CHALMERS
UNIVERSITY OF TECHNOLOGY

Functional driven product development using MBSE

Master's Thesis in Product Development

Joseph Leo Arockia Irudayaraj

Department of Industrial and Material Science
Division of Product Development
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2020

Master's thesis 2020

Functional driven product development using MBSE

Joseph Leo Arockia Irudayaraj

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Industrial and Material Science
Division of Product Development
Chalmers University of Technology
Gothenburg, Sweden 2020

Functional driven product development using MBSE
Joseph Leo Arockia Irudayaraj

Supervisor: Roger Persson, CEVT AB
Supervisor: Krister Sutinen, Siemens Digital Industries Software AB
Chalmers Supervisor: Dag Henrik Bergsjö,
Examiner: Dag Henrik Bergsjö, Department of Industrial and Material Science, Chalmers University of Technology

Master's Thesis 2020
Department of Industrial and Material Science
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Printed by [Chalmers Reproservice]
Gothenburg, Sweden 2020

Abstract

In this digitalization era, manufacturing companies are shifting towards automation in order to catch up with the technology needs of industry 4.0. To achieve a smart factory, it requires solving challenges from different domains. The research work focuses on identifying the challenges faced in the existing product development and provides recommendation upon overcoming the challenges by implementing MBSE in practise [1]. The Project was a collaboration between China Euro Vehicle Technology (CEVT) and Chalmers university of Technology. The existing product development activities are carried out in a traditional way which is the document-based activities. The Thesis aims to investigate and provide recommendations about the model-based system engineering approach.

The research work is divided into different phases, where, the first phase focused on understanding the model-based system engineering (MBSE) methodology and the barriers in the exiting product development process. To further understand the problems, several interviews has been held with the stakeholders. The barriers identified played a major role in defining the use case in the later stage of the research work.

The second phase focuses on investigating the MBSE tool i.e., Capella. This phase deals with understanding the architecture behind the modelling techniques in Capella. ARCADIA is the system engineering method which utilizes models with a collaborative, evaluation and exploitation of the architecture. The different stages of the architecture are analysed and the operations performed on each stage are documented. The third phase of the thesis work focuses on performing the product modelling in the MBSE tool. In order to perform the modelling, rear view door mirror subsystem has been chosen as a system of reference. The different model elements corresponding to the Door mirror components are modelled in Capella and the tracelinks are established. In order to implement the requirement integration with the models, System Modeling Workbench (SMW) platform provided by Siemens is utilized. SMW interface is a replica of Capella with some additional integration features. The final phase is to verify MBSE and the model elements against the use cases. A number of product development activities are defined in the form of use cases and its implementations are discussed. The MBSE platform is validated against the use cases and the results are captured. The outcome of the research work is the Proof of concept in SMW for Teamcenter which attempts to eliminate the barriers identified in the existing PD processes. The thesis work concludes with the list of benefits upon implementing MBSE and highlights the changes in the development process from the existing product development process in practise. It also provides certain guidelines and best practise methods to be followed upon implementation of MBSE in a project.

Keywords: Systems Engineering, Capella, System Model Workbench, Model based systems engineering, ARCADIA.

Acknowledgements

I would like to convey my deepest gratitude to Martin Töppner and Roger Persson, CEVT AB for believing in abilities and providing this wonderful opportunity to do a master thesis in Product lifecycle management domain.

I would like to convey my special thanks to Roger Persson, CEVT AB and Krister Sutinen, Siemens Digital Industries Software AB for your continued support and guidance throughout the thesis work. A big thanks to all the CEVT personnel for supporting the thesis work and for spending your valuable time during interview sessions and meetings. A great thanks to my Chalmers supervisor, Dr. Dag Henrik Bergsjö for believing in my abilities and guiding me in this challenging thesis work.

Last but not least, I would like to thank my Family and Friends for your love and support.

