarity and compliance in large scale agile system development. association classes, which were identified as essential for expressing complex relationships and dependencies more effectively.

7.0.1 Future Work

Future work will build upon the foundation laid by this study, aiming to extend the application of T-Reqs and BOMI to other standards similar to ISO 26262. Examples of such standards include IEC 61508 for functional safety in electrical/electronic/programmable electronic systems, ISO 9001 for quality management systems, and ISO 27001 for information security management. The goal is to develop a more versatile and adaptable tool that can cater to the needs of various industries and regulatory frameworks.

Furthermore, future research will focus on enhancing the compliance results within T-Reqs to include governance teams more comprehensively and the TTIM association classes for defining more complex relationships. This involves developing mechanisms to capture and analyze feedback from developers and teams utilizing the tool for their compliance check efforts. Engaging these users directly will provide invaluable insights into the tool's strengths and weaknesses, paving the way for targeted improvements.

References

- [1] B. Gallina and M. Nyberg, “Reconciling the iso 26262-compliant and the agile documentation management in the swedish context,” in *Critical Automotive Applications: Robustness Safety (CARS)*, (Paris, France), HAL Id: hal-01192981, Sep. 2015. [Online]. Available: <https://hal.archives-ouvertes.fr/hal-01192981/document>.
- [2] S. H. Maro, “Improving software traceability tools and processes,” PhD Thesis, University of Gothenburg, 2024. [Online]. Available: <https://gupea.ub.gu.se/handle/2077/65837>.
- [3] T. SÜD. “Iso 26262 automotive.” (2024), [Online]. Available: <https://www.tuvsud.com/en-us/services/functional-safety/iso-26262-automotive>.
- [4] S. Maro, “Addressing traceability challenges in the development of embedded systems,” 2017. [Online]. Available: <https://api.semanticscholar.org/CorpusID:114499981>.
- [5] M.-A. Peraldi-Frati and A. Albinet, “Requirement traceability in safety critical systems,” in *CARS ’10: Proceedings of the 1st Workshop on Critical Automotive applications: Robustness Safety*, New York, NY, USA: ACM, Apr. 2010, pp. 11–14. DOI: 10.1145/1772643.1772647.
- [6] treqs on git, *Treqs ng*, 2021. [Online]. Available: <https://gitlab.com/treqs-on-git/treqs-ng>.
- [7] E. Knauss, G. Liebel, J. Horkoff, *et al.*, “T-reqs: Tool support for managing requirements in large-scale agile system development,” in *2018 IEEE 26th International Requirements Engineering Conference (RE)*, 2018, pp. 502–503. [Online]. Available: <https://doi.org/10.48550/arXiv.1805.02769>.
- [8] R. Kasauli, G. Liebel, E. Knauss, S. Gopakumar, and B. Kanagwa, “Requirements engineering challenges in large-scale agile system development,” in *2017 IEEE 25th International Requirements Engineering Conference (RE)*, 2017, pp. 352–361. DOI: 10.1109/RE.2017.60.
- [9] R. Wohlrab, J. Horkoff, R. Kasauli, S. Maro, J.-P. Steghöfer, and E. Knauss, “Modeling and analysis of boundary objects and methodological islands in large-scale systems development,” Springer, 2020, p. 42. DOI: 10.1007/978-3-030-62522-1_42.
- [10] I. Inayat, S. S. Salim, S. Marczak, M. Daneva, and S. Shamshirband, “A systematic literature review on agile requirements engineering practices and challenges,” *Computers in Human Behavior*, vol. 51, no. Part B, pp. 915–929, 2015, ISSN: 0747-5632. DOI: 10.1016/j.chb.2014.10.046.

