



Herausragende Masterarbeiten

am Distance and Independent Studies Center

Studiengang:

Software Engineering for Embedded Systems, M.Eng.

Masterarbeitstitel:

Pragmatic Tracing in Model-based Systems and Software Engineering

Autor*in:

Ashwini Sripathi Rao

Declaration

Ich versichere, dass ich diese Masterarbeit selbstständig und nur unter Verwendung der angegebenen Quellen und Hilfsmittel angefertigt und die den benutzten Quellen wörtlich oder inhaltlich entnommenen Stellen als solche kenntlich gemacht habe.

Nürnberg, August 12, 2021

Ashwini Sripathi Rao

Abstract

On the one hand, Model-based Systems and Software Engineering approaches ease the development of complex software systems. On the other, they introduce the challenge of managing the multitude of different artifacts created using various tools during the system lifecycle. For understanding and maintaining these artifacts as they evolve, it is advisable to establish traceability among them. Traceability is the ability to relate the various artifacts created and evolved during the project. However, organizations often consider traceability a burden because it is time-consuming and error-prone when done manually. Hence, the objective of this thesis is to research and develop pragmatic traceability approaches that can be followed in the MBSE context. A systematic mapping study was conducted to understand and compile the various criteria that need to be followed while creating and maintaining trace links. It also provided insights on the approaches followed to ease the burden on engineers. Expert interviews with industrial companies were conducted to investigate the real-life experiences of engineers on traceability, to get an overview of best practices and known pitfalls. Based on the mapping study and the results of the interviews, various approaches and tools used to achieve traceability were discussed. A case study was conducted for state-of-the-practice traceability approaches in a toolchain consisting of Polarion, Enterprise Architect, and Doxygen. For research, open-source libraries and applications were used for analysis. A tool prototype was developed to create and maintain trace links between artifacts created in the toolchain mentioned above. The use cases in which the tool eases achieving traceability are discussed along with pros and cons.

Acknowledgments

I wrote this thesis as part of my distance learning course offered by Technical University of Kaiserslautern in the field of “Software Engineering for Embedded Systems”.

First and foremost, I would like to express my sincere gratitude to Dr.-Ing. Martin Becker, from Fraunhofer Institute of Experimental Software Engineering, who supervised and guided my research work. I am highly thankful to him for sharing his expertise and for valuable guidance, which steered me in the right direction in my research.

I would also like to thank the experts involved in the expert interviews, which helped me immensely in my investigation of the challenges and approaches involved in achieving traceability in real-life industrial scenarios.

I wish to express my sincere gratitude to Dr.-Ing. Peter Liggesmeyer, from Technical University of Kaiserslautern, my first supervisor, for the valuable feedback on the research work and presentations, which helped in the improvement of the thesis.

Last but not least, I express my gratitude to my family members, for their constant support and encouragement throughout my study.

Table of Contents

Declaration	II
Abstract	III
Acknowledgments	IV
Table of Contents	V
Abbreviations	IX
List of Tables.....	X
List of Figures	XI
1 Introduction.....	1
1.1 Motivation	1
1.2 Problem Statement.....	3
1.3 Research Approach.....	4
1.3.1 Research Objectives.....	4
1.3.2 Research Questions	4
1.3.3 Research Method	6
1.4 Thesis Structure	7
2 Related Work	8
2.1 Understanding Traceability	8
2.2 Systematic Mapping Study.....	9
2.2.1 Research Questions	9

2.2.2	Search Strings Used for Online Database Research	10
2.2.3	Inclusion and Exclusion Criteria for Search Results	11
2.2.4	Classification Scheme	11
2.3	Results	14
2.3.1	Results of Database Search.....	14
2.3.2	Answering Mapping Study Research Questions.....	17
2.3.3	Concepts of Traceability	19
2.3.3.1	Classification of Traceability	19
2.3.3.2	Basic Terminologies of Traceability	21
2.3.3.3	Prerequisites for Achieving Traceability.....	22
2.3.3.4	Key Criteria for Traceability Approaches	24
2.3.3.5	Uses of Traceability	25
2.3.4	Traceability in Automotive Domain	27
2.3.5	Traceability in Model-Based Systems and Software Engineering	28
2.4	State of the Practice	31
2.4.1	Traceability in Requirements Management Tools.....	31
2.4.2	Traceability in Application Lifecycle Management Tools	33
2.4.3	Traceability using General-purpose Tools.....	34
2.4.4	Standalone Traceability Tools	35
2.4.5	Traceability in the Issue Tracking Tools	37
3	Expert Interviews on Traceability.....	38

3.1	Introduction	38
3.2	Procedure and Topics Covered in the Interview	38
3.3	Data Collection	40
3.4	Analysis	41
3.5	Conclusions	43
3.5.1	Best Practices to Follow.....	43
3.5.2	Pitfalls to Avoid	44
4	Feasibility Study and Investigation.....	46
4.1	Approaches of Traceability across the Toolchain	46
4.1.1	Used Toolchain	46
4.1.1.1	Managing Traces in Polarion	48
4.1.1.2	Managing Traces in Other Tools.....	49
4.1.2	Establishing Traceability between Polarion and Enterprise Architect	50
4.1.2.1	Using Open Services for Lifecycle Collaboration	50
4.1.2.2	Using Polarion Connector for Enterprise Architect	51
4.1.2.3	Using Requirements Interchange Format.....	53
4.1.2.4	Using CSV Import and Export.....	54
4.1.3	Establishing Traceability between Polarion and Doxygen	55
4.1.4	Establishing Traceability between Enterprise Architect and Doxygen	56
4.2	Proposed Solution.....	56
4.2.1	Traceability Information Model.....	57

4.2.2	Features of the Solution	60
4.2.3	Use Cases	61
4.2.4	Traceability Link Creation and Maintenance Process using TraceGen.....	63
4.2.4.1	Traceability between Polarion and Doxygen using TraceGen.....	63
4.2.4.2	Traceability between Polarion and Enterprise Architect using TraceGen	66
4.2.4.3	Traceability between Enterprise Architect and Doxygen using TraceGen	68
4.2.5	Traceability Visualization in Power-BI	70
4.2.6	Analysis and Results	71
5	Conclusions.....	79
5.1	Open Issues and Future work	80
6	Bibliography	82
7	Appendix.....	87
7.1	Mapping of Research Papers based on Classification Scheme	87
7.2	Expert Interview Questions	92
7.3	Information Collected during Expert Interviews.....	94
7.4	Different Visual Reports used for Traceability in Power-BI.....	97

Abbreviations

ACM	Association for Computing Machinery
ALM	Application Lifecycle Management
API	Application Programming Interface
ASPICE	Automotive Software Performance Improvement And Capability dEtermination
AST	Abstract syntax tree
CLI	Command Line Interface
CRUD	Create, Read, Update, Delete
CSV	Comma-separated values
DOORS	Dynamic Object-Oriented Requirements System
GUI	Graphical User Interface
GUID	Globally Unique Identifier
HIS	Herstellerinitiative Software
HTML	HyperText Markup Language
HTTP	Hypertext Transfer Protocol
IBM	International Business Machines Corporation
IEC	International Electrotechnical Commission
IEEE	Institute of Electrical and Electronics Engineers
IIS	Internet Information Services
ISO	International Organization for Standardization
MBSE	Model Based System and Software Engineering
MDA	Model Driven Architecture
MDD	Model Driven Development
OMG	Object Management Group
QA	Quality Assurance
RM	Requirements Management
RS	Requirement Specification
ReqIF	Requirements Interchange Format
SIG	Softgoal Interdependency Graph
SMS	Systematic Mapping Study
SysML	System Modeling Language
TIM	Traceability Information Model
UML	Unified Modeling Language
URI	Uniform Resource Identifier
XHTML	Extensible HyperText Markup Language
XML	Extensible Markup Language

List of Tables

Table 1: Research Type Facet (Petersen, Feldt, Mujtaba, & Mattsson, 2008).....	12
Table 2: Contribution Type Facet (Petersen, Feldt, Mujtaba, & Mattsson, 2008).....	13
Table 3: Product Concept Context Facet.....	13
Table 4: Excerpt of mapping of research papers based on the classification schemes	14
Table 5: Categories covered by papers based on the classification	16
Table 6: Different Traceability approaches in MBSE.....	29
Table 7: Fields of TracedItem	58
Table 8: Fields of Trace Link	58

List of Figures

Figure 1: Systematic Mapping Study Process (Petersen, Feldt, Mujtaba, & Mattsson, 2008) ..	9
Figure 2: Building classification scheme (Petersen, Feldt, Mujtaba, & Mattsson, 2008).....	12
Figure 3: Number of articles published in traceability (trend considering the 30 papers used for the mapping study)	17
Figure 4: Horizontal and Vertical Traceability in V-model (Maro, 2020).....	19
Figure 5: Dimensions and directions of trace links (Winkler & von Pilgrim, 2010).....	21
Figure 6: Bidirectional Traceability and Consistency (SIG, 2017).....	28
Figure 7: Traceability in IBM (https://www.ibm.com).....	32
Figure 8: Link roles in Polarion (https://almdemo.polarion.com)	33
Figure 9: Link rules in Polarion (https://almdemo.polarion.com)	34
Figure 10: Traceability matrix in Excel	35
Figure 11: Traceability visualization filters in Yakindu (https://www.itemis.com)	36
Figure 12: Dashboard in Yakindu (https://www.itemis.com)	37
Figure 13: Snippet of expert interview questions.....	39
Figure 14 : Traceability in the toolchain (Polarion<->EnterpriseArchitect<->Doxygen)	47
Figure 15: Trace links in Polarion.....	49
Figure 16: Trace links outside Polarion	49
Figure 17: OSLC Provider and Consumer (Kaiser & Herbst, 2015)	50
Figure 18: Export of EA elements to Polarion using EAPO	52

Figure 19: ReqXchanger for Polarion and Enterprise Architect	54
Figure 20: Class diagram of the traced items	57
Figure 21: Association of TracedItem and Trace link.....	59
Figure 22: Trace link types between TracedItems	59
Figure 23: Features supported in TraceGen	60
Figure 24: Use case for the creation of trace links	62
Figure 25: Use case for the maintenance of trace links.....	63
Figure 26: Requirement diagram containing traces	73
Figure 27: Traces between requirements in Polarion.....	73
Figure 28: Trace link information in the Doxygen web report	74
Figure 29: Trace links maintenance in traceability matrix report	75
Figure 30: Trace tree report in Power-BI	76
Figure 31: Trace matrix with drill features of Power-BI	77
Figure 32: Indirect trace link establishment using Power-BI visualization	78
Figure 33: Trace-list representation in Power-BI.....	97
Figure 34: Trace-matrix representation in Power-BI	98
Figure 35: Traceability graph representation in Power-BI.....	99
Figure 36: Trace coverage report in Power-BI.....	99
Figure 37: Suspect links report in Power-BI.....	100

1 Introduction

With the enormous increase in complexity of embedded systems and reduced time to market, many companies have switched from code-based engineering to Model-based Systems and Software Engineering (MBSE) approach to develop software systems. In the MBSE approaches, systems are refined from requirements to architectural design to detailed design, all with respect to models until a refined model is available with enough details to implement the system (Galvao & Goknil, 2007). All models represent the same system but with different levels of detail. Hence, MBSE approaches tend to introduce the challenge of managing the multitude of artifacts created using various tools during the lifecycle of a system. Actively supporting traceability between different artifacts helps to understand the system better and maintain these model-based artifacts as they evolve (Winkler & von Pilgrim, 2010).

1.1 Motivation

During the life cycle of a huge, complex project, a large number of artifacts are created across the organization between different teams and in many cases across organizations. These artifacts are often developed in isolation with different tools as per the team's needs and demands. As a result, information is scattered across teams and organizations with various documents and repositories being developed and maintained by different resources (Koenigs, Beier, Figge, & Stark, 2012). If trace links are not created and maintained between various artifacts, the system's quality, integration, and maintenance will be at stake. Hence achieving traceability in a pragmatic way is very critical for the quality of the project.

Though MBSE approaches result in an enormous amount of artifacts making it hard to achieve traceability, in few cases it eases the process. For example, a few MBSE approaches develop parts of the system automatically using model transformations, and these automatic transformations from one model to the other can be used to generate traceability between them without any additional effort (Winkler & von Pilgrim, 2010). Hence following the MBSE approach during the development of a system might introduce certain benefits as well as some setbacks for achieving traceability. Hence the challenges faced for achieving traceability in MBSE projects must be compiled to find feasible solutions.

Knowing that the traceability process is not trivial but involves a lot of manpower, automation of the process is crucial. Also, the creation of trace links manually is error-prone and an arduous task. Hence tools must be used for the automatic creation and maintenance of the links. However, choosing the right tool is even more difficult in MBSE projects as they use a wide variety of authoring tools for the creation of artifacts which might result in compatibility and configurability issues resulting in tool breaks. For example, mid-way a project can change some of its authoring tools, and hence the tool used for establishing traceability must be compatible with the new tools too else it might result in tool breaks. Hence various approaches and tools available for establishing trace links must be explored along with their benefits and drawbacks to make an informed decision of choosing a tool.

Having a large number of artifacts produced in the Model-based Systems and Software Engineering approach might result in a large number of trace links between them. Visualizing all of them in a single report makes it incomprehensible and hence unusable. Therefore, pragmatic visualization techniques for the generation of trace links reports and navigation are crucial for the quality analysis of trace links. Also, having trace links between every artifact may not serve its purpose. Tracing the relationship between all the artifacts results in alleviating complexity and hampering its benefits as it reduces the understandability of the report. Hence cautious selection of subjects of interest (traceable artifacts for the creation of trace links) is also important. Hence having a traceability information model defined helps to create rules for the selection of artifacts, traceability link-types between the artifacts, and so on based on the project's needs.

Also, many standards like ASPICE and safety guideline documents such as ISO 26262 standard (Road vehicles – Functional safety), IEC 62304 (Medical Device Software Lifecycle Processes) insist on achieving traceability across software artifacts in the projects for improving the product quality. Traceability when achieved completely and correctly, helps in various activities of the project lifecycle. Many metrics can be extracted from the trace links, which helps in enhancing the project process. Hence knowing the various uses encourages having a traceability process in place in the project.

Hence based on all of the above reasons, the main focus of the thesis is to compile the challenges faced while achieving traceability in an industrial setting and provide probable

solutions. The research also focuses on compiling the most effective ways and approaches to achieve traceability. Also, a new pragmatic approach for achieving traceability is presented.

1.2 Problem Statement

This section covers the problems that are researched in this thesis. The various ways by which traceability can be achieved pragmatically in MBSE are discussed. Pragmatic traceability is establishing traceability sensibly and realistically in a way that is based on practical rather than theoretical considerations. Though the importance of traceability has been well understood by industrial companies for the success of complex projects, numerous issues make it difficult to achieve in practice. In theory, traceability when established completely, only then the project can reap its full benefits. However, practically it may not be possible to achieve complete traceability because of the various challenges. Hence the problems (P) faced by industrial companies to achieve traceability are discussed.

- **P1- Tool breaks and tool support:** In MBSE, the models are developed using a variety of tools. The tools chosen by a team may not match with other teams due to differences in expertise, management, project needs, etc. Sometimes it may also happen that the tools are changed in the mid-way of a project due to organizational decisions. Hence incompatibility and non-configurability result in tool breaks.
- **P2- Consistent trace links:** As and when requirements or any other artifacts change or evolve, the trace links must be reviewed and updated. Not maintaining trace links results in incorrect information of relationships between artifacts which does not serve the purpose and hence hinders the usability of models.
- **P3- Cost-effective:** One of the main challenges of traceability is proving its uses, as benefits cannot be measured sooner. Most of the engineers might consider traceability as an overhead task as it is time-consuming and also it does not show immediate advantages of maintaining it. Also, the time required for establishing traces and for its analysis is more when the process is not automated.
- **P4- Selecting the subjects of interest:** A huge number of artifacts are produced in the Model-based Systems and Software Engineering approach. Having trace links between every artifact may not serve its purpose. Sometimes even if it does, analyzing the huge number of trace links is complex.

As a result of these problems, most projects do not follow the process of traceability from the beginning and are created only during the assessment stage done at the middle/end of the project. This results in not deriving all the perks that could benefit the project when traceability tasks are integrated into the existing work process.

1.3 Research Approach

This section describes the research approach, namely the research objectives, research questions, and research procedure. The expectation from the research is to tackle the above-mentioned problems, to provide a clear view on benefits when traceability is closely linked with system and software development life cycle, and to be able to achieve traceability with pragmatic approaches in a toolchain. Many existing approaches are researched for the same with pros and cons. Based on the knowledge gained, a new approach along with a prototype tool will also be developed to automate the process.

The first step is to identify the research objectives and questions. The objectives and questions were identified to address the problems captured in the previous sections and also to help in researching various viable approaches.

1.3.1 Research Objectives

The main objective of the thesis is the identification of the challenges and viable approaches of tracing in industrial MBSE contexts. Even though the benefits of traceability are hard to measure, it has been time and again proven that it reduces the overall project time and improves the quality of the product. Hence identifying the existing challenges after discussing with industrial companies and compiling the reasonable ways to improve the gaps present in the existing approaches helps to achieve traceability successfully. The various ways by which the problems faced are handled by focusing on achieving only necessary traceability and accepting some of the inevitable shortcomings are researched.

1.3.2 Research Questions

The research objective leads to the following research questions (RQ):

- **RQ1:** What are the different criteria to be considered for the creation of traceability between different artifacts?

Depending on the project needs, the traceability process needs to be tailored. For example, some project needs feature level granularity and some might need package level. Similarly, the purpose of link visualization could be different from each other. Some stakeholders might need a graph to get the project progress and others might need matrix data to check for the missing links. Hence knowing different criteria is very important for the creation of a traceability process and for the identification of a tool based on the project needs.

- **RQ2:** How and when trace links are used in industrial practice?

Traceability is a necessary system characteristic as it supports software management, software evolution, and validation (Galvao & Goknil, 2007). Understanding the purpose and usage helps in guiding and choosing the appropriate tool/approach based on project needs.

- **RQ3:** What are the best practices and pitfalls present in the existing approaches/tools for the creation of trace links and their maintenance?

This aims at compiling all the challenges and relevant approaches which could be used for traceability with pros and cons. Gaps are identified and viable solutions are identified based on the knowledge of different approaches.

- **RQ4:** How does MBSE ease or complicate traceability?

Though MBSE helps to achieve high-quality software, the downside is its byproducts. Various models with different representations are created with duplicated or overlapping information between them. Hence generating traces among models at different levels of abstraction introduces complexities (Galvao & Goknil, 2007). However, MBSE has various tool supports which facilitate the creation of traces at various stages with automatic or minimal intervention. Hence understanding both the advantages and disadvantages of MBSE in achieving traceability helps to decide on the criteria and approaches to follow and to fill the gaps in existing approaches.

1.3.3 Research Method

To research the above-mentioned objective and questions, the following research method has been followed in this thesis.

The first step is to understand state-of-the-art with literature review. Conduct a literature search on traceability in MBSE and embedded software engineering projects in general. By conducting a systematic mapping study (SMS), a structure of the type of research reports and categorized results that have been published will be summarized (Petersen, Feldt, Mujtaba, & Mattsson, 2008). Visual synthesis and classification of the data research conducted based on the collected research papers help to understand the research development in the area of traceability. Besides, the coverage of the search area can be determined, the available research and results are identified in terms of quantity and type (Petersen, Feldt, Mujtaba, & Mattsson, 2008). A mapping study helps to identify, assess and interpret the set of research works to gather information on research questions raised. By performing SMS, some of the research questions will be answered to an extent.

The next step is to understand state-of-the-practice with expert interviews. It is used as a qualitative research technique to compile challenges and best practices in achieving traceability in industrial settings. This will help to gain more insight into traceability management practices, approaches, and tools used in real-time projects. The feedback from experts will be analyzed and shall be used to decide on important criteria, techniques that must be applied during the development of the prototype tool.

In the next step, challenges and best practices will be consolidated from the above two methods. Various state-of-the-art approaches/tools used for achieving traceability in MBSE will be analyzed by compiling both advantages and disadvantages. These help in noting the various aspects considered in these tools for the creation, management, and visualization techniques.

Finally using all the findings from the above-mentioned research methods, the feasibility of promising traceability approaches with a tool prototype will be explored. A toolchain for a MBSE system consisting of Polarion, Enterprise Architect, and Doxygen is selected to explore the viable approaches for achieving traceability. The prototype tool will be developed

to have trace links created pragmatically between the artifacts created in these three different tools.

All the above methods work in the direction of bridging gaps in the existing approaches to establish traceability in a project in a pragmatic way.

1.4 Thesis Structure

This section provides an overview on the organization of the remainder of the thesis. Chapter 2 gives an introduction to the basic concept of traceability and then describes the systematic mapping study conducted as part of the research. It also presents the results of the mapping study and gives an overview on the concepts of traceability compiled from the literature review. The discussion regarding the data collected from experts on traceability practices and tools in their projects, along with the compilation of best practices and pitfalls are presented in chapter 3. In chapter 4, the viable approaches for traceability are discussed. It elaborates the studies conducted during the thesis regarding various tools and approaches with their pros and cons. The feasibility of various traceability approaches across a selected toolchain is discussed (Polarion<->Enterprise Architect<->Doxygen). Finally, concluding remarks are presented along with the opportunities for further investigations in chapter 5.

2 Related Work

This chapter gives a general overview of traceability and covers the methods and findings of the literature research conducted. Chapter 2.1 describes the basics of traceability to have an understanding of the terminology. The adopted method for the literature research is described in chapter 2.2. And a summary of the SMS results is presented in chapter 2.3. It also gives a general overview of the concepts of traceability which were compiled during the literature search. This is followed by elaboration on various approaches and tools used to achieve traceability in MBSE projects in chapter 2.4.

2.1 Understanding Traceability

According to IEEE standard of software engineering terminology (IEEE, 1990), traceability is “the degree to which a relationship can be established between two or more products of the development process, especially products having a predecessor-successor or master-subordinate relationship to one another”. For example, it is the degree to which the requirements and design of a given software component match. Traceability is the ability to link artifacts created with one another using relationships between them. As the software development process involves the creation of a lot of unique intermediate artifacts apart from the final products, traceability helps in keeping track of the artifacts using the links over time. It helps in understanding the artifacts' relationships and their management as they evolve. Throughout the project, the artifacts keep getting created and modified either because of new requirements or changes in requirements, or due to some new insights. To apply these changes accordingly across all the artifacts, traceability helps. It helps in easing the analysis of the impact of changes across artifacts which in general is a very time-consuming and arduous task (Winkler & von Pilgrim, 2010). Traceability also has other benefits like program comprehension, validation of artifacts, monitoring and tracking project status, identification of reusable elements, for audits, etc which will be discussed in chapter 2.3.3.5.

2.2 Systematic Mapping Study

A systematic mapping study was conducted to get a statistical analysis of the research papers and reports found online on the research topic by categorizing them. It helps to get an overview of the research topic and how much it has been covered in the research. “A systematic mapping study provides a structure of the type of research reports and results that have been published by categorizing them” (Petersen, Feldt, Mujtaba, & Mattsson, 2008). The following shows the process followed in the study:

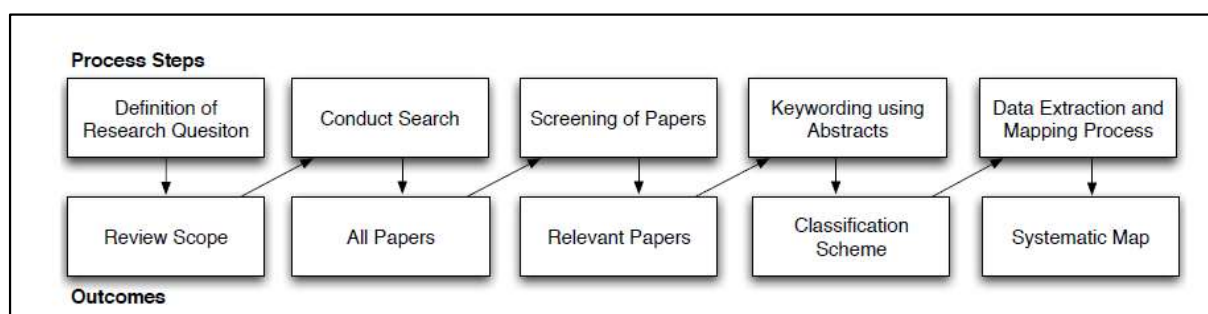


Figure 1: Systematic Mapping Study Process (Petersen, Feldt, Mujtaba, & Mattsson, 2008)

The scope of the research is defined by having research questions for the SMS (chapter 2.2.1) through which the first search for papers was conducted. The research questions for SMS are to be understood as sub-questions of the research questions listed in chapter 1.3.2, which were derived from the problem statements and objective of the thesis. Using the knowledge of the first screening of research papers, search strings were defined in chapter 2.2.2, and a refined search was conducted. Before looking into the papers to check their relevance for the research work, the inclusion and exclusion criteria were defined in chapter 2.2.3. Based on the findings, the papers were classified into the context, research, and contribution facet. The final step is the data extraction and the actual mapping process, which results in a systematic map (Petersen, Feldt, Mujtaba, & Mattsson, 2008).

2.2.1 Research Questions

The first research question is regarding the criteria to be considered for the creation of the traceability between artifacts. As criteria depend on the project’s needs, knowing the different purposes of traceability, various ways of creating and maintaining traces, etc will help in deciding the criteria. Hence research materials focusing on various aspects of traceability

must be checked. Hence systematic mapping questions should break down the research questions to be more precise if research questions are not straightforward. Below is the mapping between research questions and respective systematic mapping questions:

RQ1: What are the different criteria to be considered for the creation of traceability between different artifacts?

SMS_Q1: What different areas in traceability are addressed and how many articles cover each area?

RQ2: How and when the trace links are used in industrial practice?

SMS_Q2: What are the main objectives of traceability and are any kinds of evidence presented concerning how traceability helps in project improvement in the industrial setup?

RQ3: What are the best practices and pitfalls present in the existing approaches/tools for the creation of trace links and their maintenance?

SMS_Q3: What are the most investigated challenging topics in traceability and what are the proposed solutions for the same?

RQ4: How does MBSE ease or complicate traceability?

SMS_Q4: What are the available tools/approaches used to achieve traceability in MBSE projects with pros and cons?