Joseph Leo Arockia Irudayaraj, Gothenburg, June 2020

List Of Abbreviations

ARCADIA Architecture Analysis and Design Integrated Approach

AWC Active workspace

BOM Bill of Materials

BOP Bill of Processes

CEVT China Euro Vehicle Technology

DBSE Document Based System Engineering

EBOM Engineering Bill of Materials

GUI Graphical User Interface

INCOSE International Council on Systems Engineering

IVV Independent verification and validation

MBSE Model-based systems engineering

PA Physical Architecture

PAB physical architecture diagram

PLM Product Lifecyle management

POC Proof of concept

REC Replicable element collection

RPLs Replicas

SAE Society of Automotive Engineers

SE Systems engineering

SMW System Model Workbench

List of Figures

2.1 V model 7
2.2 Life Cycle Stages 8
2.3 Model Based Systems Engineering 10
2.4 Main Views & Perspectives structuring the Arcadia Approach 11

3.1 Research Methodology Chart 16

4.1 Operational Capability diagram for 'Auto Dim' 21
4.2 Operational Entity Scenario diagram for 'Auto Dim' 21
4.3 Operational Architecture diagram 22
4.4 System Architecture diagram 23
4.5 Logical Architecture diagram 25
4.6 Functional Exchange and Physical Link 27
4.7 Component classification Types available in Capella 27

5.1 Requirements structure in Teamcenter Active workspace 32
5.2 Functional Requirement mapping 32
5.3 Product specific Requirement mapping 33
5.4 Non-functional Requirements mapping 34
5.5 Show View Option 34
5.6 AWC view in SMW 35
5.7 Vehicle System- Sub system Structure 35
5.8 Sub systems in an Assembly 37
5.9 Sub system used in multiple systems 38
5.10 Defogging Functions and Requirements 40
5.11 Semantic Browser 40
5.12 Defogging functionality - Physical components and Requirements . . 41
5.13 Trace link Matrix 41
5.14 Parameter objects in Teamcenter and Capella 43
5.15 Working Principle 45
5.16 System Architecture - Functional chain Diagram of Auto Dim 46
5.17 Physical Architecture - Functional chain Diagram of Auto Dim . . . 46
5.18 Requirement Definition 47
5.19 Validate option in SMW 49

8.1 Digital Thread 56

A.1 Operational Architecture for Door Mirror I
A.2 Operational Capability Diagram II
A.3 Operational Activity Interaction [OAIB] - Defogging III
A.4 Operational Activity Interaction [OAIB] - Auto Dim III
A.5 Operational Entity Scenario - Defogging IV
A.6 Operational Entity Scenario - Auto Adjust V
A.7 Operational Entity Scenario - Auto Dim VI
A.8 Operational Entity Scenario - Blind spot VII
A.9 Operational Entity Scenario - Memory function VIII
A.10 Operational Entity Scenario - Power fold IX
A.11 Operational Entity Scenario - Turning Indicator signal X
A.12 Operational Entity Scenario - Show objects XI
A.13 System Architecture diagram XII
A.14 Logical Architecture diagram XIII
A.15 Physical Architecture diagram XIV

List of Tables

4.1 Rear View Mirror Sub Assemblies List 20
4.2 List of Rear View Mirror Functions 24

Contents

List Of Abbreviations viii

List of Figures ix

List of Tables xi

1 Introduction 1
- 1.1 Background . . . 1
  - 1.1.1 Background of the Project . . . 1
  - 1.1.2 CEVT . . . 1
- 1.2 Thesis Objectives . . . 2
- 1.3 Barriers in Existing product development . . . 2
  - 1.3.1 Relation between Requirement and Component . . . 2
  - 1.3.2 Handling Requirement Change . . . 3
  - 1.3.3 Relation between Hardware and Software components . . . 3
  - 1.3.4 Parts Reuse . . . 3
  - 1.3.5 Validation & Verification . . . 4
- 1.4 Research Question . . . 4
- 1.5 Delimitation . . . 5