- [11] R. Kasauli, G. Liebel, E. Knauss, S. Gopakumar, and B. Kanagwa, "Requirements engineering challenges in large-scale agile system development," in *Proc. of 25th Reqs. Eng. Conf. (RE)*, Lisbon, Portugal, 2017.
- [12] V. Vu, "Bomi view types: A design science research study," Bachelor of Science Thesis in Software Engineering and Management, Your University Name, Month of Publication 2024.
- [13] R. Wohlrab, P. Pelliccione, E. Knauss, and M. Larsson, "Boundary objects and their use in agile systems engineering," *Journal of Software: Evolution and Process*, vol. 30, no. 1, e2166, 2019. DOI: 10.1002/smr.2166. [Online]. Available: <https://doi.org/10.1002/smr.2166>.
- [14] *Iso 26262-6:2018(e) road vehicles — functional safety — part 6: Product development at the software level*, International Organization for Standardization (ISO), 2018. [Online]. Available: <https://standards.iteh.ai/catalog/standards/sist/f041e63d-1a8b-4c7e-8538-60b3df3b7bce/iso-26262-6-2018>.
- [15] R. Wohlrab, "Living boundary objects to support agile inter-team coordination at scale," Doktorsavhandlingar vid Chalmers tekniska högskola. Ny serie: 4736, Ph.D. dissertation, Chalmers Tekniska Högskola, 2020. [Online]. Available: <https://research.chalmers.se/en/publication/515968>.
- [16] R. Wohlrab, E. Knauss, J.-P. Steghöfer, S. Maro, A. Anjorin, and P. Pelliccione, "Collaborative traceability management: A multiple case study from the perspectives of organization, process, and culture," *Requirements Engineering*, vol. 23, no. 1, pp. 21–45, 2018. DOI: 10.1007/s00766-018-0306-1.
- [17] *Chalmers university of technology*. [Online]. Available: <http://www.chalmers.se>.
- [18] J.-P. Steghöfer, *T-reqs ng comparison documentation*, 2021. [Online]. Available: <https://gitlab.com/treqs-on-git/treqs-ng/-/blob/master/documentation/comparison.md>.
- [19] Nimonik, *The consequences of non compliance*, <https://nimonik.com/resources/non-compliance-risks/>, 2024.
- [20] W. contributors, *Iso 26262*, 2024. [Online]. Available: https://en.wikipedia.org/wiki/ISO_26262.
- [21] Synopsys. "What is iso 26262?" (2024), [Online]. Available: <https://www.synopsys.com/automotive/what-is-iso-26262.html>.
- [22] J. Sini, M. Violante, and F. Tronci, "A novel iso 26262-compliant test bench to assess the diagnostic coverage of software hardening techniques against digital components random hardware failures," *Electronics*, 2022. DOI: 10.3390/electronics11060901. [Online]. Available: <https://doi.org/10.3390/electronics11060901>.
- [23] V. Antinyan and H. Sandgren, "Software safety analysis to support iso 26262-6 compliance in agile development," *IEEE Software*, vol. 38, pp. 52–60, 2021. [Online]. Available: <https://api.semanticscholar.org/CorpusID:226413829>.
- [24] S. Ghobadi, "Challenges of cross-functional software development teams: A conceptual study," *Journal of Information Technology Management*, Dec. 2011, ISSN: 1042-1319.

-
- [25] J. P. Castellanos Ardila, B. Gallina, and F. u. Muram, “Need for selecting suitable languages for consolidating a generic and normative-agnostic solution, increasing automation levels, tool support, and boosting application in practice by improving usability aspects,” in *J. Softw. Evol. Process.*, Citations: 6, IEEE/ACM, 2022.
- [26] Parasoft, *Iso 26262 software compliance in automotive*, 2021. [Online]. Available: <https://alm.parasoft.com/hubfs/ISO26262-Software-Compliance-Automotive.pdf>.
- [27] C. Brenner. “How to ensure functional safety, according to iso 26262.” (2019), [Online]. Available: <https://blogs.itemis.com/en/how-to-ensure-functional-safety-according-to-iso-26262>.
- [28] B. Kaiser and J. Meyer, “Integration of functional safety in the development process,” *ATZ Elektron Worldw*, vol. 6, pp. 42–46, 2011. DOI: 10.1365/s38314-011-0053-2. [Online]. Available: <https://doi.org/10.1365/s38314-011-0053-2>.
- [29] S. Maro, J.-P. Steghöfer, and M. Staron, “Software traceability in the automotive domain: Challenges and solutions,” *Journal of Systems and Software*, vol. 141, pp. 85–110, 2018. DOI: 10.1016/j.jss.2018.03.060. [Online]. Available: <https://doi.org/10.1016/j.jss.2018.03.060>.
- [30] C. Whelan, *What is a compliance matrix and how can you build one?* <https://www.visiblethread.com/blog/what-is-a-compliance-matrix-and-how-can-you-build-one/>, 2020.
- [31] E. Knauss, “Constructive master’s thesis work in industry: Guidelines for applying design science research,” in *2021 IEEE/ACM 43rd International Conference on Software Engineering: Software Engineering Education and Training (ICSE-SEET)*, IEEE Computer Society Press, 2021. DOI: 10.1109/ICSE-SEET52601.2021.00021.
- [32] R. Orngreen and K. Levinsen, “Workshops as a means, practice, and research methodology: A literature review,” *Electronic Journal of e-Learning*, vol. 15, no. 1, pp. 72–80, 2017. [Online]. Available: <https://files.eric.ed.gov/fulltext/EJ1140102.pdf>.
- [33] T. George. “Types of interviews in research | guide examples.” (Jun. 2023), [Online]. Available: <https://www.scribbr.com/methodology/interviews-research/>.
- [34] M. O. Aduamah and M. H. Araya, *Treqs compliance check*, 2024. [Online]. Available: <https://doi.org/10.5281/zenodo.13318904>.
- [35] J. Horkoff and S. A. Brannen, *Evaluating and applying bomi to standards*, 2024.