Based on the above questions, the scope of the systematic mapping study is fixed. The first question defines the basis of this mapping study and provides an overview on the topics of traceability that need to be checked before deciding on the process and approaches of traceability in a project. The second question gives an overview of the purpose and importance of the application of traceability at the right time in real-life projects. The next question answers the most challenging aspects of traceability and how they are tackled in the existing approaches/tools. The fourth question mainly compiles the various approaches/tools used to capture trace links in MBSE projects and also on the maturity level of the approaches.

2.2.2 Search Strings Used for Online Database Research

The identified keywords along with Boolean logic to either combine alternative terms with an “OR” or connect terms with an “AND” are specified. The resulted search strings are:

- Traceability OR trace link OR tracing) AND (survey OR overview OR "literatur* review")
- (Traceability OR trace link OR tracing) AND ((challenges OR best practice OR lessons learned) AND (industry or industrial))
- (Traceability OR trace link OR tracing) AND (automotive OR health OR MBSE)

Google Scholar database is mainly used for searches. However, ACM, IEEE, and ScienceDirect were also used for cross-checking any missing articles.

2.2.3 Inclusion and Exclusion Criteria for Search Results

Before screening the research papers obtained through the above search string, below inclusion and exclusion criteria are defined.

Inclusion criteria are noted to ensure that only relevant and useful results are put forth.

IC1: Papers matching the search string and papers describing the approaches within the scope of this thesis.

Papers were considered irrelevant if they meet at least one of the following exclusion criteria:

EC1: Papers not focusing on traceability for software engineering, i.e. Tracing in supply chain activities, Medical tracing, etc

EC2: Papers not in English

EC3: Papers not accessible through one of the stated databases (IEEE, ACM and Science Direct)

2.2.4 Classification Scheme

This section describes the followed approach of Bailey (Bailey, et al., 2007) and steps mentioned in Petersen (Petersen, Feldt, Mujtaba, & Mattsson, 2008) for classifying the research papers obtained after applying inclusion and exclusion criteria. Figure 2 shows the steps followed for classification which eventually helps for the mapping. To create the classification scheme, the abstracts of the research papers were read through and keywords were identified. Also, the context of the paper is noted. All keywords were put together and a higher-level view of the research, its nature, and its contribution was obtained. This helps in defining the different categories in which the papers fall. When abstract information doesn't

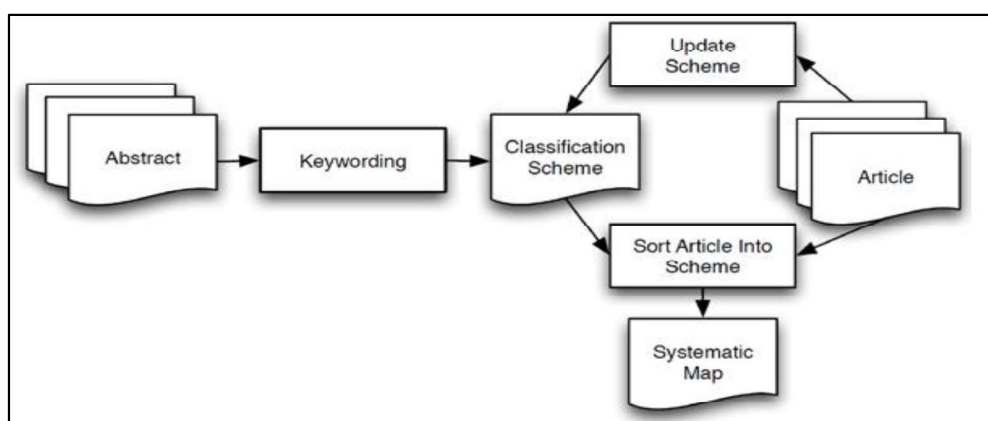


Figure 2: Building classification scheme (Petersen, Feldt, Mujtaba, & Mattsson, 2008)

contain enough information, the introduction and conclusion were also reviewed. This knowledge helps in classifying the research papers into different categories (Petersen, Feldt, Mujtaba, & Mattsson, 2008).

Research papers were classified into the following three facets: The research facet, the contribution facet, and the product concept context facet based on the research paper's content.

The research facet provides the different categories of research types that can be conducted. Below Table 1 describes the different kinds of research techniques as per (Petersen, Feldt, Mujtaba, & Mattsson, 2008).

Table 1: Research Type Facet (Petersen, Feldt, Mujtaba, & Mattsson, 2008)

Category	Description
Validation Research	Techniques investigated are novel and have not yet been implemented in practice. Techniques used are for example experiments, i.e., work done in the lab.
Evaluation Research	Techniques are implemented in practice and an evaluation of the technique is conducted. That means, it is shown how the technique is implemented in practice (solution implementation) and what are the consequences of the implementation in terms of benefits and drawbacks (implementation evaluation). This also includes identifying problems in the industry.
Solution Proposal	A solution for a problem is proposed, the solution can be either novel or a significant extension of an existing technique. The potential benefits and the applicability of the solution are shown by a small example or a good line of argumentation.
Philosophical Papers	These papers sketch a new way of looking at existing things by structuring the field in form of taxonomy or conceptual framework.
Opinion Papers	These papers express the personal opinion of somebody whether a certain technique is good or bad, or how things should be done. They do not rely on

	related work and research methodologies.
Experience Papers	Experience papers explain what and how something has been done in practice. It has to be the personal experience of the author.

Each of the papers researched was classified into the above categories based on the classification without digging deep into the papers. Also, the papers were classified based on their contributions as in Table 2 as per (Petersen, Feldt, Mujtaba, & Mattsson, 2008).

Table 2: Contribution Type Facet (Petersen, Feldt, Mujtaba, & Mattsson, 2008)

Contribution Category	Description
Metric	Papers cover mainly the standards of measurement
Tool	Papers talks about the tools that could be used to achieve the expected results
Model	Papers talk about different example systems that can be used along with various criteria to be considered.
Method	Explains different approaches of how something can be achieved
Process	Explains the different steps to be followed
Terminology	Explains mainly the concepts

Below Table 3 shows the different types of product concept context facets that were classified based on the main topic of research i.e., traceability. Hence, each of the papers was also classified based on the concept described and covered.

Table 3: Product Concept Context Facet

Product Concept Context Facet	Description
Traceability concepts	Papers cover basic concepts of traceability along with its purpose
Creation of traceability	Papers describe an approach, tool, or steps to create trace links
Traceability maintenance	Papers mainly talk about how traceability must be maintained with an approach or tool
Requirements traceability	Papers cover traceability for requirements only
End to end traceability	Papers describe steps, approaches to apply traceability to all the artifacts from requirements to test reports
Challenges of traceability and solutions	Papers cover mainly the issues faced for achieving traceability and probable solutions
Comparison of tools and techniques	Papers describe various tools and approaches followed along with pros and cons
Traceability visualization	Papers describe the various traceability reports and navigation approaches useful for the analysis

2.3 Results

In this section, the result of the performed mapping study is presented. First, a brief overview of the results of the search is described in chapter 2.3.1. The answers to the mapping study research questions are explained in detail in the next chapter 2.3.2. Basic traceability concepts along with tools and approaches found during literature search are described in chapter 2.3.3.

2.3.1 Results of Database Search

In this section, the result of the mapping study based on the classification scheme described in chapter 2.2.4 is presented. The defined keywords led to a total amount of 714 research papers. The search was mainly done in Google Scholar and other sites like ACM, IEEE, and ScienceDirect were rechecked for any missing relevant papers.

From these papers, 684 papers were excluded due to the inclusion and exclusion criteria (chapter 2.2.3) and also by removing the duplicate copies. Hence, 30 papers have been considered to be relevant, and thus, added to the list of primary studies. The following Table 4 shows an excerpt of the list of primary studies that are mapped based on the classification schemes. The classification of all the considered papers can be referred to in Appendix 7.1.

Table 4: Excerpt of mapping of research papers based on the classification schemes

ID	Year	Title	Search String	Author	Research facet	Contribution facet	Product Concept Context Facet
P01	2007	Survey of Traceability Approaches in Model-Driven Engineering	(Traceability OR trace link OR tracing) AND (survey OR overview OR "literatur* review")	(Galvano & Goknil, 2007)	Solution proposal	Method	End to end traceability/ Comparison of tools and techniques
P02	2009	A survey of traceability in requirements engineering and model-driven development	(Traceability OR trace link OR tracing) AND (survey OR overview OR "literatur* review")	(Winkler & von Pilgrim, 2010)	Solution proposal	Method/ Model/Terminology	Traceability concepts

P03	2006	Traceability Techniques: A Critical Study	(Traceability OR trace link OR tracing) AND (survey OR overview OR "literatur* review")	(Bashir & Qadir, 2006)	Evaluation Research	Method	Requirements traceability/ Comparison of tools and techniques
-----	------	---	---	------------------------	---------------------	--------	---

From the mapping information, the frequencies of research papers based on the classification were analyzed. The following summarizes papers on the research facet. The majority of papers (around 67%) talked about approaches that can be followed for achieving traceability and hence can be described as solution proposals. The papers talked about the challenges faced and also described the probable solutions that can be followed to tackle the problems. Apart from that, around 20% of the papers described the current techniques that are being followed in industrial scenarios with pros and cons, and 10% of papers investigated on experiments that are still novel and not used in real-time projects.

As mentioned in the previous facet, even in the contributions type more than half (around 70%) of the selected papers can be categorized to type “method” because papers talked about various approaches that can be followed to achieve traceability. Also, most of these papers explored the topic of the main criteria to be followed and major points to be remembered while applying traceability in a project. Around 37% of the papers described different model systems that can be used to create and maintain trace links and also talked about the main criteria to consider while using them. Few papers (28%) explained the different tools used along with their advantages and disadvantages. It was also noted that many of the papers covered more than one category resulting in an overlap. Hence overall, researching these papers gave us a fair idea of the available methods, models, and tools available for traceability.

The next facet mainly describes the division based on the various topics of traceability that the paper discusses. The papers were widespread in terms of the topics covered. Majorly papers described end-to-end traceability concepts (27%), however, few papers also talked particularly about traceability in requirements (17%). Various tools and methods used for traceability were compared with pros and cons. The challenges faced while tracing was also one of the main topics of discussion. In some papers, traceability maintenance was considered as the major topic and hence approaches that can be followed for the same were explained.

Also even in this facet, many of the papers covered more than one categories and hence overlapping of papers covering more than one category were observed. Below, Table 5 shows the division of papers based on the topics covered and based on the frequencies. As mentioned earlier, the number includes overlapping of the papers covering different topics. i.e. Same paper could cover both the Model and Tool category in the contribution facet.

Table 5: Categories covered by papers based on the classification

Classification Scheme	Categories	Number of articles
Research facet	Solution Proposal	20
	Evaluation Research	6
	Validation Research	3
	Philosophical Papers	1
Contribution facet	Method	21
	Model	11
	Tool	8
	Process	2
	Terminology	2
Product Concept Context facet	End to end traceability	8
	Comparison of tools and techniques	7
	Challenges of traceability and solutions	6
	Creation of traceability	6
	Requirements traceability	5
	Traceability maintenance	5
	Traceability concepts	3
	Traceability visualization	1

The next numerical data shows the trend of research in the field of traceability in software engineering. Below Figure 3 shows the trend of the research.

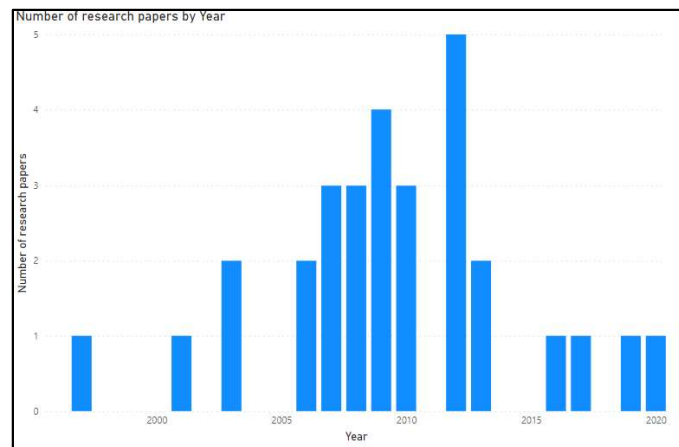


Figure 3: Number of articles published in traceability (trend considering the 30 papers used for the mapping study)

It has been noticed that around the year 2010, more researches on traceability were conducted and it was highest in the year 2012, considering the papers that have been examined for this systematic mapping study.

2.3.2 Answering Mapping Study Research Questions

The main goal of mapping study apart from building a classification scheme and structuring the types of research by providing a visual summary is to combine the results to try to find the answers to the research questions. Hence, the answers found for the mapping questions during the mapping study are discussed in this section. As systematic mapping questions are mapped to research questions, these answers give initial basic information for the research though not detailed.

SMS_Q1: What different areas in traceability are addressed and how many articles cover different areas?

This is answered by the Product Concept Context Facet classification mentioned in the previous chapter. The majority of them gave importance to tracing from requirements till the testing phase. i.e. end-to-end traceability. The papers also discussed various approaches that can be followed along with the challenges. Hence, things to consider before selecting an approach were highlighted. Considerable papers gave importance equally to the traceability creation and maintenance phase. However, the visualization and navigation aspects were not explored more.

SMS_Q2: What are the main objectives of traceability and are any kinds of evidence presented with respect to how traceability helps in project improvement in the industrial setup?

Almost all the papers talked about the uses and main purpose of traceability in the context in which it was described. However, it was observed that none of them provided evidence of how traceability helps in project improvement as measuring it is almost impossible. As the traceability process and its uses are widespread across the project's life cycle, it was understood that it was overall beneficial for the project. Also, less than 30% of the papers covered traceability considering the industrial setup.

SMS_Q3: What are the most investigated challenging topics in traceability and how have these changed/improved over time?

Around 50% of the papers talked either directly about the challenges faced and their probable solutions or indirectly discussing the problems faced in various tools and approaches and comparing them for the pros and cons. Hence a fair share of the major challenges and advised solutions were compiled by checking the abstract and conclusion sections of the papers.

SMS_Q4: What are the available tools/approaches used to achieve traceability in MBSE projects with pros and cons?

In the research papers, many MBSE traceability approaches like modeling approaches, transformation approaches, requirements-driven approaches, information-retrieval techniques, etc were discussed. It covered around 60% of the papers. Apart from these approaches, around 27% of the papers also discussed the tools developed. Some of the tools were novel too. However, almost all of them covered the benefits and drawbacks of the approaches and tools in the discussion. Hence the mapping study provided an overview of the tools used for traceability in general.

As the research questions of the mapping study were derived from the research questions of this thesis, some of them got the basic information regarding the subject of interest at this point.

2.3.3 Concepts of Traceability

During the systematic mapping study, research papers were screened to check their relevance for the research work. Hence, the main concepts of traceability and the various approaches/tools used were also compiled for future reference while screening the papers. These collected data were used during expert interviews, to identify the research gaps and for the development of a prototype tool. Hence, the compiled main concepts of traceability and various approaches/tools used for achieving traceability are described in this section.

2.3.3.1 Classification of Traceability

In typical V-model process lifecycle projects, traceability can be divided majorly into horizontal and vertical traceability (Ramesh & Edwards, 1993).

Horizontal traceability: In general, horizontal traceability is the ability to link between the same abstraction levels. For example, links are created between different requirements. In terms of the V-model, horizontal traceability is the ability to link between artifacts on the left side of the V-model to their corresponding verification artifacts present on the right side of the model (Maro, 2020). For example in Figure 4, System Requirements are linked with corresponding System Tests. Similarly System Architecture elements are linked with System Integration Tests etc.

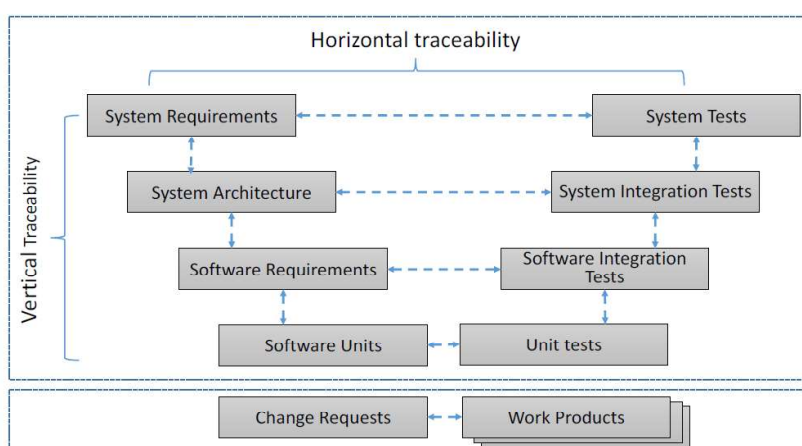


Figure 4: Horizontal and Vertical Traceability in V-model (Maro, 2020)

Vertical traceability: Vertical traceability is the ability to link between different abstraction levels. For example, links are created between requirements to architecture, architecture to design elements, etc. For example in Figure 4, vertical trace links are established between System Requirements and System Architecture. Similarly between Software Requirements and Software Units etc.

The other classification is with respect to requirements like pre-requirements specification (pre-RS) traceability and post-requirements specification (post-RS) traceability.

Pre-RS traceability: The links created during elicitation, discussion, and agreement of requirements with customers until they are included in the requirements specification document. This includes dealing with informal, conflicting, or overlapping information. Maintaining these links provides a rationale for every following artifact that will be developed.

Post-RS traceability: The links created during the stepwise implementation of the requirements in the design and coding phases (Winkler & von Pilgrim, 2010). Post-RS traceability is more important to maintain as the links are created to maintain consistency across artifacts with respect to the agreed requirements from the customer.

The other kind of classification is based on the directionality of trace links as per IEEE Std. 830-1984. It has introduced the terms backward traceability and forward traceability (IEEE, 1984).

Backward traceability: refers to the ability to follow the traceability link from a specific artifact to the sources from which it has been derived. i.e., from a successor to a predecessor

Forward traceability: refers to the ability to follow the traceability links to the artifacts that have been derived from the artifact under construction. i.e., from predecessor to successor

Both horizontal and vertical traceability can be **unidirectional** or **bidirectional**. Bidirectional traceability is the ability to trace from a predecessor to a successor or from a successor to a predecessor. Usually, many standards expect bidirectional traceability as it's important to be able to trace from one artifact to the other and back again.

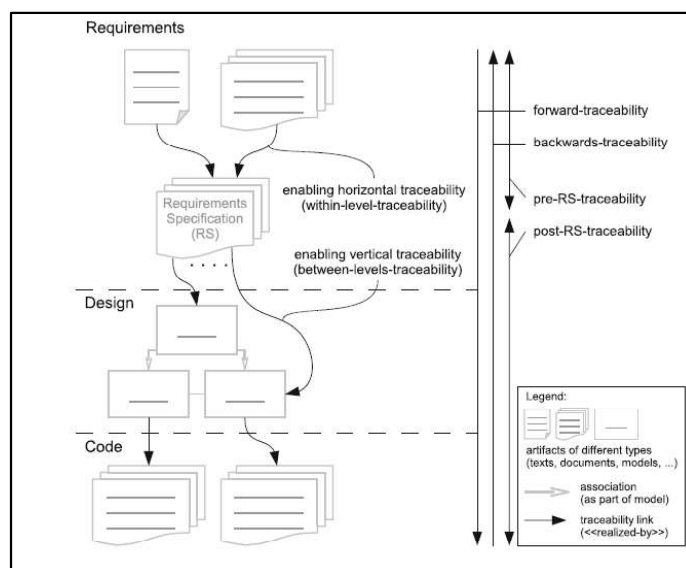


Figure 5: Dimensions and directions of trace links (Winkler & von Pilgrim, 2010)

The above classifications are shown in Figure 5. The Pre-RS traceability is from elicitation till the creation of Requirement Specification and Post-RS traceability starts after the creation of Requirement Specification. Also, the direction of the arrows in the figure represents whether links are created from predecessor to successor or vice-a-versa.

2.3.3.2 Basic Terminologies of Traceability

The other basic terminologies used in traceability are as follows. A trace encompasses trace artifacts as well as a trace link associating the trace artifacts. And traceability information model (TIM) defines the permissible trace artifact types and the permissible trace link types, often utilizing the traceability meta-model (Holtmann, Steghöfer, Rath, & Schmelter, 2020).

Types of trace links:

Trace links can be classified into implicit and explicit trace links based on the specificity of links. Implicit links are not established directly instead links are formed because of naming conventions used in artifacts. Explicit links are created using tool support. Based on the storage of links, trace links are classified as volatile and persisted. Volatile links are not stored and are used only until an operation using them is completed. Whereas persisted links are stored for later retrieval. They are further classified as internal, trace links that are stored within one of the traced artifacts, and external, trace links that are stored separately outside

the traced artifacts. Based on the creation process, they are further classified as manual and computed trace links. As the name suggests, they are either manually created or generated using a tool (Holtmann, Steghöfer, Rath, & Schmelter, 2020).

2.3.3.3 Prerequisites for Achieving Traceability

Creating traceability without knowing the background of why it is required or how it is used etc may not yield the best results. Hence knowing the background is very important. Below are some of the core questions to be asked as a pre-requisite for achieving traceability in any project. Discussing the below questions also answers part of the research questions RQ1 and RQ2.

- **What?** - This question mainly answers the scope (boundary) of traceability. The specific questions to ponder upon are: What are the subjects of interest for the creation of trace links? What kind of inter-relationships between artifacts are to be considered based on their later usage in the project context? This helps a project to decide on the traceability scope before starting the process and hence can be planned accordingly. For example, some safety-related projects might need a rationale for the decisions that are taken for non-functional requirements. Hence in such cases, trace links must be created and maintained considering the non-functional requirements and their corresponding architecture and design elements. However, most of the projects mainly are interested in covering functional requirements traceability as it helps in change impact analysis and change propagation. Hence as per the needs of the project, the scope for traceability must be decided beforehand for informed planning.
- **Why?** - Knowing the purpose helps not to consider the traceability process lightly. Hence the questions to be asked are: Why are the trace links being created? Before deciding on achieving traceability, the main purpose of traceability must be understood as in most cases the benefits of traceability are not immediate. Hence making engineers aware of its purpose will help in achieving high-quality traceability. The purpose could vary depending on the needs of the projects and on the usage context of the engineers. For example, some might require it for proving system adequateness to their customers, some might need it for monitoring progress, and for

some to understand the system. Hence understanding why traceability is required is considered as one of the prerequisites in achieving traceability.

- **How?** - Deciding on tools is critical to avoid manual work. Hence to consider various aspects of tools, the following questions must be discussed: How can we achieve traceability? Achieving traceability is not trivial because of the sheer number of artifacts developed in a project. Creating and maintaining traceability manually is not only error-prone, it is a too laborious task as well. Hence deciding on the tools to create and maintain traceability is critical. The decision of choosing traceability tools depends on various aspects. Such as tools used to create different artifacts, cost-effectiveness, traceability tool features for navigation, report generation, and visualization, etc. Hence answering the question “how” helps in choosing the right tool.
- **When?** - Defining the process beforehand and making the steps mandatory helps to achieve traceability on time. Hence to decide on the process, the following questions are to be discussed: When should the links be created and updated? When should the links be reviewed? Having a process in place is very important to achieve traceability. Creation and its maintenance must be part of the software development life cycle and the task of creating it and maintaining it must be made mandatory. As not doing it might not create any problem in the short run, many ignore the long-run benefits of doing it. Hence imposing the activity of maintaining traceability while creating and updating artifacts is crucial for its quality.
- **Who?** - Defining the process of who creates is also a very important step to achieve traceability for the same reasons mentioned above. Hence the responsible person for the creation and maintenance of trace links must be decided during the planning phase. As the person creating it may not get immediate results, the task is considered as overhead. Hence deciding on resources for the creation and maintenance of trace links and allocating sufficient time is as important as the creation of artifacts itself. Links must be carefully thought of and hence must not be considered as a low priority task. Management must ensure to give enough time allocated for this activity to reap benefits in the long run.

2.3.3.4 Key Criteria for Traceability Approaches

As discussed having traceability in a project results in long-term benefits and also is used to measure the quality of the software system. Hence achieving traceability must be regarded as a crucial part of the project. The success in achieving traceability is mainly dependent on the intrinsic motivation from all the stakeholders and trust in the process (Koenigs, Beier, Figge, & Stark, 2012). Hence the approaches/tools to be used for traceability must be carefully thought of. Below are the few main criteria to be considered while deciding on the approach. This also provides us the detailed answers for the research question RQ1 (What are the different criteria to be considered for the creation of traceability between different artifacts?)

Planning criteria: During the planning phase of traceability, the main aspects to be considered are the traceability schema, link properties, and supported artifacts.

Traceability schema: In order to systematically approach traceability, a set of rules on it must be decided early on during the planning phase. Traceability schema acts as a rulebook to follow for all the stakeholders involved in its creation and maintenance. The schema should give information on traceability type set which defines the types of links and their meaning for a project, traceability object set which determines the types of objects that can be linked with a certain type of links, a traceability role set which defines the stakeholders and their permission to access traces, a minimal links set that determines which links can exist so that the traceability information is regarded as correct and complete for a specific project. And also, a schema specification must include a metrics set that defines quality measures for traceability in the respective project (Winkler & von Pilgrim, 2010).

Traceability link properties: Trace links can be created with various levels of detail. Hence properties of the links must be decided based on the project needs. Following are some of the properties and their probable values that must be considered while making the decision. Traceability link types could be invasive or non-invasive. In invasive types, artifacts are modified to obtain the trace links however in non-invasive, no drastic changes are made to the artifacts and are minimally intrusive (Antonino, Keuler, Germann, & Cronauer, 2014). The variable arity of link types could be binary or n-ary where an artifact can have multiple sources and multiple targets (Maletic, Munson, Marcus, & Nguyen, 2003). Traceability link direction can be uni-directional or bi-directional (Winkler & von Pilgrim, 2010). Traceability link hierarchy could be non-hierarchical or poly-hierarchical. Traceability link granularity

decides on the level of details that the links capture. For example, a link could be parameter level tracing(fine-grained), feature level tracing, or model/package level tracing(coarse-grained) (Koenigs, Beier, Figge, & Stark, 2012).