2 Literature Study 6
- 2.1 Systems Engineering . . . 6
  - 2.1.1 Systems Engineering lifecycle . . . 7
- 2.2 Document Based System Engineering . . . 8
  - 2.2.1 Overview . . . 8
  - 2.2.2 Limitations . . . 9
- 2.3 Model Based Systems Engineering (MBSE) . . . 9
- 2.4 Arcadia Architecture . . . 10
  - 2.4.1 Operational Analysis . . . 12
  - 2.4.2 System Analysis . . . 12
  - 2.4.3 Logical Architecture . . . 13
  - 2.4.4 Physical Architecture . . . 13
- 2.5 Polarsys Capella . . . 13
- 2.6 System Model Workbench . . . 13

3 Methodology 15

3.1 Identifying Problem statement . . . 15
3.2 Literature study on MBSE & ARCADIA Architecture . . . 16
3.3 Identifying a Product for Case Study . . . 17
3.4 System Modelling . . . 17
3.5 Managing & Mapping Requirements in PLM System . . . 17
3.6 Use Case Implementation . . . 18

4 Case Study **19**
4.1 Case Study: Description . . . 19
4.1.1 Product Chosen . . . 19
4.2 Case Study: Implementation . . . 20
4.2.1 Operational Analysis . . . 20
4.2.2 System Analysis . . . 21
4.2.3 Logical Architecture . . . 24
4.2.4 Physical Architecture . . . 25

5 Use case Implementation **29**
5.1 Use Case 1: Create Product Model . . . 29
5.1.1 Description . . . 29
5.1.2 Purpose . . . 29
5.1.3 Expected outcome of the Use case . . . 30
5.1.4 Use case Implementation using MBSE . . . 30
5.2 Use Case 2: Identify Requirements and relate them to correct model elements . . . 31
5.2.1 Description . . . 31
5.2.2 Purpose . . . 31
5.2.3 Prerequisites . . . 31
5.2.4 Use case Implementation . . . 31
5.3 Use Case 3: System to Sub System Transition . . . 35
5.3.1 Description . . . 35
5.3.2 Purpose . . . 35
5.3.3 Prerequisites . . . 36
5.3.4 Limitations . . . 36
5.3.5 Working Principle . . . 36
5.4 Use Case 4: Handling Requirement Change . . . 38
5.4.1 Description . . . 38
5.4.2 Purpose . . . 39
5.4.3 Prerequisites . . . 39
5.4.4 Use case Implementation . . . 39
5.5 Use Case 5: Parameter management in TC and Models . . . 42
5.5.1 Description . . . 42
5.5.2 Purpose . . . 42
5.5.3 Limitations . . . 42
5.5.4 Prerequisites . . . 42
5.5.5 Use case Implementation . . . 42
5.6 Use Case 6: Identify Conflicting requirements . . . 43
5.6.1 Description . . . 43

5.6.2 Prerequisites . . . 43
5.6.3 Working Principle . . . 44
5.7 Use Case 7: Requirement Fulfilment Dashboard . . . 44
5.7.1 Description . . . 44
5.7.2 Purpose . . . 44
5.7.3 Limitations . . . 44
5.7.4 Working Principle . . . 44
5.8 Use Case 8: Automated test and Validation of Requirements . . . 45
5.8.1 Description . . . 45
5.8.2 Purpose . . . 45
5.8.3 Prerequisites . . . 45
5.8.4 Use case Implementation . . . 46
5.9 Use Case 9: Solution Variant as Subsystem SMW Project . . . 47
5.9.1 Description . . . 47
5.9.2 Prerequisites . . . 47
5.9.3 Limitations . . . 47
5.9.4 Purpose . . . 48
5.9.5 Use case Implementation . . . 48
5.10 Use Case 10: Validate SMW models and manage divergency . . . 48
5.10.1 Description . . . 48
5.10.2 Prerequisites . . . 49
5.10.3 Implementation . . . 49
5.11 Use Case 11: Joint construction of architecture and product variability 49
5.11.1 Description . . . 49
5.11.2 Prerequisites . . . 50
5.11.3 Purpose . . . 50
5.11.4 Implementation . . . 50