A

Appendix 1

A.1 A sample TReqs Type and Traceability Information Model

```
---
name: Sample T-Reqs Type and Trace Information Model (TTIM)
version: 0.0.2
description: An example how types and traces can be defined in T-Reqs.
author: Grischka Liebel, Eric Knauss
types:
# ###
# Requirements are in fact system requirements: A condition or capability that must be
# met by treqs to satisfy its intended use (which we aim to express through stakeholder
# requirements, see below).
- name: requirement
  links:
# ##
# Requirements may relate to any other treqs element and this can be expressed through
# the relatesTo trace link.
# They can also be a refinement of another requirements, which can be expressed
# through the hasParent linktype
# Finally, they should directly or indirectly (e.g. through a parent) be motivated by
# a stakeholder-requirement
- type: relatesTo
- type: hasParent
  target: requirement
- type: addresses
  target: stakeholder-requirement
# ###
# Stakeholder needs describe a challenge that key stakeholders of treqs have.
- name: stakeholder-need
# We do not anticipate the need to trace from stakeholder needs to any elements.
  links: []
# ##
# Stakeholder requirements are requirements from a stakeholder's point of view and
# capture our understanding on why somebody would like to use a tool such as treqs
- name: stakeholder-requirement
  links:
# ##
```

A.2 Implementation

```
You, 7 days ago | 1 author (You)
1 import logging
2 import os
3 import yaml
4 from typing import List
5 from treqs.treqs_element import *
6 from treqs.list_elements import list_elements
7 from reportlab.lib.pagesizes import landscape, legal
8 from reportlab.platypus import SimpleDocTemplate, Table, TableStyle, Paragraph
9 from reportlab.lib import colors
10 from reportlab.lib.styles import getSampleStyleSheet
11
12
13 You, 7 days ago | 1 author (You)
14 class check_compliance:
15     def __init__(self):
16         self.__logger = logging.getLogger('treqs-on-git.treqs-ng')
17         self.__list_items = list_elements()
18         self.treqs_element_factory = treqs_element_factory()
19         self.work_items= []
20         self.compliance_results = []
21         self.standards_list =[]
22         self.standards_list_path =str
23         self.required_work_items = []
24         self.required_work_item_uids=[]
25         self.roles_responsible=[]
26         self.roles_responsible_uids=[]
27         self.processed_targets = []
28         self.selected_standard_part = str
29         self.compliance_percentage = str
30
31
32     def check_compliance(self,recursive,ttim,standards,verbose):
33
34
35         self.standards_list_path = standards
36         # Extract the names of available defined standards from standards.yaml
37         self.standards_list = self.load_standards(standards)
```

```

38
39     # Prompt user to select standard
40     self.selected_standard = self.prompt_standard_selection()
41
42     # Trace required work items in project directory
43     self.trace_requirements(self.selected_standard)
44
45     # Log compliance check results
46     self.log_compliance_report()
47
48     # Export compliance check results
49     self.prompt_options()
50
51 def load_standards(self, file_path):
52     if os.path.isfile(file_path):
53         with open(file_path, 'r') as yaml_file:
54             yaml_data = yaml.safe_load(yaml_file)
55
56             standards_data = yaml_data.get('standards', [])
57
58             # Extract the names of available standards
59             for standard in standards_data:
60                 self.standards_list.append(standard['name'])
61
62     return self.standards_list
63
64
65
66 def prompt_standard_selection(self):
67
68     # Display the list of standards
69     print("\n Please select a standard:")
70     for i, standard in enumerate(self.standards_list, start=1):
71         print(f"{i}. {standard}")
72
73     # Prompt the user for selection
74     selection = input("\n Enter the number assigned to the standard you wish to check compliance against: ")
75
76     # Validate the input and return the selected standard