Traceability recording criteria: This helps in the selection of the creation process of trace links. The decision on tool types is one of the main criteria which have to be considered based on the project needs. Various kinds of tools are available for the creation and maintenance of trace links which are discussed in chapter 2.4. The other criteria to be considered for recording trace links is the degree of automation, the option could be manual, semi-automatic, or full-automatic based on the budget and tools chosen. Interoperability of tools must also be considered as per projects needs to check for the supported formats, supported standards, supported tool's artifacts, and multi-user tool support.

Usage criteria: The following category addresses the aspired uses of trace links and hence tools should be selected based on needs. When used for change handling, the tool usually provides different possible ways of informing users about changes in artifacts. Change notification like suspect links provides useful information regarding changes based on the artifacts involved in the link and change propagation triggers a predefined action on affected artifacts. Traceability could also be used for verification purposes. In that case, a tool providing visualization in terms of graphs and matrices along with filters fit the purpose. When used for monitoring purposes, traceability tools must have the ability to create progress reports. Many other usages will be discussed in chapter 2.3.3.5. Hence purpose is considered as one of the main criteria to decide on the traceability approach (Koenigs, Beier, Figge, & Stark, 2012).

2.3.3.5 Uses of Traceability

As discussed above, usage is one of the main criteria that need to be considered. Hence in this chapter, various uses and applicability of trace links are discussed.

Change management: Changes in complex systems are inevitable and bear a huge risk. Changes may be caused by changing user requirements and business goals or may be induced by changes in implementation technologies. There is a need to analyze the impact of requirement changes to determine possible conflicts and design alternatives influenced by

these changes (Amar, Leblanc, & Coulette, 2008). As interdependencies between artifacts are well documented and maintained by traceability, when a change request comes along, the proposed change propagation across the artifacts can be easily identified. Hence traceability helps in managing the change requests from customers systematically.

Impact analysis: Before making any changes to the systems, the decision must be well thought of as changes lead to risks. Traceability helps in generating an impact report by identifying all the dependent artifacts. This report helps in the analysis of the impact and decision-making (Amar, Leblanc, & Coulette, 2008).

Coverage analysis: Coverage analysis helps in checking whether every requirement has a corresponding design, implementation, test record, etc. The same can be analyzed for every artifact by tracing forward or backward. The coverage analysis report also helps in proving to customers the system's adequateness.

Reuse of product components: Traceability helps in the reuse of components because the maintained trace links from the requirements to the corresponding design, implementation, test reports, etc for that component can be easily obtained. If explicit trace links are not present, then analysis needs to be done again as information may not be readily available which calls for extra time and effort (Lago, Muccini, & Van Vliet, 2009). Reusing of components helps particularly in product line engineering where a new variant of the product is created by reusing.

Support for audits: In audits, trace links can be used to ensure that a downstream artifact satisfies the upstream specification. It also helps in special audits like the system being examined for security audit, where traces help for the identification of critical elements across the system (Winkler & von Pilgrim, 2010).

For monitoring progress: As traceability maintains links across the system between various artifacts, monitoring the status of each requirement till its implementation and testing status is easy. Hence generating reports from trace links can be used for monitoring the status (Winkler & von Pilgrim, 2010).

For easy understanding: Since traceability maintains trace links across artifacts, it can be used for gathering different level's corresponding information of the system. Information with

links helps for better comprehension. In most cases, resources might join the project in mid-way and might find it difficult to gain overall knowledge of the project. Traceability when maintained across artifacts, it helps to connect various artifacts. Hence traceability helps in knowledge engineering.

For tracking rationale: Traceability helps in understanding the decisions made which is critical during the maintenance of the project. Maintenance mainly relies on up-to-date documentation. Trace links help to link back to track the decisions made for a particular component or module. For example, in which design modules a given requirement is realized (Lago, Muccini, & Van Vliet, 2009).

The above usage scenarios provide a fair idea on the research question RQ2. (How and when the trace links are used in industrial practice?)

2.3.4 Traceability in Automotive Domain

In the automotive domain, the development of all safety-critical systems has to comply with safety standards such as ISO 26262. These standards require the establishment of traceability, the ability to relate artifacts created during the development of a system, to ensure resulting systems are well-tested and therefore safe (Maro, 2020). Also, traceability is a key aspect and a necessary prerequisite for successful Automotive SPICE evaluation. Automotive SPICE is a quality standard that allows automotive manufacturers to assess the performance of suppliers' engineering processes.

According to ASPICE, bidirectional traceability must be achieved in both horizontal and vertical abstraction levels. In the below Figure 6, the relationships demanded by ASPICE between artifacts in a V-model project are shown. The line with two-sided arrows represents the expected bidirectional traceability between the two artifacts. According to ASPICE, bidirectional traceability must be achieved between requirements, architectural elements, software design, and software units on the left side of the V-model and between the work products on the left side to the corresponding test cases and test reports on the right side of the V model. Hence, both vertical and horizontal traceability is expected.

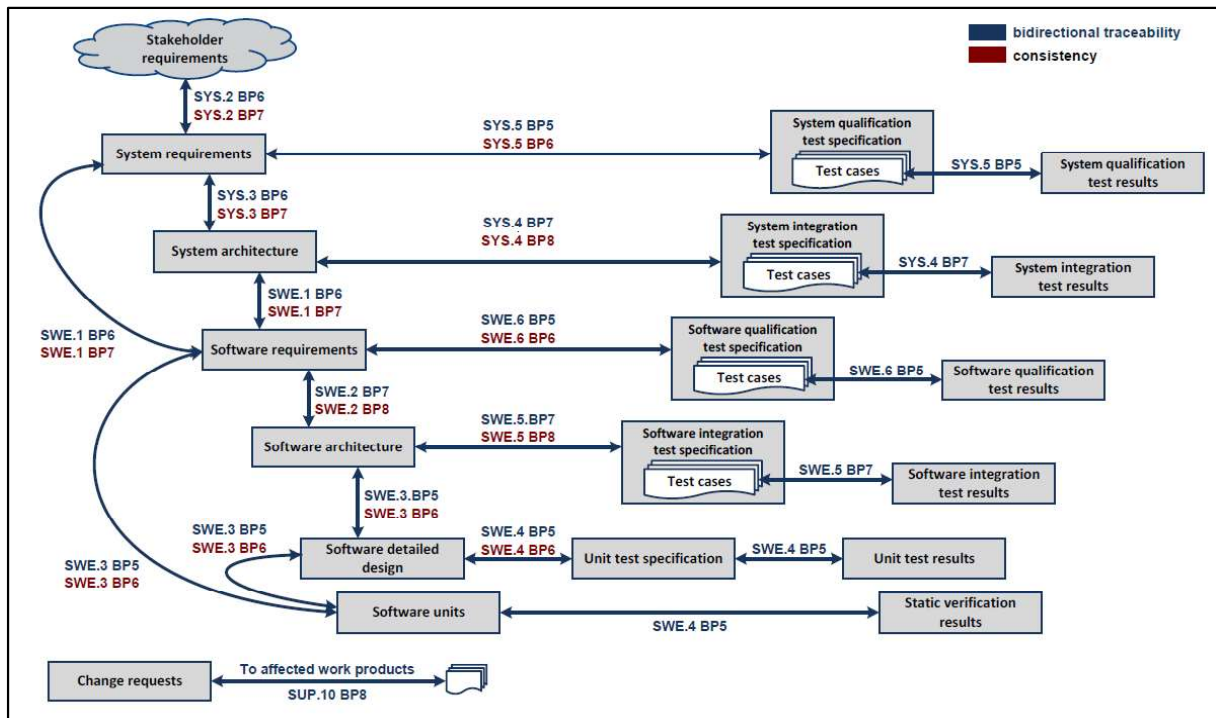


Figure 6: Bidirectional Traceability and Consistency (SIG, 2017)

2.3.5 Traceability in Model-Based Systems and Software Engineering

In MBSE, software development is done by implementing abstractions of the system with varied details and precision. The software systems are developed into subsystems and components and then these models are automatically transformed into source code (Gomaa & Hussein, 2007). Having traceability between models helps in understandability, maintainability, and adaptability. Traceability between the models can be achieved during transformations from one model to the other as a byproduct. Having traceability information between models helps in the evolution of the system. The different versions of the system with reusable components/models could be developed with much less effort if trace links between models are well maintained. However as mentioned earlier, the number of models generated will be more, and hence having trace links between every model may not be understandable and maintainable. Also, as information overlaps between models with varied details based on the stakeholders' viewpoint, generating trace links between every model may not be optimal because of overlapped information. Hence, through the years many different approaches have been developed and tried for achieving traceability successfully in MBSE systems. However, not all approaches are suited for every system. Every approach is unique

and it mainly depends on the usage criteria. Hence approach has to be selected based on the needs of the project.

Below Table 6 describes the various approaches followed to achieve traceability in projects based on MBSE with pros and cons.

Table 6: Different Traceability approaches in MBSE

Approach	Summary	Advantages	Disadvantages
Rule-based approach (Mader, Gotel, & Philippow, 2009)	This approach defines a set of rules based on the attributes of the artifacts. It automatically generates traceability links between artifacts based on the rules. The first kind of rule, Requirement-to-object-model rules, and a technique based on information retrieval are used to automatically establish traceability relations between requirements and analysis models. The second kind of rule analyzes the relations between requirements and object models to recognize intra-requirements dependencies and establishes these relations automatically.	The advantage of the approach is that it can be used with all the artifacts such as requirements, use cases, object models, code, etc.	The disadvantage is structural changes are hard to identify.
Hypertext approach (Maletic, Munson, Marcus, & Nguyen, 2003)	Manage traceability with XML markup specifications. The approach uses an XML-based representation of both the models and the links. Models are represented in XML with no restrictions as to the content, organization, or schema. This allows for full interoperability and flexibility of models in the approach including document-oriented models, e.g., DocBook, XHTML, etc., and data-oriented models, e.g., UML, ASTs, etc.	Works well with code and requirements.	Weekly supports the other types of artifacts.
Event-based approach (Mader, Gotel, & Philippow, 2009)	Manages traceability using publish-subscribe links and event-based subscriptions. Requirements and other artifacts are linked through a publish-subscribe relationship stored in a central database. Created events are published to an event server that sends notifications to subscribers of the changed requirement. These notifications	Maintains dynamic links	Scalability issues in maintaining the dynamicity of the traceability.

	contain detailed information about a change to facilitate the manual update process of the subscribing artifacts.		
Constraint-based approach (Gates & Mondragon, 2002)	The model provides support for establishing linkages between constraints and artifacts. Constraints are used to automatically define links. Because constraints are automatically inserted into the source code, it is possible to automate the tracking of requirements to the source code that is being monitored. The approach allows tracing of areas where constraints are checked and conflicts may occur.	Views artifact types as constraints among them.	Difficult to refer to all the traceability links with the constraints.
Transformations-based approach (Grammel & Kastenholz, 2010)	In implicit traceability link generation, no additional effort is necessary to obtain trace links between input and output models, as they are generated automatically in parallel to the actual model transformation. In explicit trace link generation, traceability is a regular output model of the transformation and incorporates additional transformation rules to generate it. Uses incremental and graph transformations-based methodologies for trace links.	Suited for model-based software systems.	Difficult to apply to artifacts that are not generated using MDD.
Goal-centric approach (Cleland-Huang, Hayes, & Domel, 2009)	Models non-functional requirements and their dependencies using a SIG. A probabilistic network model is then used to dynamically retrieve links between classes affected by a functional change and elements within the SIG. Manages the change impact of non-functional requirements.	Ensure quality by assessing the change impact of functional v/s non-functional aspects.	Lack of scalability and tool support.
Model-based approach (Badreddin, Sturm, & Lethbridge, 2014)	Manages traceability using template-based models. Model-oriented methodologies, which adopt the Model Driven Architecture (MDA) or similar approaches, place the focus on models, rather than code. Such models are typically more concerned with system entities (e.g., classes) and behavior (e.g., state machine) than the system's	Supports different types of artifacts.	Lack of support towards non-MDD.

	goals, use cases, and non-functional properties. All artifacts are modeled and trace links are effectively managed between them.		
Scenario-driven approach (Egyed, 2001)	The approach first creates trace information between the running system and scenarios followed by comparing those traces with hypothesized traces. The approach then generates new trace information and validates existing ones.	The approach works once an executable or simulatable software system becomes available which may not necessarily be the final release of a system.	The approach relies on monitoring tools for spying into software systems during their execution or simulation. (e.g., Coverage tool)

2.4 State of the Practice

This section provides an overview of the available tools used for establishing traceability in MBSE projects. For each of the below approaches uses a tool example will be considered for the explanation for better understanding.

2.4.1 Traceability in Requirements Management Tools

Maintaining requirements in Excel spreadsheets or a Word document would be very cumbersome as the project evolves. Hence, for maintaining and visualizing the requirements and for collaboration between users (like reviews, approvals, inputs, etc), requirements management (RM) tools are used. Some of these tools also provide traceability features like the creation of trace links between requirements and also between requirements and other artifacts. They are used for various purposes like impact analysis and for running test cases based on the impact after any changes in the system. Some of the popular RM tools with traceability features are Jama software, IBM Engineering Requirements Management DOORS Next, Visure, and Modern requirements. Traceability in IBM Engineering Requirements Management DOORS Next is considered as an example tool to list down the features. Similar features will be present across RM tools for traceability.

Following are some of the key features of the tool to help in the creation and maintenance of trace links in a project:

- The creation of trace links is simple using the drag and drop feature

- Grid and tree views are available for the analysis of the relationships between various artifacts. In below Figure 7, the tree view shows the hierarchical view of the links. The immediate link to the artifact with ID 152 is artifact 78. However, when expanded, the other links to artifact 78 are also shown which are indirect links to 152 as well. Hence the user can decide the level of information that is required and choose the view accordingly.

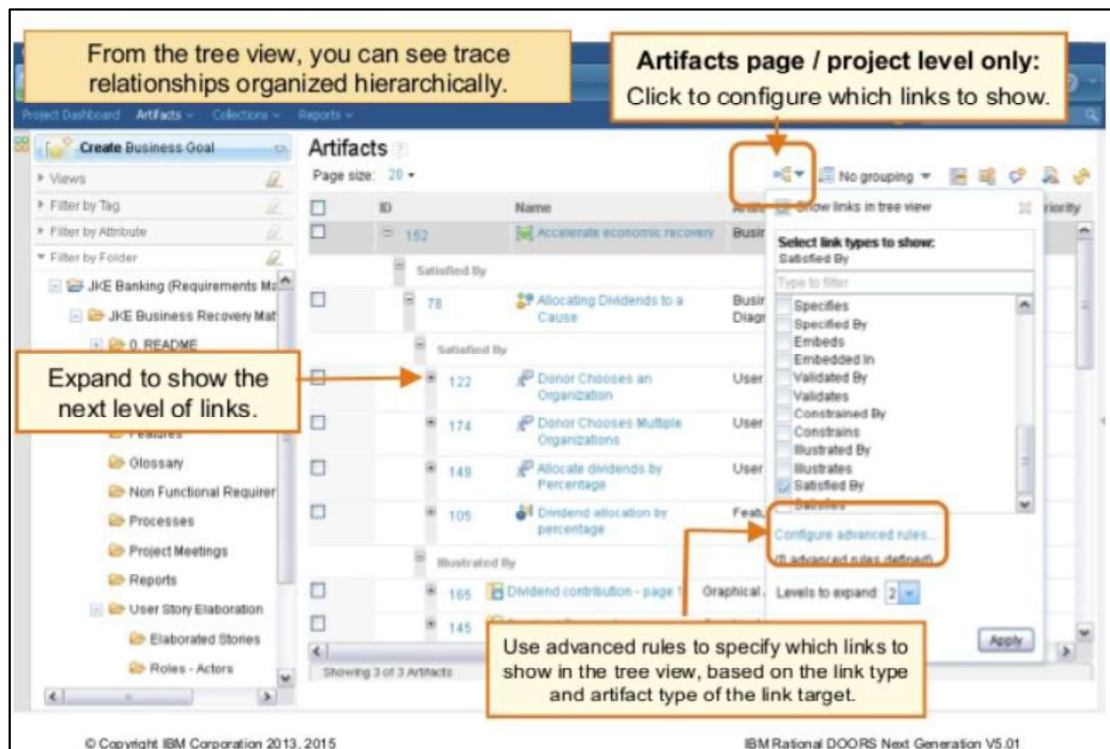


Figure 7: Traceability in IBM (<https://www.ibm.com>)

- Requirement coverage analysis can be done using filters. In Figure 7, a filter was set to show “satisfied by” link types only. Filters help to exclude some of the data and focus on data that is of interest to the user. This helps in managing a huge number of links.
- Different views are available for impact analysis with various impact analysis profiles and diagrams. In this depth, direction, and kinds of links, as well as kinds of artifacts that are included in the diagram, can be restricted based on the need.
- Suspect links feature are present which highlights the artifacts that need to be rechecked after change requests are approved.
- Dynamic reports of end-to-end traceability can also be generated.

2.4.2 Traceability in Application Lifecycle Management Tools

Application Lifecycle Management (ALM) tools are used for managing all the artifacts from requirements to test reports. They cover the entire lifecycle of a project. These tools help in collaboration between different stakeholders and help to manage the artifacts efficiently. Most of the tools help in estimation, planning, software development, quality management, configuration management as well as provides features to maintain traceability efficiently. They provide lifecycle views i.e., views are provided to help to check the relationships across software development artifacts. They also feature lifecycle queries and filter to focus on finding gaps and assessing quality. Some of the popular ALM tools are Polarion, System Weaver, and PTCIntegrity. Polarion is considered as an example tool to list down the features. Similar features will be present across other ALM tools for traceability.

Below are some of the main features of Polarion for traceability:

- Polarion supports traceability across work items (artifacts) and provides easy navigation for the analysis. It provides the feature to import artifacts from other authoring tools in various formats like Excel, Word, CSV, HTML, ReqIF, etc.
- The process can be defined to mandate the linking of work items using link roles in the day-to-day activities.
- The relationships that can exist between artifacts can be defined based on project needs using “link roles”. In Figure 8, new link roles pvRequires and pvConflicts are created apart from the default tool-provided roles.

workitem-link-role-enum.xml					
ID	Name	Opposite Name	Parent	Default	Rules
relates_to	relates to	is related to	<input type="checkbox"/>	<input checked="" type="checkbox"/>	
parent	has parent	is parent of	<input checked="" type="checkbox"/>	<input type="checkbox"/>	1
refines	refines	is refined by	<input checked="" type="checkbox"/>	<input type="checkbox"/>	1
duplicates	duplicates	is duplicated by	<input type="checkbox"/>	<input type="checkbox"/>	1
verifies	verifies	is verified by	<input type="checkbox"/>	<input type="checkbox"/>	1
pvRequires	requires	is required by	<input type="checkbox"/>	<input type="checkbox"/>	1
pvConflicts	conflicts with	is in conflict with	<input type="checkbox"/>	<input type="checkbox"/>	1
			<input type="checkbox"/>	<input type="checkbox"/>	

Figure 8: Link roles in Polarion (<https://almdemo.polarion.com>)

- Polarion provides queries to visualize and analyze the trace links. New queries can as well be created based on user-specific needs to focus information on only the required trace links.
- Suspect functionality is present in the tool which is used to highlight suspect links when work items are modified. This ensures that users look into the changed artifacts and update the related artifacts and links after analysis.
- Rules can be defined based on the link roles and work items to control the creation of trace links and to avoid the creation of ambiguous/incorrect relationships. Figure 9 shows the rule creation option in Polarion. i.e., System Requirements can be linked to only other System Requirements. A task can be linked to artifacts types like Change Request, Issue, and System Requirement. Similarly, Change Request can be linked to any other type of artifacts and Issue can be linked to artifacts of type Task only.

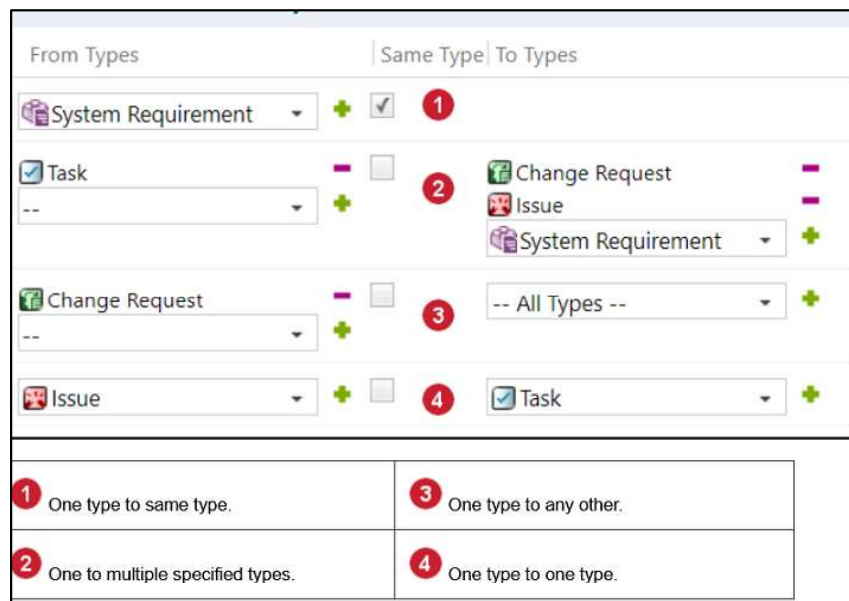


Figure 9: Link rules in Polarion (<https://almdemo.polarion.com>)

2.4.3 Traceability using General-purpose Tools

Traceability can be created with the help of general-purpose tools like Microsoft Excel using matrix representation. It is a simple and effective tool to have bidirectional traceability for the project. However, if the process of creation and maintenance is not automated, it is error-prone. If during project evolution, a matrix is not maintained then it leads to trace links decay making the traceability matrix unusable. Also, trace links when maintained in excel are not

scalable. In Figure 10 below, trace links between Requirements and Architectural elements are maintained in the form of a matrix. Requirements are maintained in the rows and architectural elements in columns. The trace links are marked “x” and no other information like trace link type, cardinality, etc are present. An example trace link is REQ_ID1 is traced to ARCH_ID1 and ARCH_ID6. Similarly, ARCH_ID4 can be traced to requirements REQ_ID4 and REQ_ID11. Hence the view is bidirectional.

		Architectural elements									
		ARCH_ID1	ARCH_ID2	ARCH_ID3	ARCH_ID4	ARCH_ID5	ARCH_ID6	ARCH_ID7	ARCH_ID8	ARCH_ID9	ARCH_ID10
Requirements	REQ_ID1	x					x				
	REQ_ID2										
	REQ_ID3										x
	REQ_ID4				x						
	REQ_ID5										
	REQ_ID6	x									
	REQ_ID7										
	REQ_ID8									x	
	REQ_ID9		x			x					
	REQ_ID10										
	REQ_ID11				x						
	REQ_ID12										
	REQ_ID13									x	
	REQ_ID14						x				
	REQ_ID15							x			

Figure 10: Traceability matrix in Excel

2.4.4 Standalone Traceability Tools

These are the tools specifically created to manage traceability. It imports artifacts from other tools using adapters of different tools and establishes trace links in the traceability tool. They provide a wide range of features like trace queries, visualization, navigation, etc to use and analyze the trace links efficiently. Some of the standalone traceability tools are Yakindu and Eclipse Capra. Yakindu is considered as an example tool for compiling some of the main features present in Standalone traceability tools.

Following are some of the key features supported by YAKINDU, a professional requirement traceability management tool:

- Yakindu supports a wide range of tool adapters. Hence supports importing of the software artifacts in various file formats for the creation of trace links.
- It provides full visualization of all the data from different artifacts and how they are linked. Also, Yakindu supports a wide range of filters and user-defined queries that help in the analysis of traceability information. Figure 11 shows the possibilities in

selecting the depth and the filter for visualization (filter to show only suspicious links, duplicate links, etc). This helps to analyze the traceability information part by part when the number of artifacts and the links between them is huge.

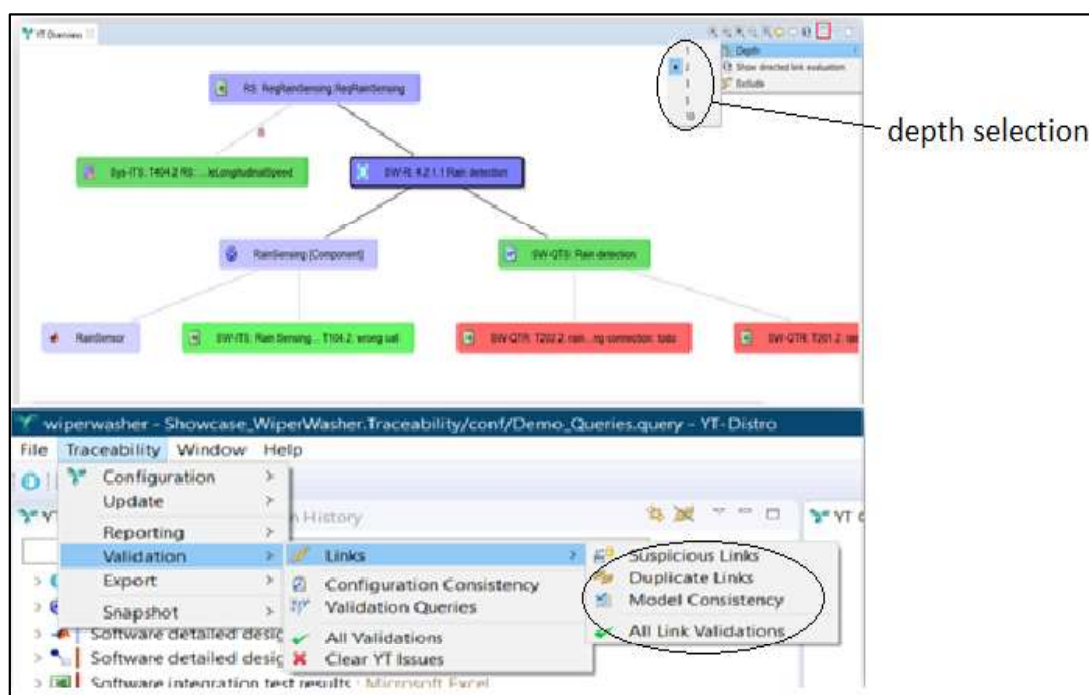


Figure 11: Traceability visualization filters in Yakindu (<https://www.itemis.com>)

- The tool provides an overview of the project and enables access to all the relevant information by just exploring the graph. It can be configured based on project needs (link type, rules, depth, query, suspicious links, etc)
- It also provides various project metrics and traceability reports in the dashboard. Figure 12 shows an example dashboard containing two different traceability reports. In the left window, the percentages of trace links between various linked artifacts are shown in the form of a pie chart and hence displaying coverage report. In the right window, traceability links are shown in the tree format from the requirements level to the test report level. Each artifact level can be differentiated by colors. Here again, filters or queries have been used to select only two requirements.