6 Recommendation 51
6.1 Teamcenter & Integration Recommendations . . . 51
6.2 Modelling Recommendations - MBSE . . . 51
6.3 Implementing MBSE in a new Project . . . 52

7 Discussion 54

8 Conclusion 55
8.1 Benefits of implementing Model based systems engineering . . . 56
8.1.1 Master product change . . . 56
8.1.2 Digital Thread . . . 56
8.1.3 Transparency . . . 56
8.1.4 Time to market . . . 57

Bibliography 58

A Appendix I
A.1 [OAB] Operational Architecture Diagram . . . I
A.2 [OCB] Operational Capabilities & Operational Actors . . . II

A.3 [OAIB] Operational Activity Interaction - Defogging III
A.4 [OAIB] Operational Activity Interaction - Auto Dim III
A.5 [OES] Operational Entity Scenario - Defogging IV
A.6 [OES] Operational Entity Scenario - Auto Adjust V
A.7 [OES] Operational Entity Scenario - Auto Dim VI
A.8 [OES] Operational Entity Scenario - Blind spot VII
A.9 [OES] Operational Entity Scenario - Memory function VIII
A.10 [OES] Operational Entity Scenario - Power fold IX
A.11 [OES] Operational Entity Scenario - Turning Indicator signal X
A.12 [OES] Operational Entity Scenario - Show objects XI
A.13 [SAB] System Architecture diagram XII
A.14 [LAB] Logical Architecture diagram XIII
A.15 [PAB] Physical Architecture diagram XIV

1

Introduction

The Introduction includes the background of the project, company and the outcome of the thesis work. The objective of the thesis work has been formulated along with a major research questions. This chapter will also describe the delimitation considered in this thesis work.

1.1 Background

1.1.1 Background of the Project

Systems Engineering has become increasingly common in the automotive sector in the past decade. Systems engineering deals with designing and managing of the systems, starting from gathering the customer needs, proceeding with the design synthesis and validation. Even though, Automotive companies strive hard to satisfy the changing customer needs but on the other hand, companies are struggling to close the loopholes due to the disconnected product development activities. The disconnected domains creates a major impact when there is a requirement change. This creates a need for a system or a method which provides traceability throughout the development of a product. MBSE provides advantages over the systems engineering approach by providing end to end traceability and enables to manage the complex systems easily.

More detailed information on systems engineering and model based systems engineering and its methods will be discussed in the upcoming chapters.

1.1.2 CEVT

The thesis work is performed in collaboration with China Euro Vehicle Technology AB, hereon referred as CEVT, at the department of Product Lifecyle management (PLM). CEVT is a development center for future cars of the Geely Group. CEVT covers all aspects of passenger car development – from the total architecture, powertrain and drive line components as well as the vehicle's exterior design. It develops global solutions in many different areas especially in automotive sector through modular development, virtual engineering and continuous innovation. One of the main domains in the automotive sector is Product Lifecycle management, hereon referred to as PLM, which is responsible for managing the data and integration between the

PDM, CAD, ERP thorough out the lifecycle of the product. At CEVT, a part centric PLM solution is implemented within Teamcenter to manage documents, CAD models visualization, product configuration, Engineering Bill of Materials (EBOM), Bill of Process (BOP). All the development teams collaborate with the PLM department at the different phases of the product lifecycle.

1.2 Thesis Objectives

The Master thesis work investigates and provide recommendation about the utilization of functional driven product development at CEVT. The thesis aims to evaluate Model based systems Engineering and provides recommendation about utilizing MBSE for product modelling. In addition to the MBSE investigation, the research also examines the MBSE tool and captures the merits and the demerits in the form of use case validation. The outcome of the thesis work is to deliver a Proof of concept (POC) in an MBSE platform. The purpose of the thesis has been narrowed down to the following research question.