```

```

76     # Validate the input and return the selected standard
77     try:
78         selection_index = int(selection) - 1
79         if 0 <= selection_index < len(self.standards_list):
80             return self.standards_list[selection_index]
81         else:
82             self._logger.log(30, "Invalid selection. Please try again.")
83             return self.prompt_standard_selection()
84     except ValueError:
85         self._logger.log(30, "Invalid input. Please enter a number.")
86         return self.prompt_standard_selection()
87
88 def trace_requirements(self, selected_standard):
89     #Load treqs elements (required artifacts) from the standard definition
90     standard_directory = self.load_standard_dir(selected_standard)
91
92     self.selected_standard_part = self.load_standard_dir(selected_standard, part=True)
93     # Fetch all treqs elements from the required work items
94     self.required_work_items = self._list_items.get_element_list(standard_directory, treqs_type="boundaryObject")
95
96     # Extract UIDs of required work items
97     self.required_work_item_uids = [item.uid for item in self.required_work_items]
98
99
100     # Fetch all roles with outlinks to the required work items in the selected standard directory
101     self.roles_responsible = self._list_items.get_element_list(standard_directory, treqs_type="role", outlinks=True)
102     self.roles_responsible_uids = [item.uid for item in self.roles_responsible if item.uid not in self.required_work_item_uids]
103
104     # Check for fullfiled standards' artifacts
105     if self.required_work_item_uids:
106         self.append_compliant()
107     else:
108         pass
109
110     # Calculate compliance percentage
111     if self.processed_targets and self.required_work_item_uids:
112         self.compliance_percentage = int(round(self.calculate_compliance_percentage(self.processed_targets, self.required_work
113

```

A. Appendix 1

```
108         pass
109
110
111     # Calculate compliance percentage
112     if self.processed_targets and self.required_work_item_uids:
113         self.compliance_percentage = int(round(self.calculate_compliance_percentage([self.processed_targets,
114         self.required_work_item_uids])))
115     else:
116         self.compliance_percentage = 0
117
118     # Check for unfulfilled standards' artifacts
119     if self.required_work_item_uids:
120         self.append_non_compliant(self.processed_targets)
121     else:
122         pass
123
124     # Check for required roles for each artifact
125     self.append_roles()
126
127
128     return
129
130
131 #check through the selected standard's directory
132 def load_standard_dir(self, selected_standard,part=False):
133     if os.path.isfile(self.standards_list_path):
134         with open(self.standards_list_path, 'r') as yaml_file:
135             yaml_data = yaml.safe_load(yaml_file)
136
137     standards_data = yaml_data.get('standards', [])
138
139     # Extract the names of available standards' parts
140     for standard in standards_data:
141         if standard['name'] == selected_standard:
142             return standard['part'] if part else standard['filePath']
143
144     return None
145
```

```
149
150     for element in self.all_work_items:
151
152         #fetch all items with outlinks to the selected standard's required work items
153         filtered_outlinks = [tl for tl in element.outlinks if tl.target in self.required_work_item_uids and tl.source not in self.
154
155         for tl in filtered_outlinks:
156             source_te = self.tregs_element_factory.get_element_with_uid(tl.source)
157             target_te = self.tregs_element_factory.get_element_with_uid(tl.target)
158             label = source_te.label
159             file_line = "{file}:{line}".format(file=source_te.file_name, line=source_te.placement)
160             file_line2 = "{file}:{line}".format(file=target_te.file_name, line=target_te.placement)
161
162             compliance_result = {
163                 'Standard': self.selected_standard,
164                 'Part': self.selected_standard_part,
165                 'Required Work Item': target_te.label,
166                 'Item Uid': tl.target,
167                 'Location1': file_line2,
168                 'Fulfilled?': 'YES',
169                 'Location2': file_line,
170                 'Role Responsible': "Not found"
171             }
172
173             self.compliance_results.append(compliance_result)
174             self.processed_targets.append(tl.target)
175
176
177     return None
178
179
180 def append_non_compliant(self, processed_targets):
181
182     #remove already fulfilled artifacts from the required work item list
183     for target in processed_targets:
184         if target in self.required_work_item_uids:
185             self.required_work_item_uids.remove(target)
186
187     for workitem in self.required_work_item_uids:
```

```

207 for workitem in self.required_work_item_uids:
208     elem = self.treqs_element_factory.get_element_with_uid(workitem)
209     label = elem.label
210     file_line = "{file}:{line}".format(file=elem.file_name, line=elem.placement)
211
212     compliance_result = {
213         'Standard': self.selected_standard,
214         'Part': self.selected_standard_part,
215         'Required Work Item': label,
216         'Item Uid': workitem,
217         'Location1': file_line,
218         'Fulfilled?': 'NO',
219         'Location2': 'Not found',
220         'Role Responsible': "Not found"
221     }
222
223     if not any(d == compliance_result for d in self.compliance_results):
224         self.compliance_results.append(compliance_result)
225
226     processed_targets.append(workitem)
227
228 return None
229
230 def append_roles(self):
231     for result in self.compliance_results:
232         uid = result['Item Uid']
233
234         for role_uid in self.roles_responsible_uids:
235             role_req = self.treqs_element_factory.get_element_with_uid(role_uid)
236             links = [link for link in role_req.outlinks if link.target == uid]
237             if links:
238                 for link in links:
239                     role_e = self.treqs_element_factory.get_element_with_uid(link.source)
240                     role_label = role_e.label
241
242                     # Find the index of the current result in self.compliance_results
243                     index = next((index for (index, d) in enumerate(self.compliance_results) if d["Item Uid"] == uid), -1)
244