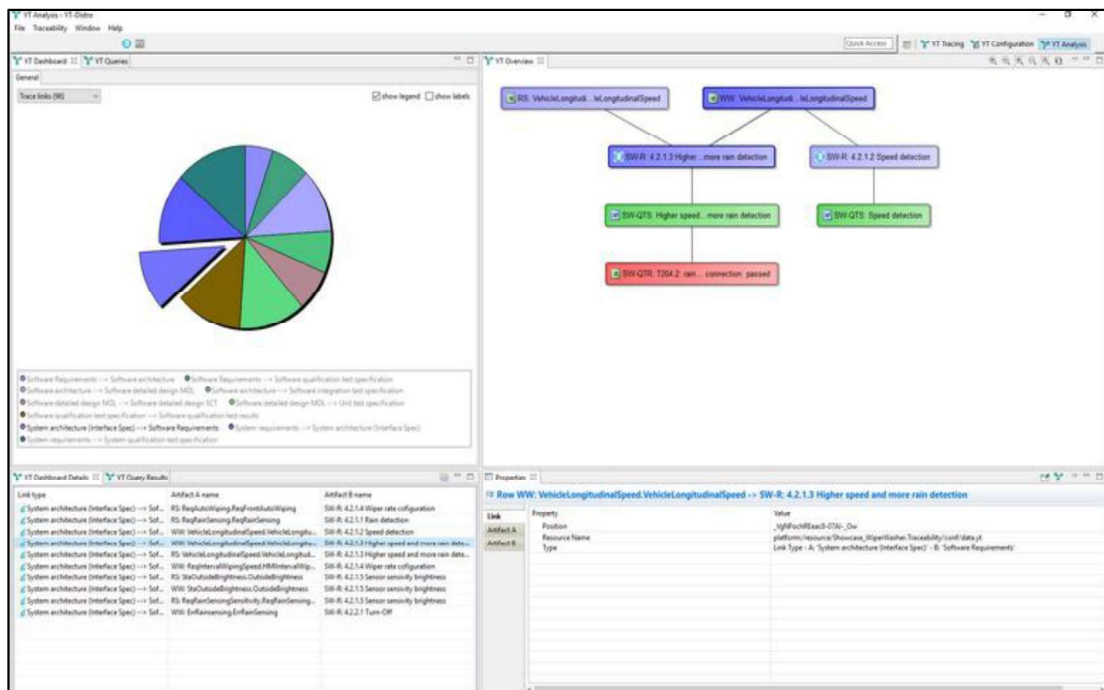


Figure 12: Dashboard in Yakindu (<https://www.itemis.com>)

2.4.5 Traceability in the Issue Tracking Tools

Some of the issue tracking tools like Jira could be used to manage requirements and tests just after plug-and-play configuration. The plugins like Requirements & Test management for Jira help to track the requirements with the development of corresponding test artifacts and hence indirectly support traceability. Following are the features supported:

- Establishes trace links between requirements and corresponding test artifacts by drag and drop feature
- Tree-structured folders during the development help to speed up test creation, execution, and management. Organization of folders by version helps to develop trace links along with version information.
- Traces help for the easy test execution for any update in requirements
- Helps for tracking the progress of the project using the trace links. Provides various kinds of reports to analyze the same. For example, traceability matrix and full requirements coverage
- Provides flexible filters to get different views of the dependencies between artifact
Also, the reports can be exported in various formats.

3 Expert Interviews on Traceability

This section describes the purpose and procedure of data collection done by conducting expert interviews. The observations made during interviews gave comparable data on the criteria considered, purpose, challenges, and best practices of traceability process in the industrial setting. Hence the data collected were used to find and map answers to the research questions.

3.1 Introduction

Structured interviews were conducted with industrial experts about the traceability process followed in their companies in general and in their projects in particular. This method of research was chosen because experts will provide valuable insights based on their real-life industrial experiences which can be very difficult to collect with any other methods. The main purpose of the interview was to gather data to identify the challenges faced while trying to achieve traceability. Also, the recommended approaches and best practices followed were compiled. The interviews involved participants from different companies and working on different embedded system projects to get a broader perspective of the challenges faced. Also, the involved participants were from different organizational levels and hence could collect varied views based on their working context.

3.2 Procedure and Topics Covered in the Interview

The interview was conducted in English. At first, the main goal of the interview was set, i.e. understanding the traceability process followed in their company. In the next step, the interview questions that were prepared by using the information collected during the literature review (chapter 2.3.3) were asked. The interview questions were divided into the following five sections: introduction, basics of traceability, approaches followed, tools used, and traceability with respect to MBSE. In the **introductory** section, the characterization of the company and projects selected for the interview were understood. To understand an overall context in which traceability is being applied, the application domain, standards used in the projects, the software development life-cycle models followed were quizzed. In the next section, the **basics of traceability** were discussed. The main purpose of traceability, different

artifacts that are considered for the creation of trace links, and the benefits of trace links in their projects were understood. The purposes discussed in chapter 2.3.3.5 were mapped to understand the reason that drives their projects to have traceability. The most benefitted stakeholder and the difficulties faced while achieving traceability were also discussed. The next section deals with the **approaches** followed to achieve traceability. Below Figure 13 shows the snippet of the questionnaire used during expert interviews. This lists the questions covering the topic “Traceability - Approach”

Traceability - Approach	
9. When do you create trace links? Is it requirements-driven or captured during transformation from one artefact to the other?	-
10. How do you represent the trace-links? (e.g. matrix, hyperlinks, graph)	-
11. What happens on the change or deletion of artefact and trace link? Is automatic change propagation expected as per trace links creation/deletion?	-
12. Is traceability also created and maintained between artefacts developed in different companies? If so, what are specific challenges / approaches to this end?	-
13. Any metrics used for measuring the quality of traceability? What does correct traceability mean according to your knowledge?	

Figure 13: Snippet of expert interview questions

The main criteria discussed in chapter 2.3.3.4 were quizzed and the approaches followed were compiled along with the timeline of the creation of trace links, their usage, and maintenance. The metrics used for traceability measurement, coverage, and quality of trace links were discussed. In the next section **tools** used in their projects for the creation of trace links were discussed. The different features supported along with visualization and navigation of trace links were compiled. The different types of reports generated and their usages were understood. Finally, to understand the process of **traceability in MBSE**, MBSE specific questions were asked. The challenges faced and the best practices encountered during the

traceability process along with their experiences were compiled. Their thoughts on how MBSE eases or complicates traceability were also gathered. The detailed questionnaire used during expert interviews can be found in Appendix: 7.2.

The interview questions were sent to the interviewee beforehand for a glance to know the goal and context of the interview. Also before interviewing, an overview of the purpose of the study was explained. The name of the company was kept anonymous when chosen so by the participants. Each interview took approximately 40 minutes and was recorded only with the consent of participants. The answers given by the interviewee for each of the questions were noted and rechecked later with the audio recording when available. Later after compilation, the answers were shared with the participants for their review and updated for any comments.

3.3 Data Collection

As per the questionnaire (refer to Appendix 7.2), the interview was conducted and the results were collected during the interview. The data collected were divided into sections and subsections based on the topic of discussion. Below is the summary of information compiled from the expert interviews.

The interview was conducted with three participants from different companies out of which two were from automotive domain backgrounds. Also, the participants belonged to different hierarchies of the organization i.e. Project Manager, System Architect, and Software Developer. Hence traceability with respect to the managerial point of view as well as system and software developmental point of view was understood. From the discussion, it was noted that traceability was mainly used for the validation and verification of artifacts. The other main purposes mentioned were mostly mapped to uses mentioned in chapter 2.3.3.5. It was mentioned that traceability was also used to maintain a consistent system and to check for the completeness of the system. According to all of the participants, traceability was mainly maintained between main artifacts like requirements, design decisions, architectural elements, design elements, implementation, test cases, and test reports. However, in few cases, deeper level links were created based on the project's needs. For example, links were created between blocks and parts of SysML diagrams, between operations of blocks and call operations of activity diagrams, between interface blocks, flow properties, and value properties, and between blocks and constraints. While discussing the approaches followed for

traceability, the following points were noted. Most of them preferred to create traces close to the establishment of the work product (time-wise) to avoid rework. However, in some cases, it was delayed because of the delayed access to work products created by different teams. The useful representations of trace links mentioned varied from one another because of their different usage contexts. Following were the most preferred visual representations mentioned by the participants which they used in their daily activities. Hyperlinks were used for navigation between artifacts, graphs for impact analysis and to check for completeness, tables for comparison and maintenance, matrix for quantity checks, and diagrams like cake diagrams were used for completeness check reports. Traceability across teams was discussed mainly with respect to requirements. Standard formats like ReqIF, HIS, etc. are used for exchanging requirements between teams. Apart from that, it was told by one of the interviewees that OSLC capabilities were also being explored for the import and export of artifacts. In the next part of the interview, the pros and cons of traceability in MBSE were compiled based on their feedback. This helped to compile the answer for the research question RQ4 (How does MBSE ease or complicate traceability?). The mentioned pros of using traceability in MBSE were a better understanding of the whole system and easy identification of reusable elements. In few cases, it was told that as trace links were one of the outputs of model transformation, it was easier to create. Also, it was mentioned that in an ideal scenario, if a single tool was used for the development of all the artifacts in a MBSE project, then it eases the trace link creation process. The main drawbacks mentioned were tool breaks and difficulty in deciding on traceability schema because of the huge number of artifacts. Also tackling a huge number of trace links posed a challenge for visualization. Complete answers recorded during expert interviews can be found in Appendix: 7.3.

3.4 Analysis

Based on the discussion during interviews and data collected on traceability practice and process, analysis was done to list down the challenges faced during the creation and maintenance of trace links between artifacts. Though criteria and approaches can be followed as per the theoretical knowledge in a project, the challenges faced will be known only during the real usage scenario. Hence based on the discussion with the experts, the issues faced were compiled. After analysis, the challenges were grouped into the following four major categories:

Understanding of traceability: Unclear information leads to uncertainty on traceability and makes it difficult for the stakeholders to abide by the process defined. An unclear process on how, what, and when the trace links must be created and maintained, leads to distrust and hence a reason to not follow the process. The other issue is deciding on the traceability schema. It must be well thought of and well documented for the stakeholders to understand. Lack of knowledge leads to misinterpretation of trace links and hence results in the wrong analysis.

Organization and process: The next category is based on the organization and process followed as traceability needs vary from one project to other. Hence tailoring of traceability process must be done for its effective application. The other challenge is distributed development environment. When more than one team is involved in a project, artifacts creation might happen parallel. Hence a process must be defined for their exchange/access and an approach must be decided for the creation of trace links between them. Hence process must be defined beforehand.

Human factors: Misuse and distrust are the major challenges. Sometimes documenting all the information of an artifact along with stakeholder name responsible, can be misused as input for employee's performance evaluation. This might result in fear of capturing all information during trace link creations. One should make sure that trace links are used for constructive purposes only. Also, all stakeholders must have complete trust in the traces so as to use them to their full potential. As traces are created in advance and used only later, engineers might even consider it as an overhead. Hence stakeholders must be educated regarding the traceability process.

Tool support: For traceability to be cost-effective, it must be automated with the help of tools. These tools must be configurable because of the variety of tools involved in the creation of artifacts and to avoid tool breaks. The framework must be reusable if the tools involved in the project change. The tool must help in the maintenance to avoid trace decay. Handling a huge number of links for the analysis is one of the major challenges. Even when the links are correct, if the report is non-readable because of the sheer number of links, then traceability cannot be used.

3.5 Conclusions

This section describes the conclusions derived based on the analysis of the data collected and based on the discussion during expert interviews. It mainly concludes on the best practices to be followed and pitfalls to be avoided during the traceability process which in turn answers the research question RQ3 (What are the best practices and pitfalls present in the existing approaches/tools for the creation of trace links and their maintenance?).

3.5.1 Best Practices to Follow

Below are the best practices that must be applied for easing the traceability process and for its optimal usage.

Integrate traceability tasks into existing work practices: Traceability tasks must be part of the development process. Also, traces must be created close to the establishment of the work product (time-wise) to avoid reanalysis of the work products during the creation of trace links.

Having just enough traces: Having trace links between every artifact to the other artifact does not serve the purpose. It results in clutter and too much information might not benefit the project. Also understanding the purpose helps in selecting value-based traces.

Aim for either a holistic solution or a completely separate TM tool: To avoid tool breaks or to avoid porting issues of the trace information from one tool to the other during a tool change, tool configurations must be considered before selecting a traceability tool for a project. Since software development environments usually will be heterogeneous, choosing a holistic approach that can easily be adapted to various tools or choosing a completely separate traceability management tool where import/export of artifacts information from various other tools is possible is the better way to achieve traceability in a project.

Usage of the common standard: Using common standards and formats for exchanging information between different tools ensures configurability and hence avoids any tool breaks. Having a meta-model for traceability also helps to have consistent trace links across the project. Meta-model describes rules like which kinds of artifacts can be linked to each other and what types of links are allowed.

Artifacts linked must be version controlled: As software keeps evolving due to requirements changes or due to design changes, having traces between artifacts with version information is very important. Having configuration management coupled with a traceability process helps. Traceability reports considering versions of the artifacts give the latest information on the evolution of the artifacts involved and hence are much more reliable. Also applying traceability after a freeze and baselined versions ensures that the further artifacts are being developed on stable versions.

Have explicit trace links: Even though implicit traceability does not require any tools, it is not the best solution as it does not provide any visual report nor provide immediate trace information when needed. Hence maintaining explicit trace links must be considered as the optimal way to achieve traceability.

Trace links should be maintained across life-cycle: If trace links are not maintained across the life cycle, the traceability information gets decayed. As the software system keeps updating, not changing the trace link information accordingly will lead to wrong relationship information between artifacts in trace links. Hence maintaining trace information consistently is the key to achieve success in traceability. Trace links must be reviewed and enhanced iteratively to detect wrong links at the early stages.

3.5.2 Pitfalls to Avoid

During the process of the interview, the usual difficulties faced and the probable steps/process to be avoided while trying to achieve traceability were also discussed. Below are the major pitfalls to avoid:

Trying to achieve traceability at the middle or end of the project: One of the major problems in achieving traceability is that it is not thought of as critical as it should be during the beginning of the project. In many cases, the trace links are tried to be created during the middle or end of the project to achieve certification which mandates the process. Because of which the quality of trace links is compromised. This results in no actual use of the trace links.

Collection of large data: Having traceability links between every artifact without considering the overlap might result in an unmanageably large number of links. It not only increases the

project cost during creation, but it also demands a huge amount of effort for its analysis and maintenance.

Not allocating work resources and effort for traceability creation and maintenance: As we already know that for achieving traceability, a good amount of effort needs to be spent on its planning, creation, and maintenance. However, mostly traceability is perceived as a low-priority task due to which dedicated time will not be allocated for this task. Hence the developer considers it as low priority task and does not give required attention during its creation leading to incorrect or inadequate traces.

Lack of communication: A clear knowledge of artifact ownership, artifact sharing, security, trace link creation, and maintenance ownership should be present across all stakeholders. Regular review meetings between teams must be conducted.

Rigid or toolchain specific: As the project evolves, tools used might also change as per needs. Hence if the traceability tool/approach used is too rigid, the same cannot be ported and hence all previous trace link data will be lost. Also, huge effort needs to be spent for its re-creation as the previous data is not reusable.

Inconsistent/erroneous links because of automation: Tools must be tested thoroughly before using. The trace link information created must be reviewed iteratively so that erroneous or inconsistent trace links are found at the beginning. Even though automation reduces effort in the traceability process, it might cause erroneous links as well. Hence reviews and trustworthy tools must be considered for the better quality of trace links.

4 Feasibility Study and Investigation

From the literature review on traceability, it was found that traceability is a critical element of the system and software development process. Also as mentioned in chapter 2.3.4, ASPICE demands bidirectional traceability and most of the automotive projects follow ASPICE. Hence, achieving bidirectional traceability has become a necessity along with the need. However, based on the findings from the expert interviews, engineers face various kinds of challenges to achieve traceability successfully. One of the major challenges was tool breaks as various tools are usually used for the development of artifacts in MBSE. To investigate further on the tool support, various approaches and tools available for establishing trace links between artifacts were explored. As there are numerous tools present for the development of artifacts, a feasibility study was conducted on a particular toolchain consisting of Polarion, Enterprise Architect, and Doxygen. Investigation on various tools and approaches for establishing bidirectional traceability between the artifacts created on this toolchain was performed.

4.1 Approaches of Traceability across the Toolchain

This section describes the various viable approaches for achieving traceability across the toolchain consisting of Polarion, Enterprise Architect, and Doxygen. The different approaches/tools present that help in achieving traceability for the above tools are explored.

4.1.1 Used Toolchain

The toolchain consists of **Polarion ALM** which has a requirements module to work on requirements specification. In this module, it is possible to manage from stakeholder requirements to system requirements and to finally software requirements. It also has review, approval, and traceability functionality. Hence managing tracing between requirements can be achieved easily in Polarion. Polarion also consists of a QA module that includes test planning, test execution, and test reports management. In Polarion all the artifacts created are considered as work items and traceability functionality provided manages the trace relationships between any two work items. Hence traceability between requirements and

corresponding test cases, test reports can be achieved from the built-in traceability functionality. The next tool considered is **Enterprise Architect**, a graphical tool designed to help build visual models and architectural design elements based on the OMG UML. Enterprise Architect provides useful tool functionalities for exploring the relationships between various model elements. The last tool considered in the toolchain is **Doxygen**. Doxygen is a document generator tool that extracts information from the comments of the source code. It provides a cross-reference between design documentation and code.

For the feasibility study, bidirectional traceability must be achieved between artifacts created in the above tools as per ASPICE requirements. Figure 14 depicts the desirable trace links expected between various artifacts. The dotted line with arrows represents the bi-directional trace link between artifacts. For clarity purposes, each line is represented by only one trace link type. For example, the trace link type between architectural element and requirement is mentioned as “satisfy”. This means that the architectural element “satisfies” the corresponding requirement and in the other direction, the requirement is “satisfied by” the corresponding architectural element. Following are the other various trace links between the

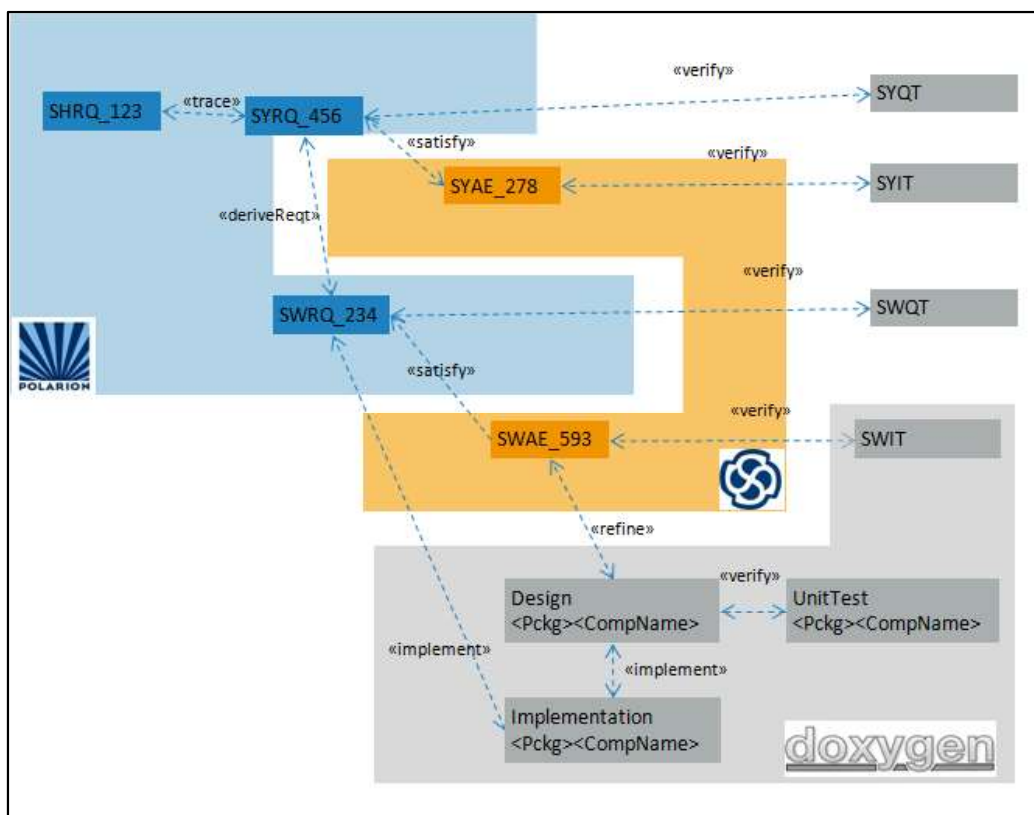


Figure 14 : Traceability in the toolchain (Polarion<->EnterpriseArchitect<->Doxygen)

artifacts described in Figure 14. The stakeholder requirements (SHRQ), system requirements (SYRQ), and software requirements (SWRQ) are created inside Polarion and trace links between them are also managed in Polarion. The system architectural elements (SYAE) and software architectural elements (SWAE) developed in Enterprise Architect must trace to requirements in Polarion using “satisfy” relationship. The implementation and design elements in Doxygen must trace to requirements in Polarion and architectural elements in Enterprise Architect using “implement” and “refine” trace link type respectively. The test artifacts like System qualification test specification (SYQT), System integration test specification (SYIT), Software qualification test specification (SWQT), and Software integration test specification (SWIT) are not considered for the feasibility study. However, traceability can be achieved easily when all of them are developed in the QA module of Polarion. The trace link between test specification and other artifacts is of type “verify”. All the work items created in Polarion uses traceability functionality provided by the tool. To achieve traceability between the requirements, architectural elements, and implementation elements generated in the toolchain, below two main approaches can be considered.

4.1.1.1 Managing Traces in Polarion

In this approach, all traceable artifacts must be present in Polarion as work items. Artifacts created in other tools must be represented in Polarion as proxies. i.e. traceable artifacts must be imported to Polarion. All the artifacts, including imported ones, are considered as work items in Polarion. Trace links are then created and maintained easily between work items using the traceability feature in the Polarion. For example, consider the artifacts mentioned in Figure 15. The architectural elements SYAE_278 and SWAE_593 are exported from Enterprise Architect to Polarion and implementation element SWUI_849 is also exported from Doxygen to Polarion. Hence all the artifacts are present in the Polarion as work items and using tracing functionality trace links can be created easily inside Polarion. Hence in this approach, Polarion acts as the master and controls the creation and maintenance of trace links. The benefit of this approach is that all trace links are created and maintained in a single tool. Various trace reports can be generated as Polarion provides the feature by default. However, the drawback is the risk of missing the updated traces because of missed synchronization from other tools. Also, experts must switch between tools and hence must be well-versed with the Polarion tool as well for the creation and maintenance of trace links.

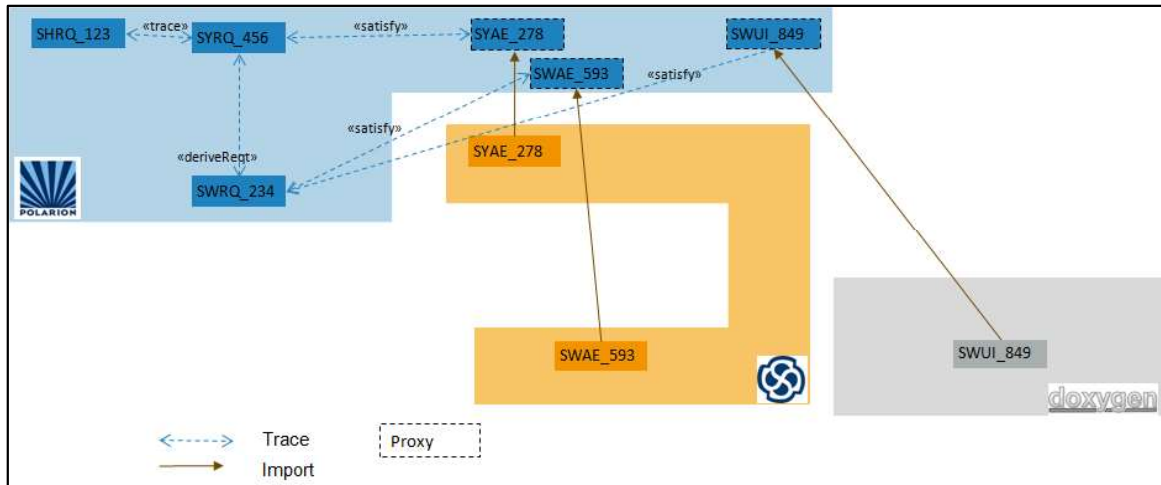


Figure 15: Trace links in Polarion

4.1.1.2 Managing Traces in Other Tools

In this approach, requirements are represented in the respective tools as proxies. i.e. Polarion exports the requirements to EA and Doxygen. Traces are established between imported requirements and respective artifacts in both tools. Trace information is then transformed to general formats like Microsoft Excel or HTML page. For example, consider the artifacts mentioned in Figure 16. The system requirement SYRQ_456 is exported from Polarion to Enterprise Architect. And the software requirement SWRQ_234 is exported from Polarion to both Enterprise Architect and Doxygen to Polarion. Hence trace links are created separately in Enterprise Architect and Doxygen. The trace information (for example SYAE_278 satisfies

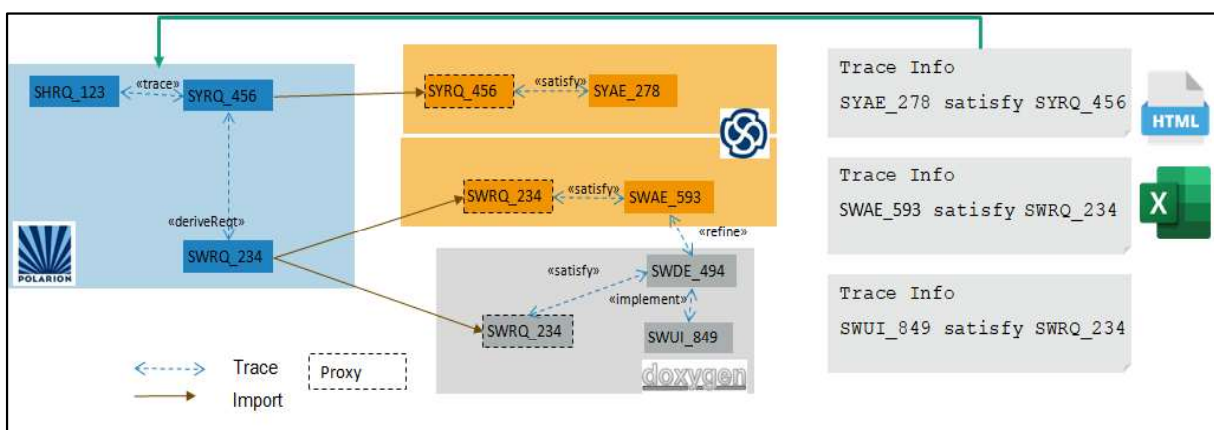


Figure 16: Trace links outside Polarion

imported requirement SYRQ_456) can then be transformed into a Microsoft Excel report or HTML page report for the analysis. These reports can then be exported to Polarion as well.