1.3 Barriers in Existing product development

This section will predominantly aims to identify the problems that is being encountered in the existing product development practices. The section focuses on identifying the problems that are existing from collecting the stakeholder needs, documenting the needs, followed by concept generation, evaluation and production phases. Even though there is an existing PLM system that connects several dots in the product development process, there are several disconnected domains. This section summarises a list of problems that are existing in the current product development activities. A qualitative data collection method is carried out with CEVT professionals in the form of interviews to understand the problems in the existing product development. Also, several international journals have been read to identify the prevailing problems in the automotive industry. Below is the list of problems that are identified in the existing product development activities.

1.3.1 Relation between Requirement and Component

The lack of integration between the components and the technical requirements is one of the major problems faced by the firms nowadays. One reason is manufacturing firms utilize separate software's to manage technical requirements and Bill of Materials (BOM) and BOP of the parts. Another reason is due to the lack of cross platform integrations. The requirements are managed in a separate tool [System weaver is used in CEVT]. All the components, bill of materials and bill of processes are managed in different PLM tool [Teamcenter is used in CEVT]. The system viewer does not provide integration of requirements with the parts in Teamcenter. Identifying the associated parts to the requirements becomes a tedious process due to the disconnected systems.

1.3.2 Handling Requirement Change

Considering, there is a requirement change, it becomes increasingly difficult to identify the impacted parts. Sometimes, there is also chance of missing any associated requirements to the primary requirement. For example – If there is a requirement to change the diameter of the tyre, not only the tyre that must be redesigned, the corresponding rim must also be redesigned. So, In the existing product development process, all these parts identification for a particular requirement is purely depends on the experienced professionals. The current situation is purely depending on the experienced and skilful users to understand the impacted parts and the requirements. This becomes tedious task and creates dependencies on the user.

1.3.3 Relation between Hardware and Software components

Automotive industry has seen a remarkable increase in the use of the electronic components in the last decade. Even though the increasing number of electronic components in the automotive systems provides enormous advantages through improvement in the performance, comfort and safety, the complexity of the components has increased in a rapid pase [2].

According to this journal[2], today's modern cars have up to 70 ECU's connected by various system buses. The cost plays a significant role in the growing system complexity. In order to overcome the vehicle development cost, there are several standards that has been introduced in the automotive industry. Even though the standards increase the reusability and modularity, automotive industries require powerful tools for managing the increasing complexity in the HW and SW development.

The hardware and software development are carried out by a separate department. Even though the parts related to the HW and SW development are managed in a PDM tool. The trace link between the HW and SQ components are not established due to the high complexity. The missing tracelinks makes it tedious task to identify the relevant software components for a chosen hardware. This creates a huge impact during Validation & Verification stage of the product development process and increases the margin of errors while testing. In certain situations, some software component goes unnoticed while testing its corresponding hardware component. These issues create a huge impact which ends up in the vehicle recalls. According to NHTSA [3], 70,000 vehicles has been recalled by Volvo cars due to the missing software update[**Expresson**]. This creates a need for a transition from a conventional development process to a model-based approach.

1.3.4 Parts Reuse

Reusing the product architectures and the components is one of the easiest ways to leverage the investments in the research and development area. Companies achieve values by reusing the technologies across different products in their portfolios. Even though the benefits achieved from the reusing strategies are enormous, finding the

right technology and the components for reuse is always a challenge in the automotive industry. The capability to reuse highly depends on a knowledge platform or established traceability between the product and its functions[4].

According to Prieto and Freeman [5], reuse is the use of previously acquired concepts and objects in new context. There are two levels of reuse are considered in this journal. One is the reuse of the ideas and knowledge and another is the reuse of the artifacts and component. In this thesis work, component and artifacts reuse will only be considered.

Knowledge from the previous projects is a great source of information for the system engineers which they can employ to develop solutions to new technical requirements that arise from the new projects [6]. But Identifying the components and the artifacts for reuse is the major problem. The lack of knowledge platform and missing tracelinks in an organization restricts the reuse of parts. This results in the creation of the duplicate part for the existing requirement and part.