```

```

245 def calculate_compliance_percentage(self, workitemsDone, workitemsNotDone):
246     common_elements = set(workitemsDone).intersection(set(workitemsNotDone))
247     num_common_elements = len(common_elements)
248
249     total_elements = set(workitemsDone).union(set(workitemsNotDone))
250     num_total_elements = len(total_elements)
251
252     if num_total_elements == 0:
253         return "Cannot calculate compliance percentage. Division by zero."
254
255     percentage = (num_common_elements / num_total_elements) * 100
256
257     return percentage
258
259 def log_compliance_report(self):
260     self.__logger.log(20, "\n\n")
261     self.__logger.log(20, "COMPLIANCE CHECK RESULTS TO %- %s %% COMPLETE." % (self.selected_standard, self.compliance_percentage))
262     self.__logger.log(20, "-----")
263     self.__logger.log(50, "| Standard | Part | Required Work Item | Location")
264     self.__logger.log(20, "-----|-----|-----|-----")
265
266     for result in self.compliance_results:
267         self.__logger.log(20, f"| {result['Standard']} | {result['Part']} | {result['Required Work Item']} | {result['Location1']}")
268
269 def export_compliance_report(self):
270     # Prepare the data for the table
271     data = [['Standard', 'Part', 'Required Work Item', 'Fulfilled?', 'Work Item Location', 'Role Responsible']]
272     for result in self.compliance_results:
273         data.append([result['Standard'], result['Part'], result['Required Work Item'], result['Fulfilled?'], result['Location2'], result['Role Responsible']])
274
275     # Create a PDF document in landscape orientation
276     isoname = self.selected_standard.lower().replace(" ", "_")
277     filename = f"compliance/{isoname}_compliance_report.pdf"
278
279     doc = SimpleDocTemplate(filename, pagesize=landscape(legal))
280
281     # Adding a header for the table
282     styles = getSampleStyleSheet()
283     header = Paragraph("COMPLIANCE CHECK RESULTS TO %- %s %% COMPLETE." % (self.selected_standard, self.compliance_percentage))