The main benefit of this approach is that experts define the trace links inside the tool where artifacts are being developed. Hence no switching to a different tool is required. For trace link analysis, general-purpose tools can be used.

4.1.2 Establishing Traceability between Polarion and Enterprise Architect

This section describes various tools/approaches present to create links between requirements in Polarion and architecture model elements in EA.

4.1.2.1 Using Open Services for Lifecycle Collaboration

Open Services for Lifecycle Collaboration (OSLC) is an open community creating specifications for integrating tools. It is based on the 3C Linked Data. OSLC provides standardized self-descriptive REST APIs which allow vendors to provide a fully supported integration with many other OSLC-compliant tools. When using OSLC for trace links creation between tools, each artifact is described as an HTTP resource, identified using a Uniform Resource Identifier (URI), accessed and manipulated with the GET, PUT, POST, and DELETE HTTP methods. Figure 17 depicts integration between two OSLC-compliant tools. Tool X is the OSLC Provider which uses a web service to store and provide data by implementing CRUD (Create, Read, Update, Delete) functionality. Tool Y is the Consumer which can request and manipulate provided data via HTTP requests (HTTP GET, POST, PUT, and DELETE methods).

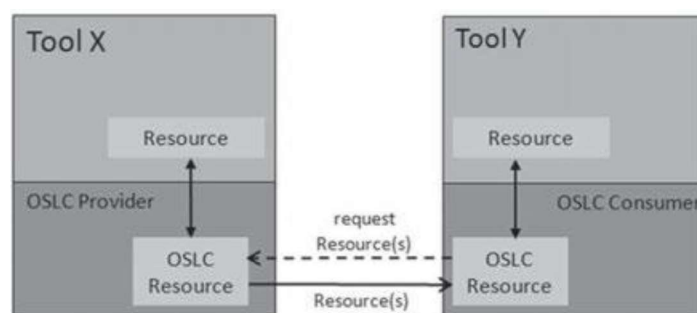


Figure 17: OSLC Provider and Consumer (Kaiser & Herbst, 2015)

To use OSLC to link between Polarion and other tools, Polarion acts as a Linked Data provider. Polarion work items can be linked to objects that reside on external Linked Data enabled tools using the linked data feature. Polarion exchanges linked data with a friend server, one that hosts an application that has been configured and mapped to exchange data

with. To exchange linked data between Polarion and Enterprise Architect, Polarion uses Enterprise Architect Pro Cloud Server (separately purchased and separately licensed edition of Sparx Systems Cloud Services) that supports OSLC and is configured as its friend server. Enterprise Architect acts as an OSLC Provider which allows for creating, retrieving, and querying Enterprise Architect resources (packages, elements, diagrams, and connectors) in a model via Pro Cloud Server. With OSLC support, resources in an Enterprise Architect model can be identified and accessed using a unique URL that can be linked to resources in Polarion.

One of the main advantages of using OSLC for traceability is that the artifact information need not be moved from one tool to the other i.e. no export/import is required. And trace links can be created across tools. Due to the resource linking approach of OSLC (instead of data synchronization), the typical integration challenges of traceability, data consistency, and data interoperability across the whole lifecycle process are appropriately managed and therefore assist collaboration, reuse, and integration (Kaiser & Herbst, 2015). Also, the links can be created and scaled easily. However, the major drawback for EA is that Pro Cloud Service must be purchased and hence may not be economical. The other disadvantage is sometimes OSLC provides a user interface with a very small amount of information of the accessed artifacts (like only version number without any other context). Without additional context, it is difficult to choose the right artifact version while creating or updating the links (Kaiser & Herbst, 2015).

4.1.2.2 Using Polarion Connector for Enterprise Architect

Polarion Connector for Enterprise Architect (EAPO) tool is used to connect Polarion with Enterprise Architect. It helps to synchronize EA diagrams and Polarion work items from the EA interface. It is mainly used to manage the approval life-cycle of EA diagrams and to generate Polarion Documents with EA diagrams images. As EA diagrams/elements can be exported and synchronized to Polarion using the EAPO connector, it can as well be used to generate trace links between EA elements and Polarion work items. Hence links between EA elements and requirements can be maintained in Polarion.

Once EAPO has been downloaded, Polarion can be connected from EA using Polarion Integrator. Integrator can be used either to import to Polarion or export from Polarion. A local database will be created for the mapping of EA elements to Polarion type. And the mapping is

done in the mapping window provided by the connector and the user can decide on the elements to export and the types to be mapped. One of the main features of EAPO is that it also provides advanced mapping. In this, EA tagged values are mapped to Polarion custom fields. Figure 18 shows the exported results. It shows the mapping of EA elements (Use cases) to Polarion types (workpackage). It can also be noted that users can navigate to open EA elements in both Enterprise Architect as well as Polarion.

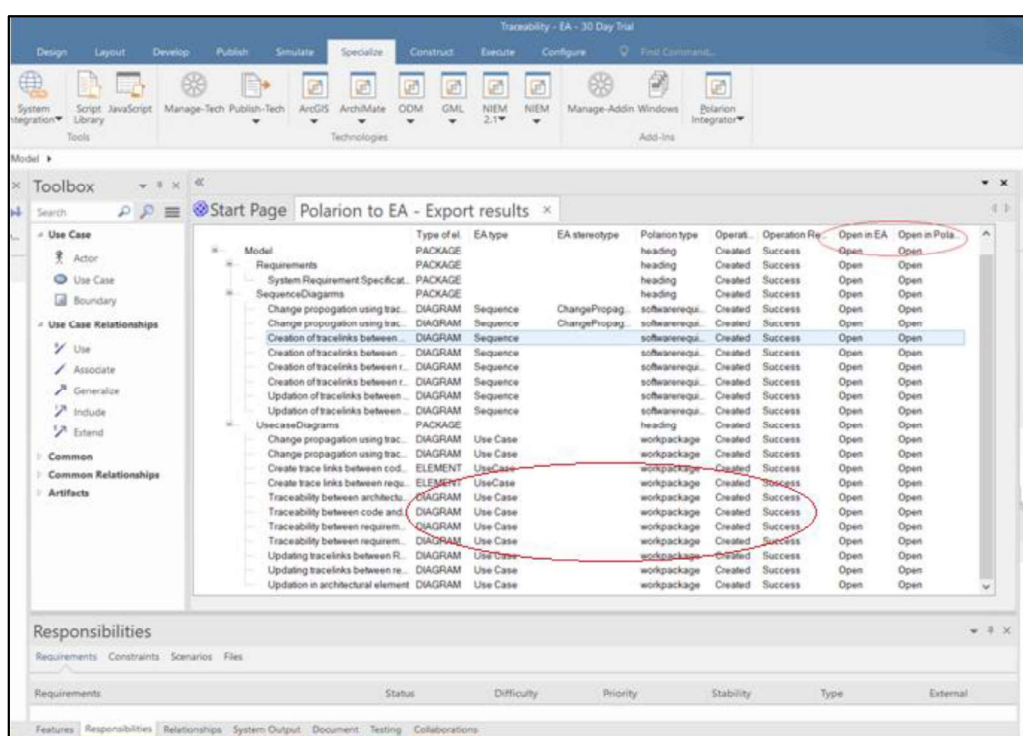


Figure 18: Export of EA elements to Polarion using EAPO

Once exported, EA elements are treated as work items in Polarion. Hence using link functionality, EA elements are linked to Polarion requirements. Re-import is done for the synchronization of EA elements in Polarion. When EA tagged values are included during imports, they can be used to identify the updated EA elements inside Polarion. By using query filters with custom fields, the updated EA elements can be marked as suspect. Suspect links are created for these marked EA elements.

One of the main advantages of EAPO is that EAPO is free and fully integrated within EA installation and hence no third-party server is required. The mapping configuration and synchronization data are stored in an external database and not within EA or Polarion

projects. Also starting point from where model elements are to be exported can be chosen which is useful for huge projects. The disadvantage of the tool is creating a mapping of EA elements to Polarion type is cumbersome and depending on the number of elements selected, export could take a lot of time. Also, the tool does not support updating the Polarion type of a mapped EA element in the subsequent imports. The other major drawback is that, when EA elements are updated and synced, the Polarion does not mark it as a suspect automatically. The user needs to check and update the element as suspect manually.

4.1.2.3 Using Requirements Interchange Format

Requirements Interchange Format (ReqIF) is an exchange file format for exchanging requirements, attributes across software tools from different vendors. The format is a metamodel defined by an XML schema. ReqIF is used for exchanging information between RM tools and MDD tools (which are based on UML). The following approach is followed to establish traceability links between RM and MDD tools using ReqIF. As a first step, requirements are exported from the RM tool to the MDD tool using ReqIF. MDD tool imports these requirements in ReqIF and architectural models are created. The trace links are then created between requirements and models in the MDD tool. These trace links are then exported back to the RM tool using ReqIF representations. Analysis of the links can be done in the RM tool after importing the link representations in ReqIF.

The above approach can be followed for establishing trace links between Polarion (RM Tool) and Enterprise Architect (MDD Tool). Polarion supports ReqIF and hence requirements can be exported in the form of ReqIF files. Whereas for Enterprise Architect, plugin support is required to transfer representations of requirements in ReqIF to understandable UML format. A plugin called ReqXChanger is available which enables requirements synchronization between Polarion and UML models from Enterprise Architect (<https://extensions.polarion.com/extensions/232-reqxchanger-for-polarion>). This helps to link requirements to UML model elements in Enterprise Architect. The link information can be exported back to Polarion in ReqIF format for trace-links analysis. Figure 19 depicts the mapping of the approach when ReqXChanger is used for achieving traceability between Polarion and Enterprise Architect.

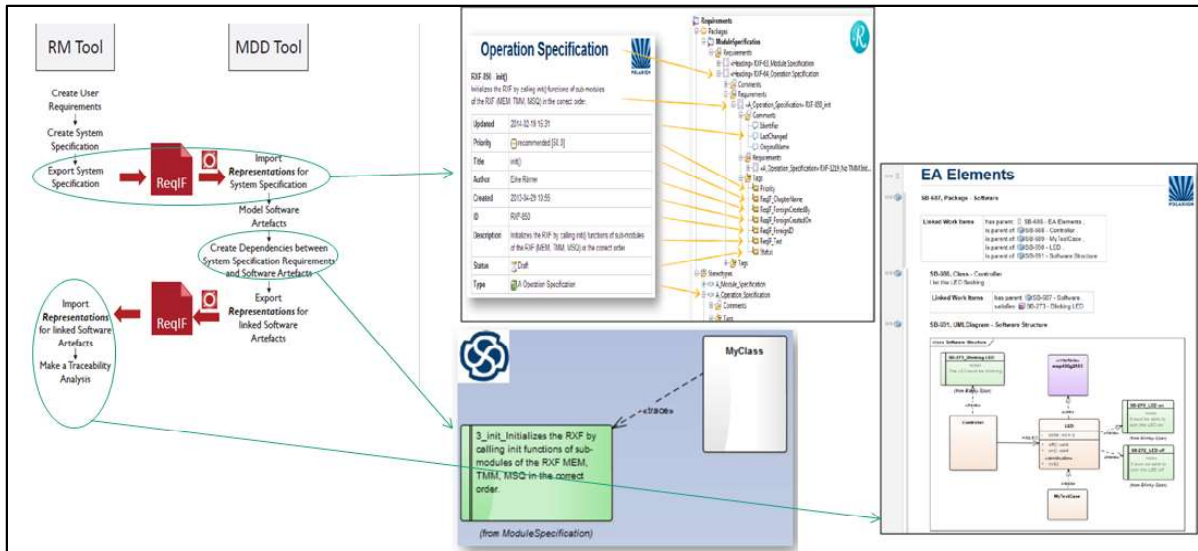


Figure 19: ReqXchanger for Polaron and Enterprise Architect

One of the major advantages of using ReqIF is that it contains structured data of requirements with related elements. This clear information helps to interpret the data in the other tool easily. When data between the tools are synchronized by re-import, the updated elements, attributes, and trace links will be shown clearly and automatically. Also, navigation to Polaron requirements from UML tools is possible. The major disadvantage is that the ReqXchanger is not available for free. Also, trace link creation in the MDD tool is cumbersome when a huge number of trace relationships need to be created.

4.1.2.4 Using CSV Import and Export

A comma-separated values (CSV) file is a delimited text file that uses a comma to separate values. Each line of the file is a data record. Each record consists of one or more fields, separated by commas. Almost all the RM tools, UML tools can represent their artifacts in CSV format and they also support the import and export of CSV files. Hence, the CSV format can be used for the exchange of artifacts between tools. Once the artifacts are exchanged, the trace links can be established in the RM or UML tools using the tool's link functionality.

This approach can be used to create trace links between Polaron and EA artifacts inside EA. Both Polaron and Enterprise Architect support import and export of artifacts in CSV format. Polaron requirements are exported in CSV format with the requirement's properties like ID, Name, Status, Version, etc. The requirements CSV file is then imported to the Enterprise

Architect tool. Enterprise Architect provides CSV Import/Export specifications using which requirement's properties can be mapped to Enterprise Architect's elements. Trace links are then created in EA between imported requirements and architectural model elements using requirements diagrams. Also, unique GUIDs are created in EA for each of the imported requirements. Hence the requirements when updated can be re-imported to EA without creating a new copy for each import. Traceability information in the form of a matrix can be visualized in EA. However, trace matrix can also be exported in CSV or Microsoft Excel format for further analysis. A prototype tool was created considering this approach and more details are provided in chapter 4.2.4.2.

The major advantage of this approach is that most of the tools support the CSV format. Hence can be used easily even if the tool changes. Also, users can decide on the elements of the artifacts that are imported and exported based on the project's needs. The process of importing and exporting can also be automated to avoid any manual errors. The major drawback of the approach is that it is prone to human errors when done manually. Also as the trace links are created in the UML tools, the creation of trace links requires more effort.

4.1.3 Establishing Traceability between Polarion and Doxygen

This section describes an approach that can be used to create trace links between requirements and implementation elements using the Doxygen tool. Custom tags in Doxygen comments can be used to describe trace information along with the corresponding requirement's ID. Using these custom tags, a traceability report in the form of an HTML report is generated by Doxygen. To add custom tags with trace information, Polarion requirements information is needed for the Doxygen tool. Hence requirements are exported into the implementation directory as a file with requirement anchors. Aliases are also added in the Doxygen configuration file which helps to map these anchors with custom tags in comments. Finally, when Doxygen is run, trace links are created in the form of a Doxygen web report. A list of requirements along with traced code elements will be generated as one of the web pages in the Doxygen report. Using the Doxygen HTML report, a CSV file can also be generated for further analysis of the traces. Further information on how this approach can be automated is explained in chapter 4.2.4.1.

4.1.4 Establishing Traceability between Enterprise Architect and Doxygen

This section describes an approach that can be used to create trace links between architectural and implementation elements using the Doxygen tool. The approach is similar to the one described above but instead of requirements ID, architectural elements information are included in custom tags. The Doxygen comments are updated for including the trace information with corresponding architectural model elements using custom tags. These tags in Doxygen comments represent the relationship between the elements. To add these tags in Doxygen comments, architectural elements information from EA is required. Hence these elements are exported into a code file with the model elements as anchors. Aliases are also added in the Doxygen configuration file which helps to map these anchors with custom tags in comments. Finally, when Doxygen is run, trace links are created in the form of a Doxygen web report. A list of architectural elements along with traced code elements will be generated as one of the web pages in the Doxygen report. Using the Doxygen report, a CSV file can also be generated for further analysis of the traces. Further information on how this approach can be automated is explained in chapter 4.2.4.3.

4.2 Proposed Solution

As already mentioned, it was noted from the expert interviews that one of the main challenges faced in achieving traceability in projects following MBSE approaches was tool breaks. Hence, a holistic solution must be aimed at rather than using different tools/approaches for a toolchain used in a project. Also, common formats supported by most of the authoring tools to exchange information between them must be used to avoid any rework when one of the tools is replaced in the future. Hence keeping these in mind, a prototype tool called **TraceGen** was built to aid trace link creation and maintenance across the tools of the considered toolchain. i.e. Polarion, Enterprise Architect, and Doxygen. The CSV and HTML formats are the most common formats supported by most of the tools and hence, they were chosen to exchange information between tools. Also for the proposed solution, the second approach of creating and storing links outside Polarion (refer to chapter 4.1.1.2) was followed because experts can define trace links within the tool that they are comfortable in without switching to other tools.

For the design and development of TraceGen, the acquired knowledge on research questions was used. The proposed solution was mainly based on the three main criteria discussed in

chapter 2.3.3.4. Also from the expert interviews, it was noted that one of the other challenges faced in achieving traceability in MBSE projects is having no clarity on the selection of subjects of interest (traceable artifacts) during the creation of trace links. Hence to tackle this issue, a traceability information model was created to define the traceable items properties and the trace-link properties. The tool was mainly developed using python scripts and batch scripts were used for automation. As the usage of trace links was one of the main criteria, the tool was also expanded to prepare inputs for the Power-BI tool for visualization. Power-BI is a free desktop tool used mainly for data analysis and supports a wide range of report generation (refer to chapter 4.2.5). Based on the inputs from expert interviews, the various possibilities of report generation in Power-BI were explored.

4.2.1 Traceability Information Model

A basic traceability information model (TIM) describes main two elements of traceability, traceable artifacts and trace links between these artifacts. The properties of the above two elements must be fixed to achieve consistent results across tools. This section describes the traceability information model created for the TraceGen tool.

Figure 20 represents the class diagram of the traceable artifacts. The main class “TracedItem” represents the traceable artifacts and is the parent class.

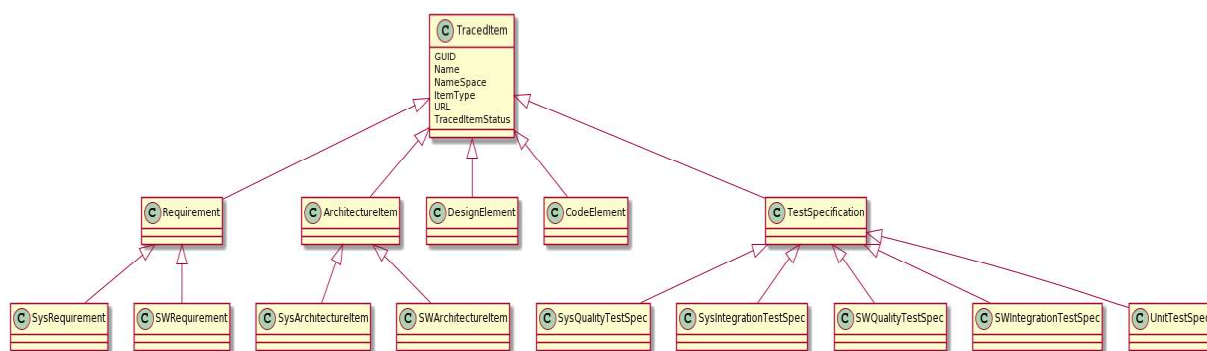


Figure 20: Class diagram of the traced items

The different types of traceable artifacts like Requirement, ArchitectureItem, DesignElement, CodeElement, and TestSpecification are child classes inheriting the member fields of the parent class. Each of these child artifacts is further classified into their child classes representing unique artifacts developed in a project. For example, the Requirement class is

further classified to system requirement (SysRequirement) and software requirement (SWRequirement) which are unique artifacts developed at a different period in a project. The classes are mapped to the artifacts considered in ASPICE (Refer to Figure 6).

The member fields considered for the main class “TracedItem” are listed in Table 7. During exporting and importing of the artifacts between various tools, the below fields are considered.

Table 7: Fields of TracedItem

Fields	Description
GUID	A unique identifier for the traced artifact
Name	Readable identifier of the traced artifact
Namespace	Type of the artifact (Requirement, ArchitectureItem, DesignElement, etc)
ItemType	Requirements can be functional or non-functional, ArchitectureItems can be Use case diagrams or Class diagrams, etc
URL	Hyperlink of the traced artifact for navigation purpose
TracedItemStatus	The status of the traced artifact (Status can be added, deleted, updated, unchanged)

The other element of TIM is TraceLink. It represents the relationship between any two traceable artifacts. Table 8 lists the member fields considered for the class TraceLink. A trace link is created between two TracedItems. i.e. source and destination. The field traceType represents the trace relationship type between the two TracedItems. For example, traceType can be “implements”, “verifies”, “satisfies” etc depending on the source and destination artifacts. The other member is status which represents the current status of the trace link. If either one or both of the artifacts (TracedItems) involved in the trace link is updated, then the status of the TraceLink will become “suspect” from the default state “non-suspect”.

Table 8: Fields of Trace Link

Fields	Description
source	TracedItem from which the trace links are created
destination	TracedItem to which trace links are created
traceType	The relationship type
status	The status of the trace link (Status can be either suspect or non-suspect)

The below Figure 21, depicts the relationship between the two classes of TIM i.e. between TracedItem and TraceLink. The TracedItem involved in a TraceLink can either be a destination (represented as is_destination) or source (represented as is_source).

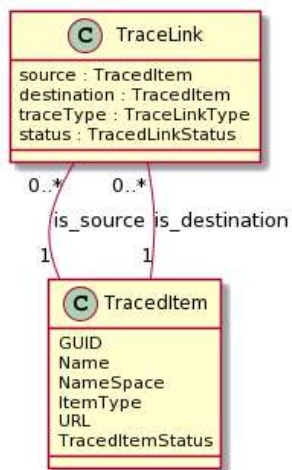


Figure 21: Association of TracedItem and Trace link

Also, each of the TracedItem can have zero or many links to different artifacts. And every TraceLink has one source and one destination TracedItem.

The next Figure 22, shows the allowed trace relationships (TraceLinkType) between source and destination TracedItems. The allowed different TraceLinkTypes are as follows:

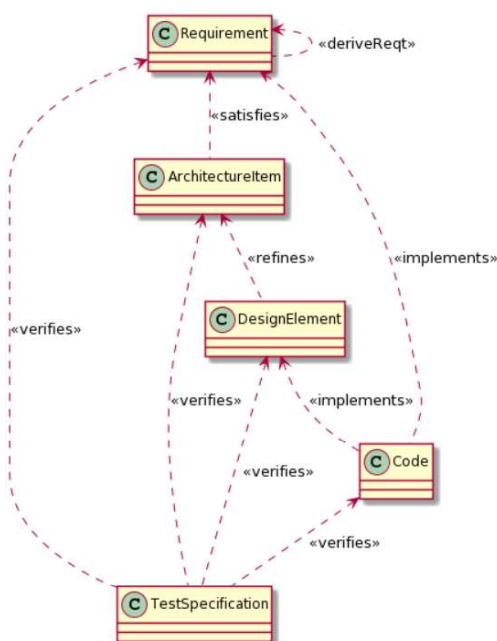


Figure 22: Trace link types between TracedItems

Requirements can be linked to other requirements using the “deriveReq” relationship. Architecture elements are linked to requirements using “satisfies” relationship. DesignElement is linked to ArchitectureItem by the relationship type “refines”. Similarly,

CodeElement can be linked to DesignElement and/or Requirements using “implements”. Finally, all kinds of TestSpecification are linked to the other artifacts using “verifies” relationship.

4.2.2 Features of the Solution

Based on the results of the systematic mapping study and expert interview, the main features of the tool TraceGen were decided. The prototype tool was developed to have three main features. Below Figure 23 shows the three main features (Creation, Usage, and Maintenance) along with the sub-features supported in the tool.

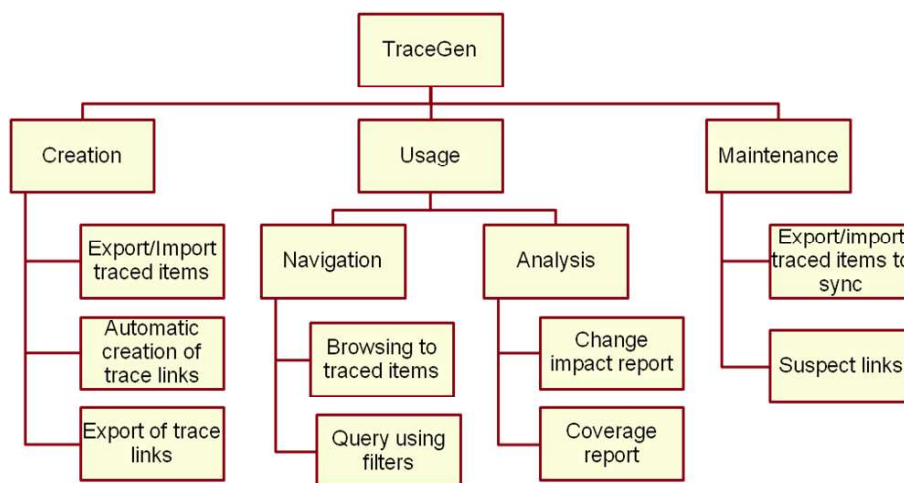


Figure 23: Features supported in TraceGen

Creation: As the name suggests, this feature helps in creating trace links between artifacts. The artifacts that must be traced (TracedItem) are exported/imported between different tools along with the selected fields as per TIM (Table 7) for the creation of trace links. The links are then created between artifacts in either Enterprise Architect or Doxygen tools. The trace relationships (TraceLink) are created as per TIM (Table 8) and then exported from the tools to have the visualization in Microsoft Excel and Power-BI.