1.3.5 Validation & Verification

Validation and Verification is a process of checking whether the technical requirements defined in the pre-development stage is fulfilled and the results are inline with the stakeholder needs defined. According to Houdt & Vrancken[7] , V&V can be performed at any level of the project and it is not just related to the end result of the project. During the V&V process, client and R&D team must collaborate in sharing information to make sure the desired outcome of the project is obtained. But inorder to accomplish all the above activities, a standard V&V procedures are required. In the existing product development practise, test methods are used for validating the requirement. The test methods are in the form of documents which has information about the requirement that needs to be tested and its corresponding test method and part id. Even though the test methods has the part id [In PLM system] associated with it, there is no direct tracelinks between the parts and the test methods. This may look simple, but due to the increased level of automation, the level of testing is increased which forces the system testers to validate million scenarios for the Society of Automotive Engineers (SAE) level 5 – Autonomous vehicles. This makes the Validation and Verification more challenging and the dependencies require cross domain verification and validation.

1.4 Research Question

The thesis focuses on investigating the following research questions.

RQ1: How MBSE could be used to address the barriers identified in the existing product development process in practise?

The research question investigates the Model based systems engineering technique and also the MBSE methodology followed. The research also examines the MBSE tool that is utilizing the methodology. A list of product development activities

which is considered as primary importance are formulated as use cases. And MBSE is testing against it to verify the use case fulfillment.

RQ2: What are the benefits that could be achieved by implementing MBSE over DBSE?

The research question investigates, analyse and compares existing product development activities with the MBSE and documents the various beneficial aspects upon implementation.

1.5 Delimitation

There are several delimitation that are considered in this thesis work. The Thesis work is performed by only one person for the course of 20 weeks. The thesis will not focus on modelling and managing the requirements in PLM platform. But the existing requirements structure is utilized to verify the integration functionalities. The work will not discuss about any release procedures for the models defined in the MBSE platform. The cost savings upon implementation of the MBSE in business will not be discussed. The research work will discuss only about ARCADIA methodology. It will not focus on other MBSE methodology such as SysMl or OPM. The thesis work will not focus on the release procedures of the project models in Capella.

2

Literature Study

In this Chapter, a theory about systems engineering, model-based systems engineering (MBSE), Document Based System Engineering (DBSE) will be discussed. The limitations of DBSE will also be discussed in this section.

2.1 Systems Engineering

The International Council on Systems Engineering (INCOSE) SE Handbook defines System Engineering as a 'transdisciplinary and integrative approach for a successful realization, use and retirement of engineering systems using system principles and concepts, scientific, technological and management methods'. The inter disciplinary approach focuses on defining the stakeholder needs and the required functionality early in the development cycle, documenting and performing design synthesis and system validation while considering the problems [8]. Houdt [7] states that even though there are different definitions for SE, they show a high degree of similarity and the ultimate goal is to create a successful system which solves a complex problem. He also adds by saying that SE should contribute to the traceability of the requirements and effectiveness of the working process. According to INCOSE, SE integrates all the speciality groups from different disciplines forking a structured development processes that proceeds from the concept to production to operation. SE considers all the business and the technical needs of the customer. The success of the approach is mainly determined by the tools, techniques and the methods. [9] [10]

According to International Council on Systems Engineering (INCOSE), some of the key elements of the systems engineering are system, system engineer and the life cycle processes. INCOSE [11] defines system as a combination of interacting elements organized to achieve one or more stated purposes. The system elements the forms a system may include hardware, software, firmware, people, techniques, facilities and services. A system engineer is a role who supports the transdisciplinary approach by eliciting the stakeholder needs into technical specifications that are realized by the system development team. The last element is a set of Lifecyle processes which are followed beginning in the conceptual design and continuing throughout the lifecycle of the system through its manufacturing, use and disposal.[8]