```

A. Appendix 1

```
266
267 # Adding a header for the table
268 styles = getSampleStyleSheet()
269 header = Paragraph("COMPLIANCE CHECK RESULTS TO %- %S %% COMPLETE." % (self.selected_standard, self.compliance_percentage),
270 table = Table(data)
271
272 # Styling table
273 table_style = TableStyle([
274     ('BACKGROUND', (0, 0), (-1, 0), colors.grey),
275     ('TEXTCOLOR', (0, 0), (-1, 0), colors.black),
276     ('ALIGN', (0, 0), (-1, -1), 'CENTER'),
277     ('FONTNAME', (0, 0), (-1, 0), 'Helvetica-Bold'),
278     ('FONTSIZE', (0, 0), (-1, 0), 14),
279     ('BOTTOMPADDING', (0, 0), (-1, 0), 12),
280     ('BACKGROUND', (0, 1), (-1, -1), colors.antiquewhite),
281     ('GRID', (0, 0), (-1, -1), 1, colors.black)
282 ])
283
284
285
286 # Style cells based on the compliance status
287 for row in range(len(data)):
288     if data[row][3] == "YES":
289         table_style.add('TEXTCOLOR', (3, row), (3, row), colors.green)
290         table_style.add('TEXTCOLOR', (4, row), (4, row), colors.green)
291     elif data[row][3] == "NO":
292         table_style.add('TEXTCOLOR', (3, row), (3, row), colors.red)
293         table_style.add('TEXTCOLOR', (4, row), (4, row), colors.red)
294
295 table.setStyle(table_style)
296
297
298 # Build the PDF
299 doc.build([header, table])
300
301 self.__logger.log(20, '\n\n -----')
302 self.__logger.log(20, 'COMPLIANCE CHECK RESULTS EXPORTED TO %s.', filename)
303
304
```

```
300
301 self.__logger.log(20, '\n\n -----')
302 self.__logger.log(20, 'COMPLIANCE CHECK RESULTS EXPORTED TO %s.', filename)
303
304
305 return
306
307 def prompt_options(self):
308     while True:
309         self.__logger.log(20, '\n\n -----')
310         user_input = input("Choose an option: \n1) Export Compliance Results \n2) Exit\nEnter the corresponding number: ")
311         if user_input == '1':
312             print("Exporting data...")
313             self.export_compliance_report()
314         elif user_input == '2':
315             print("Exiting...")
316             break
317         else:
318             print("Invalid option. Please try again.")
319     return
320
```

A.3 Survey for evaluation in Iteration 2

A SHORT SURVEY FOR A THESIS ON ENHANCING COORDINATION, TRACEABILITY AND STANDARDS COMPLIANCE IN SYSTEM PROJECTS

B *I* U ☰ ✕

TReqs Demonstrator - [link](#)

BOMI Demonstrator - [link](#)

In our thesis, we explore how the BOMI (Boundary Objects between Methodological Islands Model) metamodel might be effectively integrated within TReqs, a traceability tool that is intended to make traceability more effective in large-scale agile systems engineering projects. Our goal is to learn how incorporating BOMI into TReqs-enabled projects may significantly improve coordination and the effectiveness of compliance check activities. Our research is based on the understanding that manual compliance checking is expensive, time-consuming, and error-prone, underscoring the urgent need for automation in this field. Our goal is to enhance best practices in system development by showcasing the potential of automated compliance checking to achieve greater standards of safety, reliability, and quality in software development projects.

Study conducted by:

Michael Osei Aduamah - (gusaduami@student.gu.se)

Michiale Hadgu Araya- (gushadgmi@student.gu.se)

By completing and submitting this form, you confirm that you authorize your responses to be ^{*} recorded and utilized in our thesis. Your identity will remain anonymous. The questions are optional and you are free to skip any as you choose.

Yes, I agree

Do you believe incorporating BOMI into TReqs will enhance the coordination processes in systems development?