Maintenance: The links (TraceLink) created in the tool are updated whenever there are updates in traced artifacts (TracedItem). The maintenance of the current status of TracedItem and TraceLink is done by TraceGen. Synchronization of updated artifacts using export/import ensures the maintenance of trace links. The trace link status will be updated based on the

status(TracedItemStatus) of the traced artifacts. Suspect link functionality is supported to inform the users about the updated links.

Usage: After creating or updating the relationships between artifacts, the trace link information must be analyzed in various ways to have optimal usage. This feature handles the same. TraceGen prepares input (TracedItems and TraceLinks) to Power-BI to have various kinds of traceability reports. The research answers to question RQ2 helped to consider the various features supported based on the uses of traceability. TraceGen supports Power-BI for navigation of artifacts during the analysis of the links, for application of filters, and for the visualization of suspect links.

4.2.3 Use Cases

Mainly two use cases were considered for the development of the tool based on the features supported. The two use cases are the Creation use case and the Maintenance use case. The Usage use case was not considered as the tool Power-Bi was used for the generation and analysis of various kinds of reports.

Creation use case: The creation use case lists the actions involved in the creation of trace links from the beginning of the creation of source and destination artifacts until the generation of traceability reports for analysis. Figure 24 depicts the creation use case. In the diagram, source artifacts (SrcArtifacts) are the artifacts created at the beginning from which the next level artifacts (DstArtifacts) are derived. Hence a trace link has to be created between these two artifacts. Stakeholders can be Manager/Architect/Developer depending on the artifacts that are being considered and the artifacts could be requirements, architectural elements, and implementation elements. For example, consider SrcArtifacts as requirements and DstArtifacts as architectural elements. Then the initial actions involve the creation of requirements and the creation of trace links between the requirements by the Manager (Stakeholder1) in the RM tool (i.e. Src Tool in Figure 24). In the subsequent steps, Architect (Stakeholder2) imports requirements to the Modeling tool (i.e. Dst Tool in Figure 24) and creates the architectural models (DstArtifacts) based on the requirements in Dst Tool. Once the models are created, links between the DstArtifacts and corresponding SrcArtifacts, and links between DstArtifacts are established inside the Dst Tool.

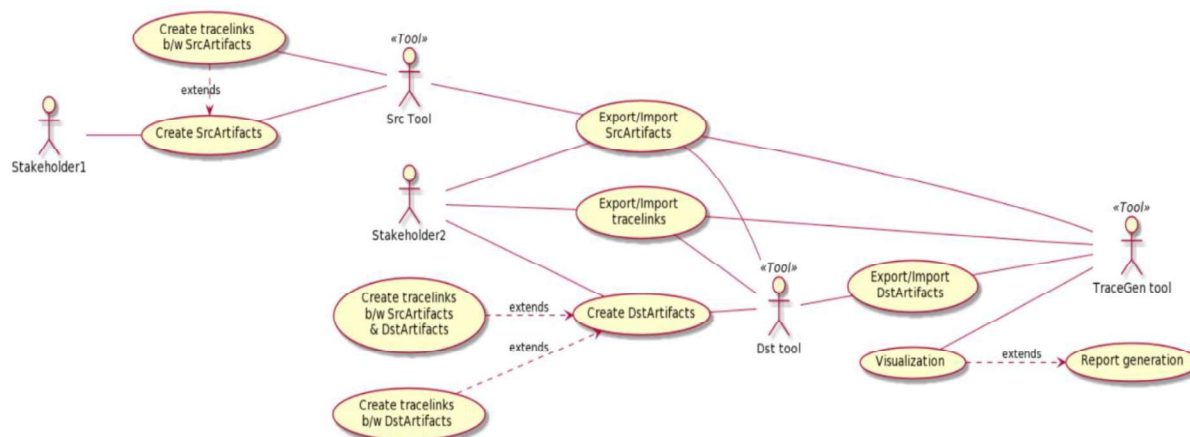


Figure 24: Use case for the creation of trace links

The trace links are then exported to the TraceGen tool which helps in visualizing and analyzing the traceability information. TraceGen tool prepares input data (TracedItem) for Power-BI in CSV format by using the exported source and destination artifacts with the fields mentioned in Table 7. It also prepares the trace information (TraceLink) along with the decided fields (Refer Table 8) as input for Power-BI in CSV format. These data will then be used by Power-BI for the visual analysis by generating various reports.

Maintenance use case: This use case describes the steps when either SrcArtifacts or DstArtifacts or both are updated. Trace links must be revisited and updated as per the changes. Below Figure 25 depicts the maintenance use case by showing the actions that typically happen between stakeholders and the different tools when artifacts get updated. If we continue the above example of requirements being the SrcArtifacts and model elements being the DstArtifacts, when either the requirements or model elements or both get updated, the below actions will take place. When Manager (Stakeholder1) updates requirements in the RM tool (Src Tool), the RM tool updates the trace links. Architect (Stakeholder2) re-imports the updated requirements into the Modeling tool (Dst Tool) updates the model elements (DstArtifacts) if any. The trace links along with SrcArtifacts and DstArtifacts are exported to the TraceGen tool. The TraceGen tool with the help of previous import information (of both source and destination artifacts) and newly synced import information (updated artifacts), generates the trace reports with suspect link information. The status of TracedItems and TraceLink is updated. TraceGen tool with the help of these updated statuses notifies users about suspect links.

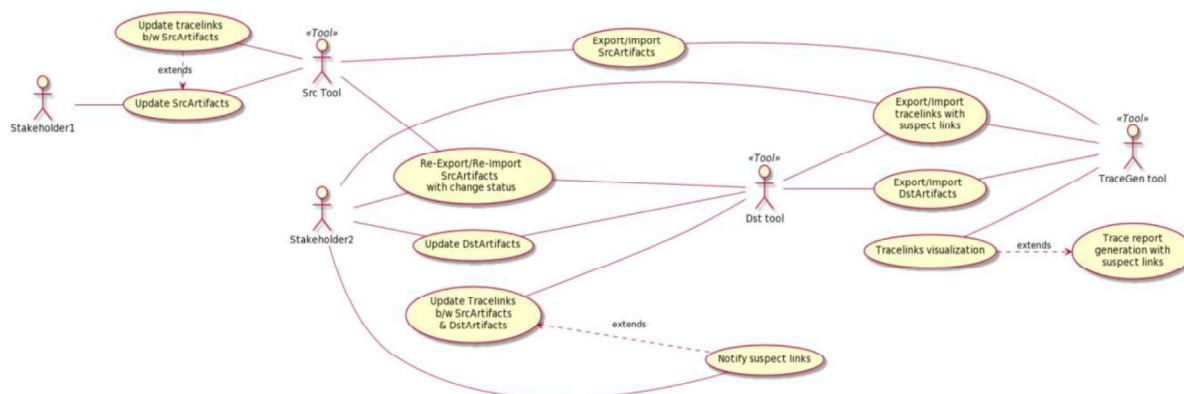


Figure 25: Use case for the maintenance of trace links

As in the creation use case, even during maintenance TraceGen tool prepares updated input data with the fields in Table 7 for Power-BI. The input will be created in CSV format and TraceGen uses the exported source and destination artifacts from the Src Tool and Dst Tool. It also prepares the updated trace information, TraceLink (Refer Table 8) as input for Power-BI in CSV format. These data will then be used by Power-BI for the visual analysis of trace links by generating various kinds of reports along with suspect links.

4.2.4 Traceability Link Creation and Maintenance Process using TraceGen

In this section, the steps followed to achieve traceability between the artifacts across the tools Polarion, Enterprise Architect, and Doxygen using the TraceGen tool are described.

4.2.4.1 Traceability between Polarion and Doxygen using TraceGen

This section describes the process and steps followed to achieve traceability between requirements and implementation elements using TraceGen. Tracelinks are created in Doxygen and TraceGen tool helps to automate most of the steps to reduce manual work. It also helps to recognize the suspect links automatically when artifacts are updated. It generates a traceability matrix in Microsoft Excel and prepares inputs to the Power-BI tool for visualization purposes which helps to analyze the trace links better.

Below are the steps followed for achieving traceability. The steps and process are described based on the use case of trace link creation, usage, and maintenance.

Creation:

1. Export Polarion requirements in Excel and CSV format
2. Use TraceGen to transform Polarion Excel-Export into an anchor file which aids in the creation of traces. This step uses requirement IDs and hyperlinks of requirements to create the file. Below is the snippet of the anchor file generated. Here, REQID-487 and its hyperlink are placed along with the Doxygen command “anchor”. This helps to link the REQID with its hyperlink and can be referred to from other places in the Doxygen document.

A snippet of the Anchor file :

```
/** @page Requirements
 * @section Link
 * @anchor REQID-487
 * <a href="hyperlink"> REQID-487</a> */
```

The anchor file is then placed in the implementation directory where Doxygen is run.

3. The next step is to use custom tags in the Doxygen comments and add Aliases in the Doxygen configuration file.

Custom tags are used to create trace links between requirements and code elements. Using custom tag “implements” along with requirement ID in Doxygen comment creates the trace link between code elements and requirements. Below is the snippet of the Doxygen comment with the custom tag “implements”. It creates trace links of type “implements” between the function “DisplayTemperature” and requirements with ID 487 and 398.

```
/** @implements{@req{487}}
 * @implements{@req{398}} */
void DisplayTemperature(tld sld, TemperatureUnit temp)
```

Adding Aliases in the Doxygen configuration file helps Doxygen to understand the custom tags and to map to the anchor file. Following is the example of aliases that can be added to the configuration file for the above Doxygen comment snippet. This generates a separate page called "Requirement Implementation" in the Doxygen report, displaying all

```
"req{1} = \ref REQID-\1 \"REQID-\1\"
"implements{1} = \xrefitem implement \"Implements requirement\"
\"Requirement Implementation\" \1"
```

the code elements with their links to requirements.

4. When Doxygen is run, the trace links are generated as part of the Doxygen report as mentioned above. TraceGen Tool exports these trace links information into Excel to have a matrix report.
5. The source and destination TracedItems along with the TraceLinks information (chapter 4.2.1) are also prepared by the TraceGen tool in CSV format for Power-BI visualization.

Maintenance:

When either requirements or code or both changes, traceability reports can be updated by syncing the artifacts. Following steps are taken for the sync:

1. Export updated Polarion requirements in CSV and Excel format
2. TraceGen tool uses previously exported data (Step 1 of Creation) and newly updated ones (Step 1 of Maintenance), to update the status of TracedItem (updated, added, deleted, and unchanged). This status is used to highlight in the traceability report for suspect links. The same applies to the code elements as well. Hence TraceGen tool supports bi-directional traceability.
3. When Doxygen is run, the trace links are updated as part of the Doxygen documentation (HTML report). Updated links are highlighted. TraceGen Tool exports these updated trace links information into Excel to have the matrix report. Links generated from the updated artifacts will be shown as a suspect in the report.
4. The updated source and destination TracedItems along with the updated TraceLinks (chapter 4.2.1) are prepared by the TraceGen tool in CSV format and are used for Power-BI visualization.

Usage:

The different visualizations supported are as follows: A list in the Doxygen web report, traceability matrix in Excel report, and various reports like traceability table and matrix with drill features, change impact report, coverage reports in Power-BI. For navigation of Polarion requirements from reports, exported hyperlink information is used. And for Doxygen code elements, TraceGen generates an HTML link to each of the code elements based on the unique identifier created by Doxygen during document generation. Also, Internet Information

Services (IIS) web server is configured to serve the static Doxygen HTML files. Hence these static Doxygen HTML files are used for the navigation of the code elements from the reports.

4.2.4.2 Traceability between Polarion and Enterprise Architect using TraceGen

This section describes the process and steps followed to achieve traceability between requirements and architectural elements using TraceGen. Tracelinks are created in Enterprise Architect and the TraceGen tool helps to automate most of the steps to reduce manual work. It also helps to recognize the suspect links automatically when artifacts are updated. It generates a traceability matrix in Microsoft Excel and prepares inputs to the Power-BI tool for visualization purposes which helps to analyze the trace links better.

Below are the steps followed for achieving traceability between them. The steps and process are described based on the use case of trace link creation, usage, and maintenance

Creation:

1. Export Requirements from Polarion in CSV and Excel format. The required fields of requirements are imported to EA in CSV format. User-defined tag values for requirements are created in EA. These are used for mapping requirement fields that are not present in EA by default. In this case, tag values are space holders to store hyperlinks and the status of requirements. Also, EA assigns a unique GUID for the imported requirements.
2. Create trace links between EA elements and imported requirements in Enterprise Architect using Requirement diagram.
3. Generate Relationship matrix in Enterprise Architect
4. Export the matrix in CSV format and this information is used by TraceGen to generate traceability matrix in Excel and to prepare TraceLink input to Power-BI for other visualization.
5. Export EA elements in CSV format which is used by TraceGen to support bi-directional traceability
6. Export EA HTML reports for supporting the navigation to EA elements from traceability reports

7. TraceGen tool uses the exported CSV files of steps 1 and 5 to prepare source and destination TracedItems as inputs to Power-BI. It uses the HTML report of step 6 to generate hyperlink information to architectural elements which are used for navigation.

Maintenance:

When either requirements or code or both changes, traceability reports can be updated by syncing the artifacts. Following steps are taken for the sync:

1. Requirements and architectural elements are re-exported.
2. As requirements are assigned a unique GUID in EA, subsequent imports in EA update the existing requirements rather than recreating them. TraceGen tool uses previously exported requirements data and newly updated ones, to update the status of TracedItem (updated, added, deleted, and unchanged). The updated requirements are highlighted in the Requirement diagram (created in step 2 of the Creation use case) using the user-defined tag “status”.
3. Export the updated matrix of EA in CSV format and this information is used by TraceGen to update the traceability matrix in Excel. TraceGen tool uses previously exported architectural elements and newly updated ones, to update the status of architectural elements in the trace matrix report. The status update of both requirements and architectural elements by TraceGen helps to highlight “suspect” links in the trace matrix report.
4. The updated source and destination TracedItems (from step 1 of Maintenance) along with the updated TraceLinks (chapter 4.2.1) are prepared by the TraceGen tool in CSV format and are used for Power-BI visualization.

Usage:

The different visualizations supported are Requirements diagram in EA, Traceability matrix in Excel report, and various reports like traceability table and matrix with drill features, change impact report, and coverage reports in Power-BI. For navigation of Polarion requirements from reports, exported hyperlink information from step 1 of both creation and maintenance use cases are used. EA HTML files exported in step 6 of the creation use case and step 1 of the maintenance use case are used for navigation to EA elements from reports.

They are obtained from the Standard HTML Web report feature of EA. IIS web server can also be configured to serve these static Enterprise Architect HTML files. EA protocol can also be used to access EA elements directly from the reports.

4.2.4.3 Traceability between Enterprise Architect and Doxygen using TraceGen

This section describes the process and steps followed to achieve traceability between architecture elements and code elements using TraceGen. Tracelinks are created in Doxygen and TraceGen tool helps to automate most of the steps to reduce manual work. It also helps to recognize the suspect links automatically when artifacts are updated. It generates a traceability matrix in Microsoft Excel and provides inputs to the Power-BI tool for visualization purposes which helps to analyze the trace links better.

Below are the steps followed for achieving traceability between EA and Doxygen. The steps and process are described based on the use case of trace link creation, usage, and maintenance.

Creation:

1. Export architectural elements from Enterprise Architect using CSV export. Architectural elements are also exported to Standard HTML Web reports. IIS(Internet Information Services) web server is configured to serve these static Enterprise Architect HTML files. Use TraceGen to transform EA CSV and HTML reports information into an anchor file. This step uses architecture elements name and GUID to create a hyperlink for each of the elements. This information is used to create an anchor file. Below is the snippet of the anchor file generated. Here, EA-elementxyz and its hyperlink are placed along with the Doxygen command “anchor”. This helps to link the EA-elementxyz to its hyperlink and can be referred to from other places in the Doxygen document.

Snippet of Anchor file:

```
/** @page ArchitecturalElements
 * @section Link
 * @anchor EA-elementxyz
 * <a href="hyperlinktoEAElement"> EA- elementxyz </a> */
```

The anchor file is then placed in the implementation directory where Doxygen is run.

The next step is to use custom tags in the Doxygen comments and add Aliases in the Doxygen configuration file.

Custom tags are used to create trace links between architectural elements and code elements. Using custom tag “refines” along with architectural elements name in Doxygen comment, creates the trace link between code elements and architectural elements. Below is the snippet of the Doxygen comment with custom tags. It creates a trace link between the function “ShowHumidity” and architectural elements “elementxyz” and “elementabc”.

```
/** @refines{@ae{ elementxyz }}
 * @refines {@ae{ elementabc}} */
void ShowHumidity(tld sensorId, float value, char *unit)
```

Adding Aliases in the Doxygen configuration file helps to understand the custom tags and to map to the anchor file. Following is the example of aliases that can be added to the configuration file for the above Doxygen snippet.

```
"ae{1} = \ref EA-\1 \"EA-\1\"
"refines{1}=\xrefitem refines \"Refines ArchitecturalModel\" \"AE Refinement\" \1"
```

This generates a separate page called "AE Refinement" in the Doxygen report, displaying all the code elements with their links to architectural elements.

2. When Doxygen is run, the trace links are generated as part of the Doxygen report as mentioned above. TraceGen Tool exports these trace links information into Excel to have a matrix report.
3. The source and destination TracedItems along with the TraceLinks information (chapter 4.2.1) are also prepared by the TraceGen tool in CSV format for Power-BI visualization.

Maintenance:

When either architectural elements or code or both changes, traceability reports can be updated by syncing the artifacts. Following steps are taken for the sync:

1. Re-export architectural elements in CSV and HTML format
2. TraceGen tool uses previously exported data and newly updated ones, to update the status of TracedItem (updated, added, deleted, and unchanged). This status is used to highlight

suspect links in the traceability report. The same applies to the code elements as well. Hence TraceGen tool supports bi-directional traceability.

3. When Doxygen is run, the trace links are updated as part of the Doxygen documentation (HTML report). Updated links are highlighted. TraceGen Tool exports the updated trace links information into Excel to have the matrix report. Links generated from the updated artifacts will be shown as a suspect in the report.
4. The updated source and destination TracedItems along with the updated TraceLinks information (chapter 4.2.1) are generated by the TraceGen tool in CSV format and are used for Power-BI visualization.

Usage:

The different visualizations supported are a list in the Doxygen web report, traceability matrix in Excel report, and various reports like traceability table and matrix with drill features, change impact report, coverage reports in Power-BI. For navigation of architectural elements from reports, EA HTML files are obtained from the exported Standard HTML Web report (step 1 of both creation and maintenance use case). IIS web server is configured to serve these static Enterprise Architect HTML files. Navigation to a page for a specific diagram or element can be achieved by specifying the appropriate GUID (which is obtained from CSV export). TraceGen tool maps each architectural element with corresponding HTML pages. TraceGen also generates an HTML link to each of the code elements based on the unique identifier created by Doxygen during document generation. IIS web server is also configured to serve the static Doxygen HTML files. Hence these static Doxygen HTML files are used to navigate to the code elements from the reports.

4.2.5 Traceability Visualization in Power-BI

Based on the findings for research question R2 (How and when the trace links are used in industrial practice?), various possibilities of representing the traceability information were explored. The matrix way or list way of representing links may not be useful for every stakeholder. Hence Power-BI, a business analytics service by Microsoft is used to generate various reports. Power-BI is used for the analysis of TraceLink data and TracedItem data (refer to chapter 4.2.1) prepared from the TraceGen tool. Power-BI pulls the data together and processes it to turn into intelligible insights by generating charts and graphs.

A template for the report can be created by defining the relationship between the TracedItems and TraceLinks and by having various graphs, charts depending on the project needs. This report template can then be reused again by refreshing the data as and when the TraceLinks and TracedItems are updated. Refer to Appendix 7.4 for various kinds of visual reports generated in Power-BI with the help of provided data (TracedItems and TraceLinks).

Key benefits of using Power-BI: Power BI Desktop is free of cost and a huge amount of data can be processed by Power-BI with no limits. Power BI compresses each data set effectively before loading it into memory, hence occupies less space. Personalized reports can be created based on project/stakeholders' needs. Using features like drill down and drill up, data can be filtered and reports of only the required data can be viewed. Hence it helps in managing a huge number of traces. Drill through feature helps to navigate from one report to another report using the relationship created between them. Hence can be used for indirect tracing.

4.2.6 Analysis and Results

For the demonstration of feasibility, a practical example is used. BCON, the virtual company introduced in the master's thesis of Zurbuchen (Zurbuchen, 2014), is utilized for this purpose. BCON is characterized as a company with a long-lasting competence in building control. About two dozens of engineers engage in several teams in the development and maintenance of the products. The company offers a product line engineering project called weather station and the same is used for establishing traceability. For this project, the system and software requirements were created and maintained in Polarion and the corresponding system and software architecture models were developed in Enterprise Architect. The code for this project was implemented as a VC++ project and the tool Doxygen was used for generating design documentation from the source code. Hence the tools used in the BCON project matches with the toolchain supported by TraceGen. Also as per chapter 2.3.4, to comply with ASPICE, bi-directional vertical traceability must be achieved between the requirements, architectural models, design, and implementation artifacts present on the left side of the V model. And horizontal bidirectional traceability must be established between the test artifacts on the right side of the V model and the corresponding artifacts on the left side. The latter was not considered for the feasibility study analysis. However, to demonstrate vertical traceability using TraceGen, the BCON project was selected.

For the creation and maintenance of trace links between the artifacts, the TraceGen tool consisting of Python scripts and batch scripts is placed in parallel to the system engineering and software engineering artifacts. The artifacts are exported from the respective authoring tools and placed in the system or software folders by following the steps mentioned in chapter 4.2.4. The main batch file placed in the TraceGen folder is executed with arguments based on the selection of artifacts. For example, to generate trace links between system requirements and system architectural models, the script to be used is: “TraceGen.bat SYS REQ_ARCH generate” and to create trace links between software requirements and implementation elements, the script to be used is: TraceGen.bat SW REQ_IMPL generate. Similarly, trace links are generated across the artifacts that are present on the left side of the V-model of the ASPICE as per Figure 6. To update the trace links when artifacts are modified, the same script is used but with “update” as its last argument. For example, “TraceGen.bat SYS REQ_ARCH update”. This updates the previously created trace links between system requirements and system architectural models.

The first step is to establish trace links between system requirements maintained in Polarion and system architecture models present in Enterprise Architect (Refer to chapter 4.2.4.2). Requirement diagrams are used to link between imported requirements and model elements. Because of the number of requirements and the presence of various types of model elements, more than one requirement diagram is created. Each diagram consisted of model elements that are of the same type. For example, all the use cases are traced to respective requirements in one requirement diagram. For the next requirement diagram, trace links between requirements and the block diagrams (used to describe the sensor component of the weather station) are created. Figure 26, shows the requirements with ID 398, 403, 409, and 405 that are imported from the Polarion project of BCON into the corresponding EA’s system architect model project in the folder 01_requirements. The 4 sensor block diagrams (temperature sensor, air humidity sensor, airpressure sensor, and wind sensor) present in the model are traced to corresponding requirements in the requirement diagram “requirement_SmartWeatherSensor”. Also, it can be noticed that each requirement in the requirement diagram has tag values displayed. The tag “updateReq” is used to show the status of the requirements and the URL helps to navigate to Polarion requirements with a single click.

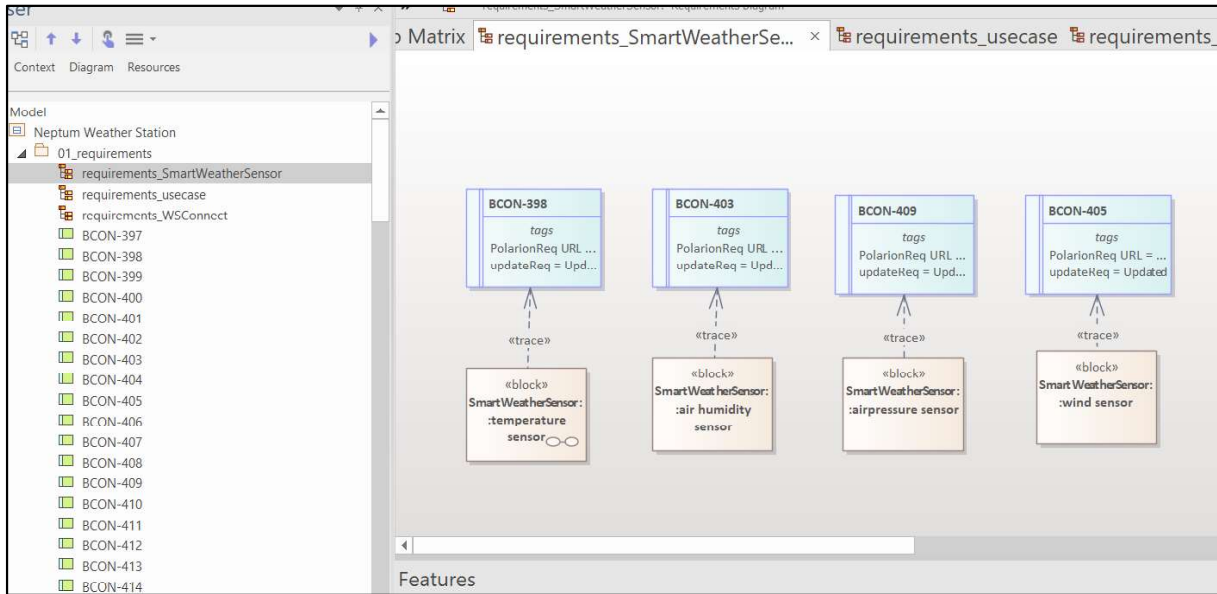


Figure 26: Requirement diagram containing traces

Finally, all the requirements diagrams created in EA are exported in CSV format to create a traceability matrix and to provide input to Power-BI for other visualization.