Yes, definitely

To some degree

Not at all

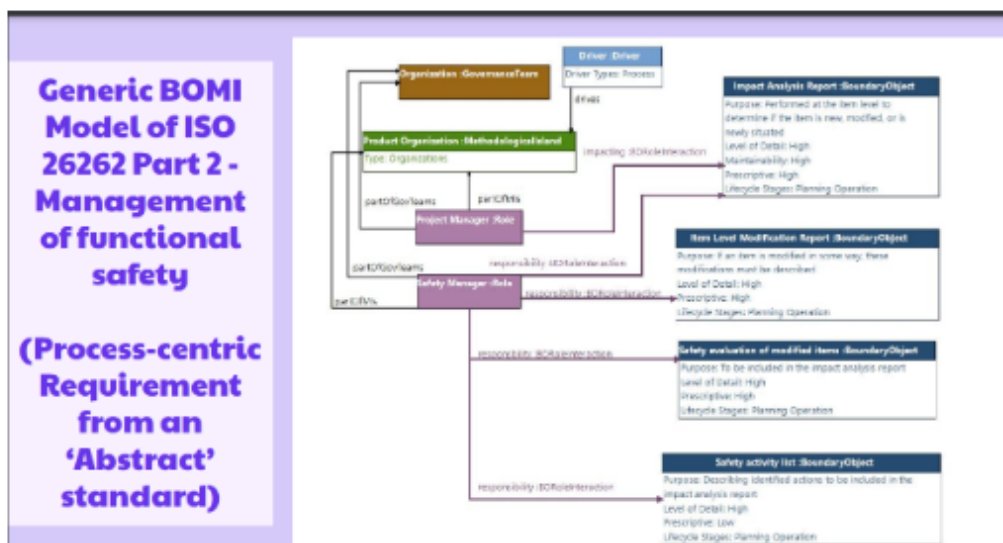
A. Appendix 1

Would you recommend incorporating BOMI into TReqs to other teams working on TReqs enabled projects?

- Yes definitely
- To some extent
- Not at all

...

From the TReqs generated UML [here](#), do you think our TTIM captures the right amount of information from the BOMI model shown below ?



- Not adequate at all
- Somewhat adequate
- Adequate
- More than adequate
- Extremely adequate

Optional: In your opinion, what improvements could be made to the representation of BOMI in TReqs to enhance its effectiveness in systems development?

Long-answer text
.....

In your view, how necessary is it to ensure compliance with ISO26262 for maintaining the safety and integrity of systems?

- Very necessary
- Somewhat necessary
- Not necessary at all

Sample representation of ISO 26262 Part 6 in TReqs

```

1
2
3 ## 5.5 Work Products
4
5 <treqs-element id="ed9ef676861411ef8c6c0bd417f19d39" type="boundaryObject">
6 ## 5.5.1 Documentation of the Software Development Environment
7 Resulting from requirements 5.4.1 to 5.4.3 and C.4.1 to C.4.11.
8 </treqs-element>
9
10
11 # 6. Specification of Software Safety Requirements
12
13 ## 6.1 Objectives
14
15 The objectives of this sub-phase are:
16 - To specify or refine the software safety requirements which are derived from the technical safety co
17 - To refine the requirements of the hardware-software interface initiated in ISO 26262-4:2018, Clause
18 - To define the safety-related functionalities and properties of the software required for the implem
19 - To verify that the software safety requirements and the hardware-software interface requirements are
20 the technical safety concept and the system architectural design specification.
21
22 ## 6.2 General
23
24 The technical safety requirements are refined and allocated to hardware and software during the system
25 6. The specification of the software safety requirements considers in particular constraints of the ha
26 This sub-phase includes the specification of software safety requirements to support the subsequent de
27
28 ## 6.3 Inputs to this Clause
29
30 ## 6.3.1 Prerequisites
31
32 The following information shall be available:
33 - Technical safety requirements specification in accordance with ISO 26262-4:2018, 6.5.1.
34 - Technical safety concept in accordance with ISO 26262-4:2018, 6.5.2.

```

...

Do you think capturing standards in markdown files within TReqs is efficient for automated compliance checking?

- Yes, might be very efficient
- To some extent
- No, not very efficient

Optional: Any reason for your answer above?

Long-answer text

An overview of our current approach to checking for compliance in TReqs enabled projects.



Do you think our approach above for compliance checking by tracing standards' work products to systems development project artifacts within TReqs is efficient for automated compliance checking?

- Yes, might be very efficient
- To some extent
- No, not very efficient

Results from compliance check exported to pdf

iso_26262_compliance - iso_26262_compliance_report.pdf

↑ ↓ 1 of 1 - + Automatic Zoom

COMPLIANCE CHECK RESULTS TO ISO 26262- 11 % COMPLETE.

Standard	Part	Required Work Item	Fulfilled?	Work Item Location
ISO 26262	6	### 5.5.1 Documentation of the Software Development Environment	YES	compliance\workitem\swdev\doc\nd.1
ISO 26262	6	### 6.5.1 Software Safety Requirements Specification	NO	NOT FOUND
ISO 26262	6	### 6.5.2 Hardware-Software Interface (HSI) Specification (Filled)	NO	NOT FOUND
ISO 26262	6	### 7.5.1 Scheme architectural design specification	NO	NOT FOUND

COMPLIANCE CHECK RESULTS TO ISO 26262- 11 % COMPLETE.