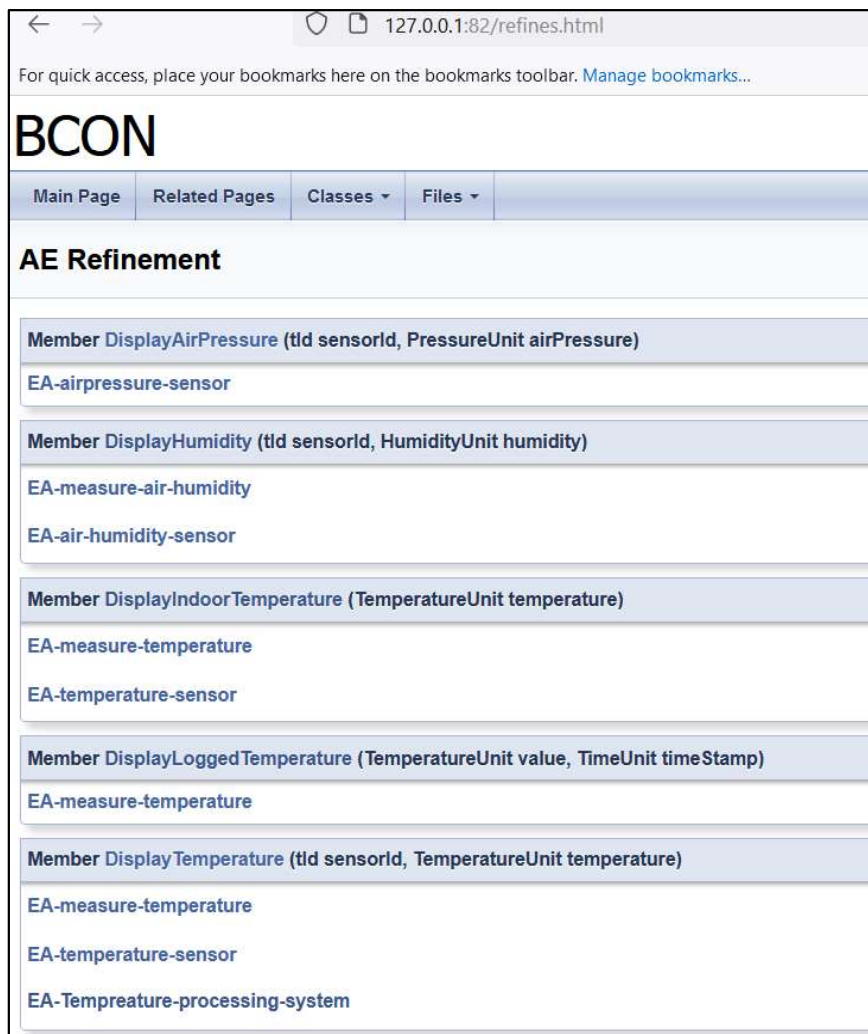
In the next step trace, links between system requirements and software requirements are created in the Polarion project of BCON using traceability link functionality. Below Figure 27 shows the traceability matrix generated in the Polarion BCON Demo Weather Station project. System requirements are depicted in the rows and software requirements in columns. For example system requirement BCON-409 is linked to software requirement WS-54.

The screenshot shows the Polarion interface for the 'BCON Demo Weather Station' project. On the left is a navigation sidebar with options like Home, Feature Model, Master Specifications, Variants, Work Items, Documents & Pages, Default Space, Features, and Index. The main area displays a traceability matrix with system requirements (BCON-397 to BCON-413) on the rows and software requirements (WS-52 to WS-63) on the columns. A grid of cells shows the relationships, with some cells containing small icons indicating the type of link. For example, BCON-409 is linked to WS-54.

Figure 27: Traces between requirements in Polarion

For the next step, traces between software requirements and software architecture are developed using requirement diagrams in the same process as mentioned above (Figure 26). However, traces between software requirements and system architecture are not established directly. As links are already created between system requirements and software requirements, and between system requirements and system architecture, indirect links can be established between software requirements and system architecture. Power-BI's cross-report drill through feature can be used for the same.

The next step is to establish trace links between software architecture and software detailed design. The detailed design is created using Doxygen. As mentioned in chapter 4.2.4.3, the trace links are created using the custom Doxygen comments. The Doxygen report is created along with the trace link information as shown below.



The screenshot shows a web browser window displaying a Doxygen report. The browser address bar shows '127.0.0.1:82/refines.html'. The page title is 'BCON'. Below the title, there are navigation tabs: 'Main Page', 'Related Pages', 'Classes', and 'Files'. The main content area is titled 'AE Refinement' and lists several members with their corresponding EA (Elementary Architecture) elements:

Member	EA Elements
Member DisplayAirPressure (tId sensorId, PressureUnit airPressure)	EA-airpressure-sensor
Member DisplayHumidity (tId sensorId, HumidityUnit humidity)	EA-measure-air-humidity EA-air-humidity-sensor
Member DisplayIndoorTemperature (TemperatureUnit temperature)	EA-measure-temperature EA-temperature-sensor
Member DisplayLoggedTemperature (TemperatureUnit value, TimeUnit timeStamp)	EA-measure-temperature
Member DisplayTemperature (tId sensorId, TemperatureUnit temperature)	EA-measure-temperature EA-temperature-sensor EA-Temperature-processing-system

Figure 28: Trace link information in the Doxygen web report

In Figure 28, the trace list showing the traces between design elements and software architecture models. For example, DisplayHumidity function refines the architecture models EA-measure-air-humidity and EA-air-humidity-sensor.

As Doxygen comments generate design documentation with a one-to-one mapping between code elements and corresponding design units, trace link creation between them is not needed. As the design elements and code elements have the same name, implicit traceability is achieved. And the tracing requirement between software requirements and code elements is established by tracing software requirements and design elements (chapter 4.2.4.1) as there is a one-to-one mapping between design elements and code elements. Hence with all the above-mentioned steps, bi-directional traceability was established between all the artifacts of the BCON project as specified by ASPICE (vertical traceability) with the help of TraceGen.

The next step to analyze is the maintenance of trace links. The trace links maintenance is done as mentioned by sub-sections “Maintenance” in chapter 4.2.4 depending on the artifacts in consideration. In Figure 29 below, the matrix represents the trace links between the software architecture model and design elements with the suspect link marked as “x?”. Suspect links represent that the trace links have been updated due to updates in artifacts. As the architecture model, EA-measure-air-humidity is updated, its corresponding trace links with the design

		DisplayAirPressure	DisplayHumidity	DisplayIndoorTemperature	DisplayLoggedTemperature	DisplayTemperature	DisplayWindDirection	DisplayWindSpeed	GetHumidityString	GetPressureString	GetTemperatureString	_weather_station_trendline	ShowAirPressure	ShowHumidity	ShowTemperature
1		Updated	Updated	Updated	Updated	Updated	Updated	Updated	Updated	Updated	Updated		Updated	Updated	Updated
2	EA-measure-values														
3	EA-display-values														
4	EA-measure-temperature			x?	x?	x?							x?	x?	x?
5	EA measure air humidity	← updated	x?									x?			x?
6	EA-inform-User-about-weather-														
7	EA-WeatherStationConnect														
8	EA-WeatherStationConnect-Basic														
9	EA-WeatherStationConnect-Pro														
10	EA-SmartWeatherSensor-Basic														
11	EA-SmartWeatherSensor-Trendline														
12	EA-SmartWeatherSensor-Pro														
13	EA-SmartWeatherActuator														
14	EA-SmartWeatherApp														
15	EA-SmartWeatherApp-Basic														
16	EA-SmartWeatherApp-Pro														
17	EA-SmartWeatherGateway														
18	EA-Neptum-Weather-Station-														
19	EA-SmartWeatherSensor														
20	EA-temperature-sensor			x?		x?					x?	x			
21	EA-Sensor-data-transmission-														
22	EA-power-supply														
23	EA-communication-system														
24	EA-function-control														

Figure 29: Trace links maintenance in traceability matrix report

elements DisplayHumidity, weather_station_trendline, and ShowHumidity are marked as suspects. Similarly, suspect links are highlighted in Doxygen web reports as well.

To have a complete traceability picture, Power-BI visualization reports were used. The below Figure 30 represents the snapshot of the trace tree established for the BCON project.

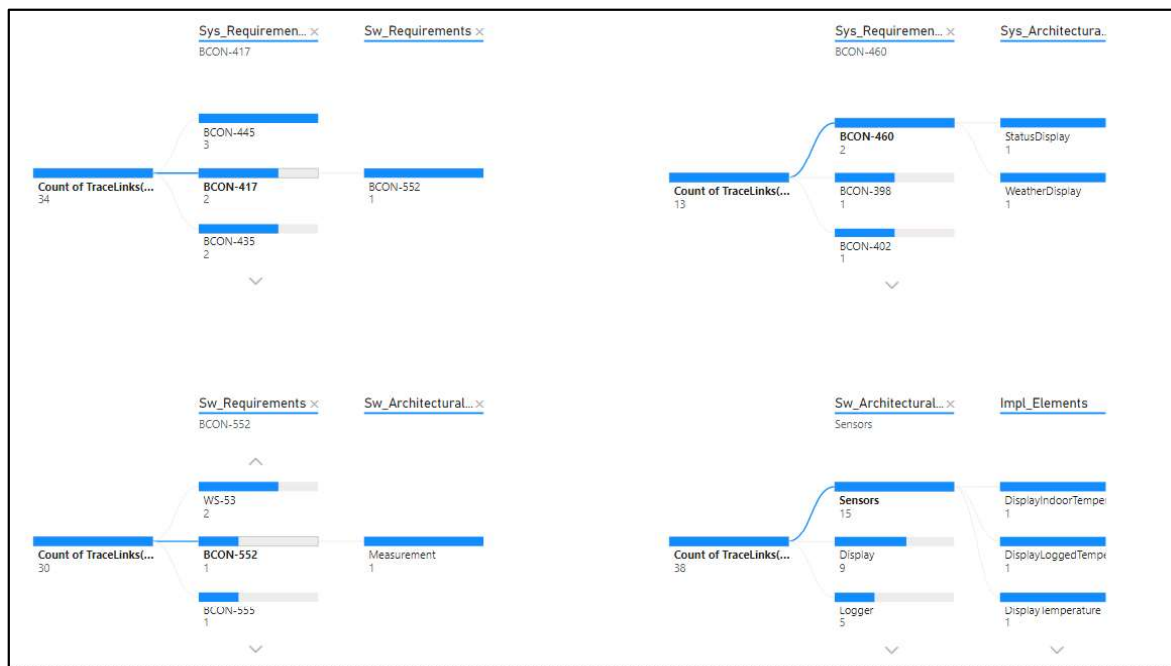


Figure 30: Trace tree report in Power-BI

The report has 4 sub-sections. The upper-left corner represents the trace tree for system requirements and software requirements. The upper-right corner is the trace tree for system requirements and system architecture. Similarly, the trace tree in the left-lower corner represents the traces between software requirements and software architectural elements and the right-lower one represents the trace links between software architectural elements and design/implementation elements. Each of the trace trees can be focused to have a complete list of traces using the focus button. Clicking each of the artifacts shows the linked destination artifacts. For example, the selected system requirement BCON-417 was linked to software requirement BCON-552 in the upper-left report. Similarly, on the right-lower report, the software architectural element Sensors can be seen linked to 15 design elements. Clicking on focus mode and the up and down arrows helps for the navigation of all the traced elements.

The other type of visualization helpful for analysis is the trace matrix report in Power-BI. The visualization is more helpful for analysis than the normal matrix generated in Microsoft Excel

because of drill-down, drill-up, and expand hierarchy features provided by Power-BI. In Figure 31, a trace matrix was formed using trace links between software architecture model elements and implementation elements. Using the drill-down and drill-up features insights to the next-level hierarchies of data can be obtained. For example, the rows and columns represent the TracedItems with fields mentioned in Table 7. The rows have been drill down up to three levels of hierarchies and hence three levels of fields have been represented. i.e. the type of the architectural model, the status of the model, and model element names. For example, the model elements like Comperator, Display, etc are of type Package and the status of these model elements is “Updated”. Similarly in the column, implementation elements have also been represented in three levels. i.e. All the code elements shown in the below diagram are of type function and the first 8 functions have not been updated, but the next functions from the function DisplayIndoorTemperature have been updated. Hence it is easy to

Type	function								Updated		
TracedItemStatus	Unchanged										
Type	GetCurveTemperatureString	GetTemperatureString	IndoorTemperatureDisplay	ShowTemperaturePointProxy	ShowTemperatureProxy	TemperatureCurveDisplay	WeatherDisplay	WindProBasic	DisplayIndoorTemperature	DisplayLoggedTemperature	DisplayTemperature
Package	-										
Unchanged	-										
Updated	-										
Comperator	-	-	-	x?	x?	x?	-	-	-	-	-
Display	-	-	x?	-	-	x?	x?	x?	x?	x?	x?
Logger	x?	-	-	x?	-	x?	-	-	-	x?	-
Power Supply	-	-	-	-	-	-	-	-	-	-	-
Sensors	x?	x?	x?	x?	x?	x?	x?	-	x?	x?	x?
Units	-	-	-	-	-	-	-	-	-	-	x?

Figure 31: Trace matrix with drill features of Power-BI

analyze the reason behind suspect links based on the artifact’s status. Even the hyperlink to elements was made part in the next hierarchical level, hence expanding down to the next level helps for the navigation of elements.

Indirect trace link establishment between system architecture and software requirements is explained with the below Figure 32 representing the trace links in the table (list) format. The left table represents the trace list between system requirements and system architecture elements and the right table represents the trace list between system and software requirements. To find the trace links between system architecture model elements and software requirements (which is one of the compulsory links to be established as per

ASPICE), an indirect tracing method can be used. Selecting the interested system architectural elements in the left table results in highlighting the corresponding traced system requirements. This in turn selects the same highlighted system requirements in the right table which in turn results in highlighting the corresponding traced software requirements. Hence with this indirect trace links between system architecture elements and software requirements are established.

TraceLinks between System Requirements and System Architecture		TraceLinks between System and Software Requirements	
Sys_ArchitecturalElements	First TraceLinkStatus	Sys_Requirements	First TraceLinkStatus
Air pressure processing system	X	BCON-451	x
Humidity processing system	?X	BCON-573	x
PowerSupply	?X	WS-58	x
BCON-451	?X	BCON-477	x
standard protocol host	?X	BCON-558	x
Station data transmission system pro	X	BCON-486	x
StatusDisplay	X	BCON-571	x
Temperature processing system	X	Total	x
WeatherDisplay	X		
WeatherDisplay Basic	?X		
BCON-477	?X		
BCON-486	?X		
WeatherDisplay Pro	?X		
WeatherDisplay Trendline	?X		
Windchill processing system	X		
Windspeed processing system	X		
wireless nearfield communication	X		
WLAN transmitter	X		
Total	?X		

Figure 32: Indirect trace link establishment using Power-BI visualization

For example in Figure 32, the system architecture elements PowerSupply and WeatherDisplay Basic were selected. The corresponding highlighted traced system requirements are BCON-451, BCON-477, and BCON-486. The right trace list report selects the same system requirements (that were highlighted in the left table) and highlights corresponding traced software requirements. i.e, BCON-451 are traced to software requirements BCON-573 and WS-58. Similarly, BCON-477 traces to BCON-558 and BCON-486 traces to BCON-571. Hence the system architectural model element PowerSupply is indirectly related to software requirements BCON-573 and WS-58. Similarly, the model element WeatherDisplay Basic is indirectly traced to software requirements BCON-558 and BCON-571.

Refer to Appendix 7.4 for various visualization report types used for the analysis of trace links in Power-BI.

5 Conclusions

This thesis focuses on the real-life experiences of traceability in the industrial context. Though most of the industries have adapted their processes to include traceability, there are still many challenges that are being encountered to establish it successfully. A literature review conducted during the systematic mapping study (in chapter 2) revealed that different elements need to be considered for achieving traceability successfully and they vary from project to project. Elements like traceability criteria, purposes, approaches, tools used are different for different projects, and hence the challenges faced are also different. It was understood that knowing the reasons for the issues faced by knowing the project context and usage context helps in tackling the issues better. For this, different criteria and approaches followed for achieving traceability were compiled along with all possible use cases. Corresponding benefits and drawbacks of the approaches were listed too. Exploring different tools also helped in analyzing their features along with pros and cons. Hence, different methods were used to compile the challenges faced, and parallelly different ways to mitigate them were also explored.

Though the literature review and mapping study provided the basis to understand the problem and probable solutions, it did not provide real insights into the actual problems faced. Hence conducting expert interviews gave a user perspective on the issues faced. Conducting expert interviews also helped in compiling the best practices to follow and pitfalls to avoid during the traceability process (chapter 3).

From both of the above approaches, it was noted that one of the crucial challenges in achieving traceability in MBSE projects is tool support. As traceability is time-consuming and error-prone, tools must be used for automation. However, it must also be cost-effective as not all projects can invest in exclusive traceability tools due to budget constraints. As MBSE projects can use different kinds of authoring tools, the traceability tool must also be configurable to avoid tool breaks. Hence various viable approaches were explored considering a toolchain consisting of Polarion, Enterprise Architect, and Doxygen (chapter 4.1).

Knowing the list of challenges faced and various approaches and tools available with their pros and cons, a solution approach was developed that would aid the traceability process

(chapter 4.2). The developed prototype tool kept the crucial challenge of tool breaks in mind. The tool is configurable as the used exchanging formats (CSV, and HTML) are standard formats supported by most of the tools. The knowledge shared by interviewees on best practices and pit-falls was also considered to approach the features supported by the tool pragmatically. The maintenance of trace links was also considered as one of the main criteria during the tool development to avoid trace decays. Navigation and handling of a large number of links were tackled by using the tool Power-BI.

To evaluate the results of the proposed solution, a feasibility study was conducted by using the prototype tool on the BCON project (Chapter 4.2.6). The project considered was a prototype developed at Fraunhofer IESE using the same toolchain that was considered for the proposed solution. The results of the case study showed that the tool developed can be used for achieving traceability across the toolchain as per ASPICE's traceability requirements.

5.1 Open Issues and Future work

Based on the results of the case study, future work has been identified.

Currently, the tool developed is purely CLI (command line interpreters) based and does not have a GUI (graphical user interface). Hence, developing it into a GUI-based tool can significantly improve the user-time as using commands is very laborious. A graphical user interface makes the tool more reliable as user mistakes can be avoided. This will also improve user satisfaction.

Achieving traceability using the prototype tool is not automated fully. Some of the non-automated steps like exporting artifacts from tools are cumbersome and users can also miss the steps or perform them differently. This results in erroneous output. Hence, the tool can be improved for automation to reduce manual work.

The complete tree of traces from a high-level artifact to multiple low-level artifacts and vice-versa helps to get the complete picture of dependencies for a selected artifact. Currently, this is not fully supported. Power-BI reports can be navigated from one level to the other using the cross-report drill through feature. However, a single report showing a complete tree

of traces (direct and indirect, with multiple levels) when an artifact is selected without the user navigating from one report to the other would be much easier for analysis.

Overall, the thesis demonstrates that it is recommended to continue research in the direction of automation of the traceability process and configuration of the tool keeping the varying needs of projects in mind.

6 Bibliography

- Albinet, A., Boulanger, J.-L., Dubois, H., Peraldi-Frati, M.-A., Sorel, Y., & Van, Q.-D. (2007). Model-based methodology for requirements traceability in embedded systems. *Proceedings of 3rd European Conference on Model Driven Architecture® Foundations and Applications, ECMDA'07*.
- Amar, B., Leblanc, H., & Coulette, B. (2008). A traceability engine dedicated to model transformation for software engineering. *ECMDA Traceability Workshop (ECMDA-TW)*, (pp. 7–16).
- Antonino, P. O., Keuler, T., Germann, N., & Cronauer, B. (2014). A non-invasive approach to trace architecture design, requirements specification and agile artifacts. *2014 23rd Australian Software Engineering Conference*, (pp. 220–229).
- Asuncion, H. U., Asuncion, A. U., & Taylor, R. N. (2010). Software traceability with topic modeling. *2010 ACM/IEEE 32nd International Conference on Software Engineering, I*, pp. 95–104.
- Badreddin, O., Sturm, A., & Lethbridge, T. C. (2014). Requirement traceability: A model-based approach. *2014 IEEE 4th International Model-Driven Requirements Engineering Workshop (MoDRE)*, (pp. 87–91).
- Bailey, J., Budgen, D., Turner, M., Kitchenham, B., Brereton, P., & Linkman, S. (2007). Evidence relating to Object-Oriented software design: A survey. *First International Symposium on Empirical Software Engineering and Measurement (ESEM 2007)*, (pp. 482–484).
- Bashir, M. F., & Qadir, M. A. (2006). Traceability techniques: A critical study. *2006 IEEE International Multitopic Conference*, (pp. 265–268).
- Chen, X., Hosking, J., & Grundy, J. (2012). Visualizing traceability links between source code and documentation. *2012 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, (pp. 119–126).

- Cleland-Huang, J., Hayes, J. H., & Domel, J. M. (2009). Model-based traceability. *2009 ICSE Workshop on Traceability in Emerging Forms of Software Engineering*, (pp. 6–10).
- Dubois, H., Peraldi-Frati, M.-A., & Lakhal, F. (2010). A model for requirements traceability in a heterogeneous model-based design process: Application to automotive embedded systems. *2010 15th IEEE International Conference on Engineering of Complex Computer Systems*, (pp. 233–242).
- Egyed, A. (2001). A scenario-driven approach to traceability. *Proceedings of the 23rd International Conference on Software Engineering. ICSE 2001*, (pp. 123–132).
- Galvao, I., & Goknil, A. (2007). Survey of traceability approaches in model-driven engineering. *11th IEEE International Enterprise Distributed Object Computing Conference (EDOC 2007)*, (pp. 313–313).
- Gates, A. Q., & Mondragon, O. (2002). FasTLInC: a constraint-based tracing approach. *Journal of systems and software*, *63*, 241–258.
- Gayer, S., Herrmann, A., Keuler, T., Riebisch, M., & Antonino, P. O. (2016). Lightweight traceability for the agile architect. *Computer*, *49*, 64–71.
- Gomaa, H., & Hussein, M. (2007). Model-based software design and adaptation. *International Workshop on Software Engineering for Adaptive and Self-Managing Systems (SEAMS'07)*, (pp. 7–7).
- Gotel, O., Cleland-Huang, J., Hayes, J. H., Zisman, A., Egyed, A., Greunbacher, P., & Antoniol, G. (2012). The quest for ubiquity: A roadmap for software and systems traceability research. *2012 20th IEEE international requirements engineering conference (RE)*, (pp. 71–80).
- Grammel, B., & Kastenholz, S. (2010). A generic traceability framework for facet-based traceability data extraction in model-driven software development. *Proceedings of the 6th ECMFA Traceability Workshop*, (pp. 7–14).

- Guo, J., Cheng, J., & Cleland-Huang, J. (2017). Semantically enhanced software traceability using deep learning techniques. *2017 IEEE/ACM 39th International Conference on Software Engineering (ICSE)*, (pp. 3–14).
- Holtmann, J., Steghöfer, J.-P., Rath, M., & Schmelter, D. (2020). Cutting through the Jungle: Disambiguating Model-based Traceability Terminology. *2020 IEEE 28th International Requirements Engineering Conference (RE)*, (pp. 8–19).
- IEEE. (1984). Guide to Software Requirements Specification *Piscataway: IEEE Press*.
- IEEE. (1990). 610.12-1990 - IEEE Standard Glossary of Software Engineering Terminology.
- Jaber, K., Sharif, B., & Liu, C. (2013). A study on the effect of traceability links in software maintenance. *IEEE Access, 1*, 726–741.
- Jiang, H.-Y., Nguyen, T. N., Chen, X., Jaygarl, H., & Chang, C. K. (2008). Incremental latent semantic indexing for automatic traceability link evolution management. *2008 23rd IEEE/ACM International Conference on Automated Software Engineering*, (pp. 59–68).
- Kagdi, H., Maletic, J. I., & Sharif, B. (2007). Mining software repositories for traceability links. *15th IEEE International Conference on Program Comprehension (ICPC'07)*, (pp. 145–154).
- Kaiser, C., & Herbst, B. (2015). Smart Engineering for Smart Factories: How OSLC Could Enable Plug & Play Tool Integration. In *Mensch und Computer 2015–Workshopband* (pp. 269–280). De Gruyter.
- Koenigs, S. F., Beier, G., Figge, A., & Stark, R. (2012). Traceability in Systems Engineering—Review of industrial practices, state-of-the-art technologies and new research solutions. *Advanced Engineering Informatics, 26*, 924–940.
- Lago, P., Muccini, H., & Van Vliet, H. (2009). A scoped approach to traceability management. *Journal of Systems and Software, 82*, 168–182.

- Mader, P., Gotel, O., & Philippow, I. (2009). Motivation matters in the traceability trenches. *2009 17th IEEE International Requirements Engineering Conference*, (pp. 143–148).
- Maletic, J. I., Munson, E. V., Marcus, A., & Nguyen, T. N. (2003). Using a hypertext model for traceability link conformance analysis. *Proc. of the Int. Workshop on Traceability in Emerging Forms of Software Engineering*, (pp. 47–54).
- Marcus, A., & Maletic, J. I. (2003). Recovering documentation-to-source-code traceability links using latent semantic indexing. *25th International Conference on Software Engineering, 2003. Proceedings.*, (pp. 125–135).
- Maro, S. (2020). Improving software traceability tools and processes.
- McMillan, C., Poshyvanik, D., & Revelle, M. (2009). Combining textual and structural analysis of software artifacts for traceability link recovery. *2009 ICSE Workshop on Traceability in Emerging Forms of Software Engineering*, (pp. 41–48).
- Meedeniya, D. A., Rubasinghe, I. D., & Perera, I. (2019). Traceability establishment and visualization of software artefacts in devops practice: a survey. *International Journal of Advanced Computer Science and Applications*, *10*, 66–76.
- Nair, S., De La Vara, J. L., & Sen, S. (2013). A review of traceability research at the requirements engineering conference re@ 21. *2013 21st IEEE International Requirements Engineering Conference (RE)*, (pp. 222–229).
- Petersen, K., Feldt, R., Mujtaba, S., & Mattsson, M. (2008). Systematic mapping studies in software engineering. *12th International Conference on Evaluation and Assessment in Software Engineering (EASE) 12*, (pp. 1–10).
- Ramesh, B., & Edwards, M. (1993). Issues in the development of a requirements traceability model (pp. 256-259). IEEE.
- Ramesh, B., & Jarke, M. (2001). Toward reference models for requirements traceability. *IEEE transactions on software engineering*, *27*, 58–93.

- Ramesh, B., Stubbs, C., Powers, T., & Edwards, M. (1997). Requirements traceability: Theory and practice. *Annals of software engineering*, 3, 397–415.
- Regan, G., Mc Caffery, F., Mc Daid, K., & Flood, D. (2013). Medical device standards' requirements for traceability during the software development lifecycle and implementation of a traceability assessment model. *Computer Standards & Interfaces*, 36, 3–9.
- Regan, G., McCaffery, F., McDaid, K., & Flood, D. (2012). The barriers to traceability and their potential solutions: Towards a reference framework. *2012 38th Euromicro Conference on Software Engineering and Advanced Applications*, (pp. 319–322).
- SIG, V. Q. (2017). *Automotive SPICE Process Assessment / Reference Model*.
- Sundaram, S. K., Hayes, J. H., Dekhtyar, A., & Holbrook, E. A. (2010). Assessing traceability of software engineering artifacts. *Requirements engineering*, 15, 313–335.
- Tang, A., Jin, Y., & Han, J. (2007). A rationale-based architecture model for design traceability and reasoning. *Journal of Systems and Software*, 80, 918–934.
- Winkler, S., & von Pilgrim, J. (2010). A survey of traceability in requirements engineering and model-driven development. *Software & Systems Modeling*, 9, 529–565.
- Zurbuchen, G. (2014). A Case Study for Integrated Lifecycle and Variant Management in a SME Context. Kaiserslautern.

7 Appendix

7.1 Mapping of Research Papers based on Classification Scheme

ID	Year	Title	Search String	Author	Research facet	Contribution facet	Product Concept Context Facet
P01	2007	Survey of Traceability Approaches in Model-Driven Engineering	(Traceability OR trace link OR tracing) AND (survey OR overview OR "literatur* review")	(Galvao & Goknil, 2007)	Solution proposal	Method	End to end traceability/ Comparison of tools and techniques
P02	2009	A survey of traceability in requirements engineering and model-driven development	(Traceability OR trace link OR tracing) AND (survey OR overview OR "literatur* review")	(Winkler & von Pilgrim, 2010)	Solution proposal	Method/ Model/Terminology	Traceability concepts
P03	2006	Traceability Techniques: A Critical Study	(Traceability OR trace link OR tracing) AND (survey OR overview OR "literatur* review")	(Bashir & Qadir, 2006)	Evaluation Research	Method	Requirements traceability/ Comparison of tools and techniques
P04	2012	Traceability in Systems Engineering – Review of industrial practices, state-of-the-art technologies and new research solutions	(Traceability OR trace link OR tracing) AND ((challenges OR best practice OR lessons learned) and (industry or industrial))	(Koenigs, Beier, Figge, & Stark, 2012)	Solution proposal	Tool	Challenges of traceability and solutions
P05	2012	The Barriers to Traceability and their Potential Solutions: Towards a Reference Framework	(Traceability OR trace link OR tracing) AND ((challenges OR best practice OR lessons learned) and	(Regan, McCaffery, McDaid, & Flood, 2012)	Evaluation Research	Method	Challenges of traceability and solutions

			(industry or industrial))				
P06	2012	Medical device standards' requirements for traceability during the software development lifecycle and implementation of a traceability assessment model	(Traceability OR trace link OR tracing) AND ((challenges OR best practice OR lessons learned) and (industry or industrial))	(Regan, Mc Caffery, Mc Daid, & Flood, 2013)	Solution proposal	Method	End to end traceability/ Traceability in medical systems
P07	2012	The Quest for Ubiquity: A Roadmap for Software and Systems Traceability Research	(Traceability OR trace link OR tracing) AND ((challenges OR best practice OR lessons learned) and (industry or industrial))	(Gotel, et al., 2012)	Solution proposal	Method/ Model/Tool	Challenges of traceability and solutions
P08	2010	Software Traceability with Topic Modeling	(Traceability OR trace link OR tracing) AND ((challenges OR best practice OR lessons learned) and (industry or industrial))	(Asuncion, Asuncion, & Taylor, 2010)	Evaluation Research	Method/ Model/Tool	End to end traceability/ Comparison of tools and techniques
P09	2010	A model for requirements traceability in a heterogeneous model-based design process	(Traceability OR trace link OR tracing) AND (automotive OR health)	(Dubois, Peraldi-Frati, & Lakhali, 2010)	Validation Research	Method/ Model	Requirements traceability
P10	2009	Model-Based Traceability	(Traceability OR trace link OR tracing) AND (automotive OR health)	(Cleland-Huang, Hayes, & Domel, 2009)	Validation Research	Method	End to end traceability

P11	2010	Assessing traceability of software engineering artifacts	(Traceability OR trace link OR tracing) AND (automotive OR health)	(Sundaram, Hayes, Dekhtyar, & Holbrook, 2010)	Evaluation Research	Method	Requirements traceability/Comparison of tools and techniques
P12	2008	A scoped approach to traceability management	(Traceability OR trace link OR tracing) AND (automotive OR health)	(Lago, Muccini, & Van Vliet, 2009)	Validation Research	Process/Model	End to end traceability/Challenges of traceability and solutions
P13	2020	Cutting through the Jungle: Disambiguating Model-based Traceability Terminology	(Traceability OR trace link OR tracing) AND (automotive OR health)	(Holtmann, Steghöfer, Rath, & Schmelter, 2020)	Philosophical Papers	Terminology	Traceability concepts
P14	2003	Using a Hypertext Model for Traceability Link Conformance Analysis	(Traceability OR trace link OR tracing) AND (automotive OR health)	(Maletic, Munson, Marcus, & Nguyen, 2003)	Solution Proposal	Model	Traceability maintenance
P15	2016	Lightweight Traceability for the Agile Architect	(Traceability OR trace link OR tracing) AND (automotive OR health)	(Gayer, Herrmann, Keuler, Riebisch, & Antonino, 2016)	Solution Proposal	Method	Creation of traceability/Traceability maintenance
P16	2008	A Traceability Engine Dedicated to Model Transformation for Software Engineering	(Traceability OR trace link OR tracing) AND (automotive OR health)	(Amar, Leblanc, & Coulette, 2008)	Solution Proposal	Model/Tool	Creation of traceability
P17	2013	A study on the effect of traceability links in software maintenance	(Traceability OR trace link OR tracing) AND (survey OR overview OR "literatur* review")	(Jaber, Sharif, & Liu, 2013)	Solution Proposal	Model	End to end traceability/Comparison of tools and techniques

P18	2019	Traceability Establishment and Visualization of Software Artefacts in DevOps Practice: A Survey	(Traceability OR trace link OR tracing) AND (survey OR overview OR "literatur* review")	(Meedeniya, Rubasinghe, & Perera, 2019)	Solution Proposal	Method/ Tool	Comparison of tools and techniques
P19	2013	A Review of Traceability Research at the Requirements Engineering Conference	(Traceability OR trace link OR tracing) AND ((challenges OR best practice OR lessons learned) and (industry or industrial))	(Nair, De La Vara, & Sen, 2013)	Solution Proposal	Method/ Tool	End to end traceability/Challenges of traceability and solutions/Comparison of tools and techniques
P20	2012	Visualizing Traceability Links between Source Code and Documentation	(Traceability OR trace link OR tracing) AND (survey OR overview OR "literatur* review")	(Chen, Hosking, & Grundy, 2012)	Solution Proposal	Method	Traceability visualization
P21	2001	Toward Reference Models for Requirements Traceability	(Traceability OR trace link OR tracing) AND ((challenges OR best practice OR lessons learned) and (industry or industrial))	(Ramesh & Jarke, Toward reference models for requirements traceability, 2001)	Solution Proposal	Model	Requirements traceability
P22	2003	Recovering Documentation-to-Source-Code Traceability Links using Latent Semantic Indexing	(Traceability OR trace link OR tracing) AND (survey OR overview OR "literatur* review")	(Marcus & Maletic, 2003)	Solution proposal	Method	Creation of traceability
P23	1997	Requirements traceability: Theory and practice	(Traceability OR trace link OR tracing) AND (model based software engineering)	(Ramesh, Stubbs, Powers, & Edward	Evaluation Research	Model/Method	Traceability concepts

			OR MBSE)	s, 1997)			
P24	2017	Semantically Enhanced Software Traceability Using Deep Learning Techniques	(Traceability OR trace link OR tracing) AND (model based software engineering OR MBSE)	(Guo, Cheng, & Cleland-Huang, 2017)	Solution proposal	Method	Creation of traceability
P25	2006	A rationale-based architecture model for design traceability and reasoning	(Traceability OR trace link OR tracing) AND (model based software engineering OR MBSE)	(Tang, Jin, & Han, 2007)	Solution proposal	Model/Tool	Creation of traceability
P26	2007	Model-based methodology for requirements traceability in embedded systems	(Traceability OR trace link OR tracing) AND (model based software engineering OR MBSE)	(Albini, et al., 2007)	Solution proposal	Method/Tool	Requirements traceability/ End to end traceability
P27	2009	Combining Textual and Structural Analysis of Software Artifacts for Traceability Link Recovery	(Traceability OR trace link OR tracing) AND (survey OR overview OR "literatur* review")	(McMillan, Poshyanyk, & Revelle, 2009)	Solution proposal	Method	Traceability maintenance
P28	2008	Incremental Latent Semantic Indexing for Automatic Traceability Link Evolution Management	(Traceability OR trace link OR tracing) AND (survey OR overview OR "literatur* review")	(Jiang, Nguyen, Chen, Jaygarl, & Chang, 2008)	Solution proposal	Method	Traceability maintenance
P29	2009	Motivation Matters in the Traceability Trenches	(Traceability OR trace link OR tracing) AND (survey OR overview OR "literatur* review")	(Mader, Gotel, & Philippow, 2009)	Evaluation Research	Process	Challenges of traceability and solutions

P30	2007	Mining Software Repositories for Traceability Links	(Traceability OR trace link OR tracing) AND (survey OR overview OR "literatur* review")	(Kagdi, Maletic, & Sharif, 2007)	Solution proposal	Method	Creation of traceability/ Traceability maintenance
-----	------	---	---	----------------------------------	-------------------	--------	--

7.2 Expert Interview Questions

Introduction - Open questions about the background of your company/projects

1. Please provide a brief characterization of your company/projects. #application domain, #standards used, #Software development life-cycle model followed

-

2. May we mention the name of your company in the study?

-

3. What is your organizational role and what part of the organization do you represent?

-

Traceability – Basics

4. What is the purpose of tracing in your project?

-

5. Which artifacts/items do you trace (features, requirements, architecture elements, implementation, test, etc.)?

-

6. When and how do you use the traces?

-

7. Are you facing challenges with traceability?

-

8. Approximately how much effort do you spend in tracing (% of the overall engineering effort)?

-

Traceability - Approach

9. *When do you create trace links? Is it requirements-driven or captured during the transformation from one artifact to the other?*

-

10. *How do you represent the trace links? (e.g. matrix, hyperlinks, graph)*

-

11. *What happens on the change or deletion of artifact and trace link? Is automatic change propagation expected as per trace links creation/deletion?*

-

12. *Is traceability also created and maintained between artifacts developed in different companies? If so, what are specific challenges/approaches to this end?*

-

13. *Any metrics used for measuring the quality of traceability? What does correct traceability mean according to your knowledge?*

-

Traceability - Tools

14. *What are the tools used for the creation and management of the artifacts?*

-

15. *Is a separate tool used to create trace links between artifacts?*

-

16. *Depending on the tools used for the artifacts and trace links, where are the trace links stored? (distributed or central)*

-

17. *Which tool features do you rely on to manage traces?*

-

18. *Do you visualize trace links? How should the trace links be represented to be useful and understandable?*

-

19. *How do you analyze the trace links?*

-

Traceability – MBSE specific

20. *Do you apply model-based systems engineering?*

-

21. *Where does MBSE ease traceability?*

-

22. *Where does MBSE complicate traceability?*

-

23. *Which specific challenges / best practices do you see here?*

-

Wrap Up

24. *Which improvement potential do you see wrt. traceability within your organization?*

-

25. *Which promising initiatives, approaches, etc. do you see wrt. traceability?*

-

26. *Which other question would you have asked in this context?*

-

7.3 Information Collected during Expert Interviews

Sections	Sub-Sections	Data collected
Introduction	Company	System services company
		An automotive supplier
		The automotive first-tier supplier company
	Projects	Consulting services for systems engineering (includes consultation for processes, methods, and tools) and Consultation services in providing project support for systems engineering (toolchain support for requirements engineering and model-based system engineering)

		Development of components/complete power train systems, Engineering service for prototype/software development of hybrid/electric vehicles
		Automated and Autonomous driving systems, projects on sensors, cameras, radar, LIDAR, main ECU part with computation power, vehicle motion control systems(braking, steering), transmission control
	Standards followed	ISO 15288
		ISO 26262 for functional safety
		ISO 15288 for systems engineering
		ASPICE, ISO 26262(safety standard), ISO 27001(security standard)
	Life cycle models	Agile
		V-model
		Both Agile and V-model
	Interviewee role	Project Manager
		Software Engineer
		System architect
Traceability- Basics	Purpose	Validation of requirements
		Verification of requirements and implementation
		Lifecycle coverage analysis
		Tracking rationale
		Change impact analysis
		Validation of artifacts
		To comply with the standard
		To maintain a consistent system
		To check for completeness
		For reusability
	Subject of interest	Between system requirements and software requirements
		Requirements and Architecture models
		Requirements and work packages
		Requirements and design elements
		Requirements and implementation
		Requirements and test cases
		Requirements and test reports
		Group of requirements to features
		Between architectural elements
		Design decisions to requirements
	Design decisions to architectural elements	
	Usage of traces	To check for completeness(quantitative check)
		To check for correctness (qualitative check)
For change impact analysis		
For estimation based on changes		
For review and validation		
For monitoring the progress of the project		
Traceability – Approach	Creation of trace links	After the creation of artifacts
		If dependent on other teams artifacts, after its access
	Representation of trace links and usage	Traceability matrix - to check missing links, for coverage report and maintainability

		Hyperlinks - to navigate between artifacts
		Graphs - for impact analysis, to check for completeness
		Tables - for comparison, for maintainability
		Diagrams(Ex., cake diagram) - for reports to check the completeness
	Maintenance during change/deletion	Suspect functionality to navigate and check the other linked artifacts
		A gateway between Rhapsody-DOORS highlights updated, deleted, and newly added artifacts
	Traceability across teams	Using import/export functionality in ReqIF format
		HIS standard is used for exchanging requirements
		Miro or baselining approach is used for exchanging the requirements
	Metrics for traceability	Traceability matrix is used for quantity checks
		A table view is used for quality checks (for correctness)
		Quality is measured through completeness by counting the links. Can be automated
		Quality is also measured through correctness (reviews are done to verify the contents of traceability)
Traceability tools	Tools used for artifacts development and maintenance	DOORS and PTCIntegrity (For requirements engineering)
		Enterprise Architect and Rational Rhapsody (for architecture modeling and model-based systems engineering)
		Matlab Simulink (For implementation)
		TPT (Testing tool)
		Polarion (for requirements and test cases)
		Cameo systems modeler for architecture models
		DOORS for test specification
	Tools/features used for traceability	ReqIF format is used for import/export of artifacts and links
		OSLC capabilities are being explored for the import/export of artifacts
		In DOORS
		In Rhapsody
		In PTCIntegrity
	Trace link storage	Centrally stored in DOORS/PTC integrity
		A major part in Polarion and fewer links in Cameo
		In DOORS the trace link information is stored in link modules
		In Rhapsody traces between architectural elements are stored as model elements as part of the diagram
	Management of trace links	Creation of traces uses link functionality
		Report functionality is used to create a matrix, bar graphs, etc
		Navigation of traces bi-directionally
		Link modules in DOORS manage links
		Gateway is used to manage bidirectional traces between DOORS and Rhapsody
		Scripts are used to pull and push artifacts between DOORS and Rhapsody

Traceability-MBSE specific	Application of traceability in MBSE	Applied at different maturity levels Level 1: In the easy level, tracing is done from requirements to components and then to interfaces Level 2: Between architectural models
	Pros of traceability in MBSE	A better understanding of the whole system architecture
		Easy analysis and navigation of design rationale using different views
		Identification of reusable elements
		Trace links could be an output of the model transformation
		If a single tool is used for the development of all the artifacts in an MBSE project, then traceability can be created with ease
	Cons of traceability in MBSE	Decision on traceability schema is difficult. Should be well thought
		Granularity level must be carefully considered (not all models should be traced)
		Tool-breaks

7.4 Different Visual Reports used for Traceability in Power-BI

- Trace-list:** Lists represent traceability links for every source TracedItem with linked destination TracedItems in one entry. In Figure 33, two lists are shown. The left list shows the Architectural elements with their traced requirements along with the TracedItemStatus. Similarly, the right list shows the requirements with their traced architectural elements along with their status. For example, the updated architectural

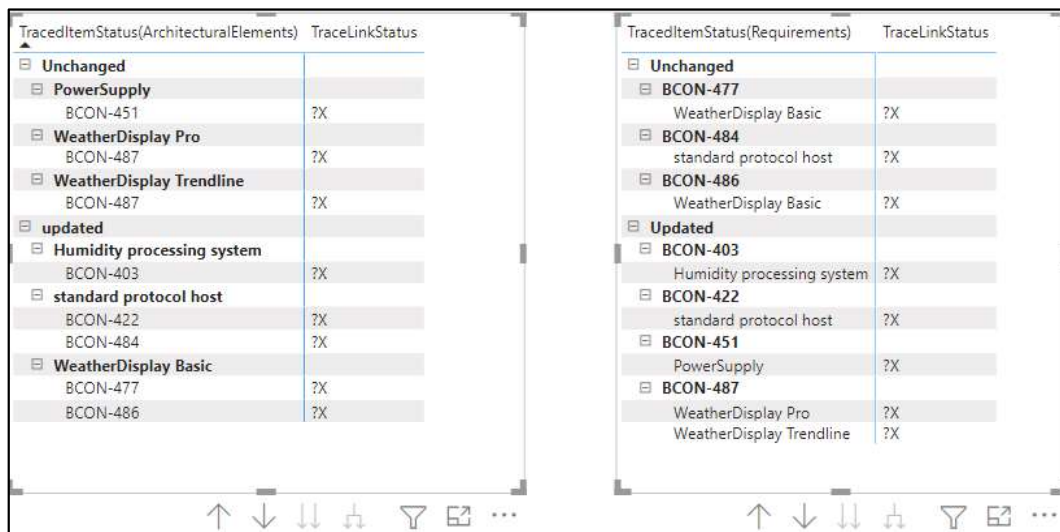


Figure 33: Trace-list representation in Power-BI

elements in the left list are Humidity processing system, standard protocol host, and WeatherDisplay Basic. And the requirements to which standard protocol host is mapped are BCON-422 and BCON-484.

- Trace-matrix:** Table-like representation that maps source TracedItem depicted in columns to destination TracedItem depicted in rows or vice-a-versa. In Figure 34, architectural elements are listed in rows and code elements in the columns. The cells with the mark “x” or “?x” represent the traceability links between the elements present in the corresponding row and column. For example, the architectural element Display is traced to code elements DisplayIndoorTemperature, DisplayLoggedTemperature, and DisplayTemperature, and so on.

Name	DisplayIndoorTemperature	DisplayLoggedTemperature	DisplayTemperature
ADC	-	-	-
AlarmConfigurator	-	-	-
Communicator	-	-	-
Comperator	-	-	-
Display	x?	x?	x?
Example Command Receiver	-	-	-
Example Value Receiver	-	-	-
HAL	-	-	-
Hardware	-	-	-
Interrupts	-	-	-
Logger	-	x?	-
Measurement	-	-	-
Neptun WeatherStation	-	-	-
PL Component	-	-	-
PL Components	-	-	-
Power Supply	-	-	-
Selector	-	-	-
Sensors	x?	x?	x?
SoftwareComponents	-	-	-
Timer	-	x?	-
TransmissionConfigurator	-	-	-
Units	-	-	x?
WeatherStation Package	-	-	-

Figure 34: Trace-matrix representation in Power-BI

Traceability-graph: It is also called a trace tree report. TracedItems are represented as nodes. Nodes are connected by edges if a TraceLink between the source and destination TracedItems exists. Figure 35 represents the graph representation of trace links between architectural elements and implementation elements. For example, the architectural element Logger is linked to five implementation elements highlighted. Similarly, when another architectural element like Sensors is selected, the linked 15 implementation elements will be highlighted in the report.

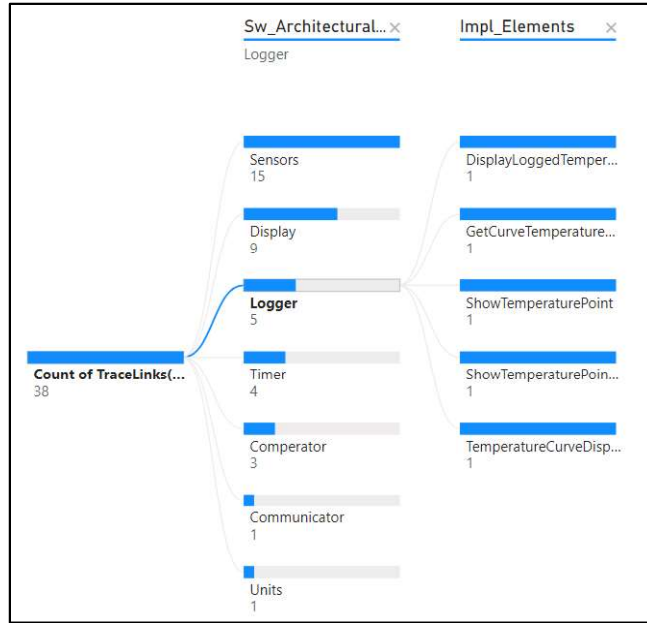


Figure 35: Traceability graph representation in Power-BI

- TraceCoverage report:** This report helps to visualize the total number of artifacts that are traced. Figure 36 below, depicts the coverage report in the form of Donut charts. It provides the visual representation of the percentage of the TracedItems. For example, the first chart represents 84.16% of unlinked architectural elements, and the remaining 15.84% of it is linked to the code elements.

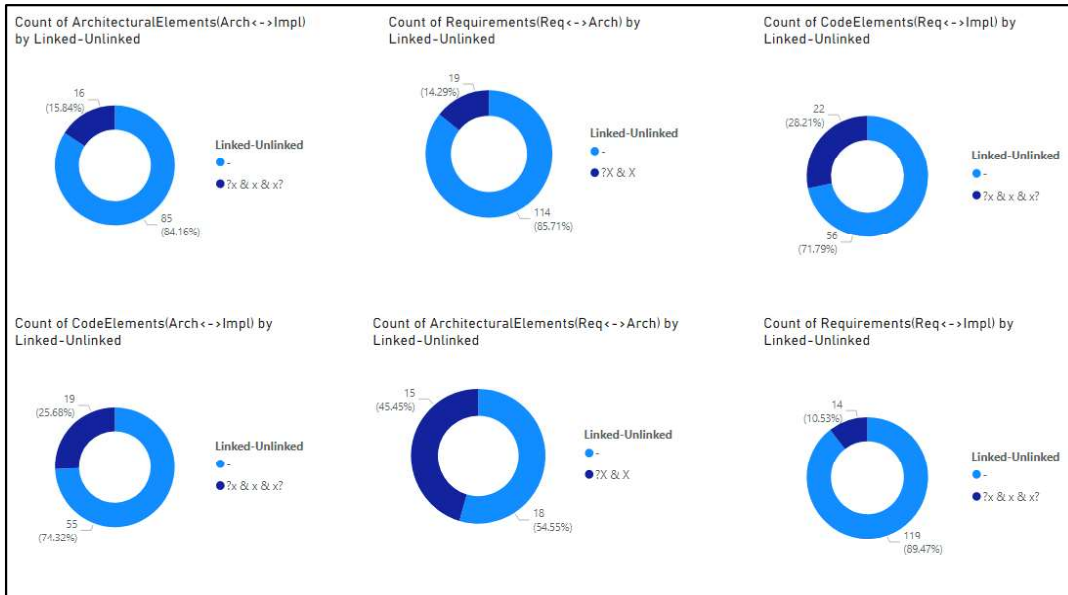


Figure 36: Trace coverage report in Power-BI

- SuspectLinks report:** This report helps to visualize how many of the TraceLinks have suspect links because of some updates in the TracedItems. Figure 37 below, represents the suspect links report in the form of Pie chart. For example, the middle Pie chart depicts that 38.1% of trace links between requirements and architectural elements are suspects, and the remaining 61.9% of trace links are not suspects.

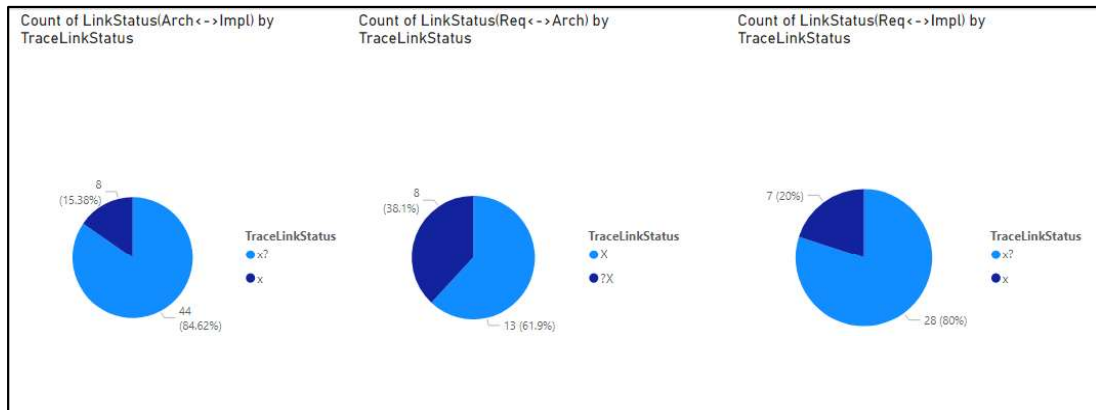


Figure 37: Suspect links report in Power-BI