Standard	Part	Required Work Item	Fulfilled?	Work Item Location
ISO 26262	6	### 5.5.1 Documentation of the Software Development Environment	YES	compliance/workitems/soa-dev-env/doc.md.1
ISO 26262	6	### 6.5.1 Software Safety Requirements Specification	NO	NOT FOUND
ISO 26262	6	### 6.5.2 Hardware-Software Interface (HSI) Specification (Refined)	NO	NOT FOUND
ISO 26262	6	### 7.5.1 Software architectural design specification	NO	NOT FOUND
ISO 26262	6	### 7.5.2 Safety analysis report	NO	NOT FOUND
ISO 26262	6	### 7.5.3 Dependent failures analysis report	NO	NOT FOUND
ISO 26262	6	### 8.5.1 Software Unit Design Specification	NO	NOT FOUND
ISO 26262	6	### 8.5.1 Software Verification Specification	NO	NOT FOUND
ISO 26262	6	### 9.5.2 Software Verification Report (refined)	NO	NOT FOUND

From the TReqs generated compliance results above, do you think it captures the right amount of information necessary for compliance ?

- Not adequate at all
- Somewhat adequate
- Adequate
- More than adequate
- Extremely adequate

Optional: Any thoughts or suggestions on this sample results from a TReqs compliance check to ISO 26262 Part 6.

Long-answer text

.....

Optional: Would you be willing to participate in a follow-up short interview to provide more detailed feedback on your experience with standards compliance ?

If yes, please enter your email below.

Short-answer text

.....

A.4 Survey for evaluation in Iteration 3

TReqs Compliance Evaluation

This is an evaluation form for a master thesis on "Enhancing Coordination, Traceability and Compliance Checking in Systems Development".

Kindly fill out this form after following the setup and tasks required in the link below

[Setup Guide](#)

We appreciate your time and support!

michael.o.aduamah@gmail.com [Switch accounts](#)



Not shared

What is your role currently?

- Bachelor's student
- Master's student
- PhD student
- Employed

Describe briefly your role if you are an employed or your course of study if you are a student:*

Your answer

Your answer

Have you worked with TReqs or familiar with how it works?

- Yes
- No

To what extent do you think TReqs' compliance check functionality will help improve coordination and compliance check activities in system development

To what extent do you think this implementation enhance the effectiveness and efficiency of the traditional manual compliance checking process?"

	1	2	3	4	5	
Not so effective	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Very effective

To what extent do you think the representation of roles and boundary objects will enhance team performance and project success

	1	2	3	4	5	
Not likely	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Very likely

How straightforward and intuitive was the process of tagging boundary objects and assigning roles within the project directory?

	1	2	3	4	5	
Not clear at all	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Very clear and straightforward

How clear and understandable are the compliance check results presented in the exported PDF format ?

	1	2	3	4	5	
Not clear at all	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Very clear

How clear and understandable are the defined standards presented in markdown ?

	1	2	3	4	5	
Not clear at all	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Very clear

XIV

How likely are you to recommend this TReqs feature to development teams that require automated compliance check activities ?

How likely are you to recommend this TReqs feature to development teams that require automated compliance check activities ?

	1	2	3	4	5	
Not so likely	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Very likely

Rate your overall satisfaction using this TReqs automated compliance check feature

	1	2	3	4	5	
Not satisfied	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Very satisfied

Please share any additional comments, suggestions, or feedback you have regarding this TReqs Compliance Check Feature

Your answer

A.5 Interview Guide

Interview Guide

Introduction

- Welcome the participant and introduce yourself and your role.
- Briefly explain the purpose of the interview and assure confidentiality.

Background Information

- Ask about the participant's current role in system development, focusing on their responsibilities and experiences.

Compliance Standards

- Query about the participant's experience with compliance to various standards, especially in relation to telecommunications and safety standards.
- Discuss the challenges encountered in ensuring compliance to these standards.

Tools and Automation

- Explore the participant's familiarity with tools designed to aid in compliance and automation.
- Discuss the potential benefits and limitations of such tools in streamlining the compliance process.

Challenges and Solutions

- Delve into the difficulties faced in maintaining compliance, including the complexity of standards, the need for constant updates, and the lack of automation in document formats.
- Explore ideas for improving the compliance process, such as automating change detection between document versions and integrating compliance checks into development workflows.

Closing

- Thank the participant for their time and express gratitude for their insights.

|