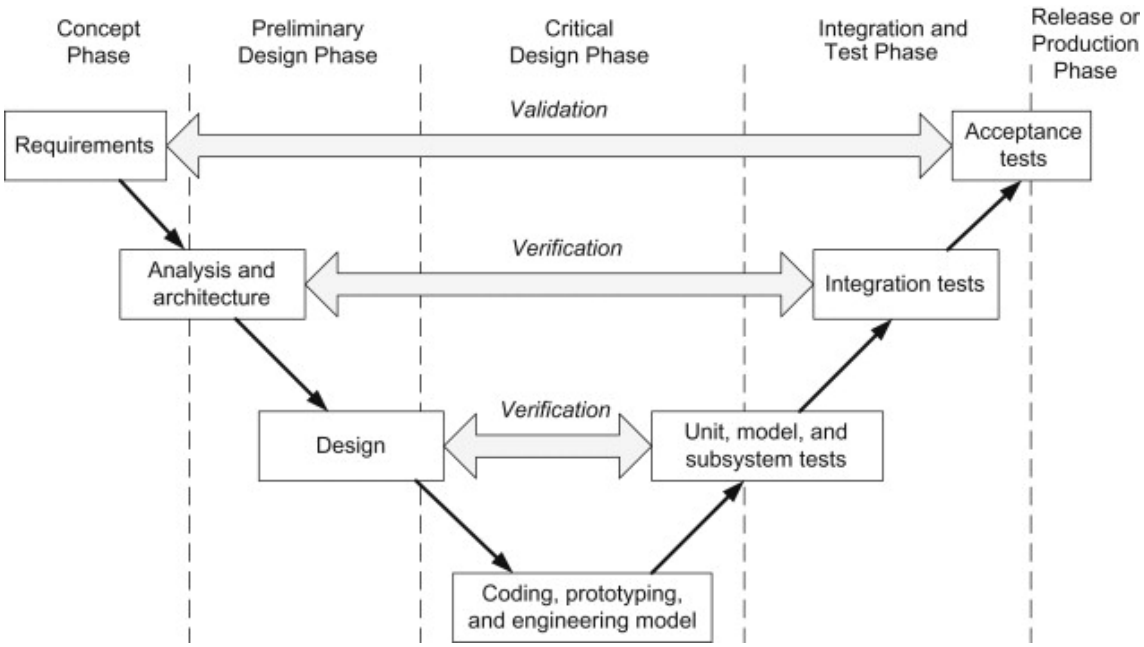

Figure 2.1: V model [12].

According to INCOSE, systems thinking is a way of thinking used to address the complex problems. It focuses on the behaviour of the complete system rather than a individual components. It is the ability to think about the interactions between the elements of the system.[13]

2.1.1 Systems Engineering lifecycle

According to INCOSE, there are large number of lifecycle process models, but the life cycle process mentioned in this chapter will be based on the Vee model. The lifecycle models differ from industry to industry. Several Lifecyle models has been defined by the key players like The National Aeronautics and Space Administration (NASA), the US Department of Defense (DoD) and others. Even though there are different lifecycle process models, all address the same set of system engineering activities. According to INCOSE[8], the key difference is the way the activities are grouped and represented.

The different variations of the process models deal with the generic stages of life cycle. Once such generalization done by INCOSE is shown in the figure 2.2. There are seven stages of life cycle. Exploratory research stage involves with identification of the stakeholder requirements and constraints. Concept stages focuses on determining the best approach to meet the stakeholder needs. Also, several concepts are also explored in parallel. Development stage deals with defining the system functions and identifying the best solution that meets the stakeholder requirements. The validation and verification of the concept is also performed at this stage by enabling user involvement. Production phase focuses on the manufacturing of the system.

All the production constraints must be considered, and it may influence the system requirements and may require re-verification or re-validation. Support stage ensures that the services are provided to ensure proper functioning of the system and its components.Finally, the retirement stage deals with removing the product and its services from the operation. The system activities at this stage focuses on ensuring the disposal requirements are satisfied[8].

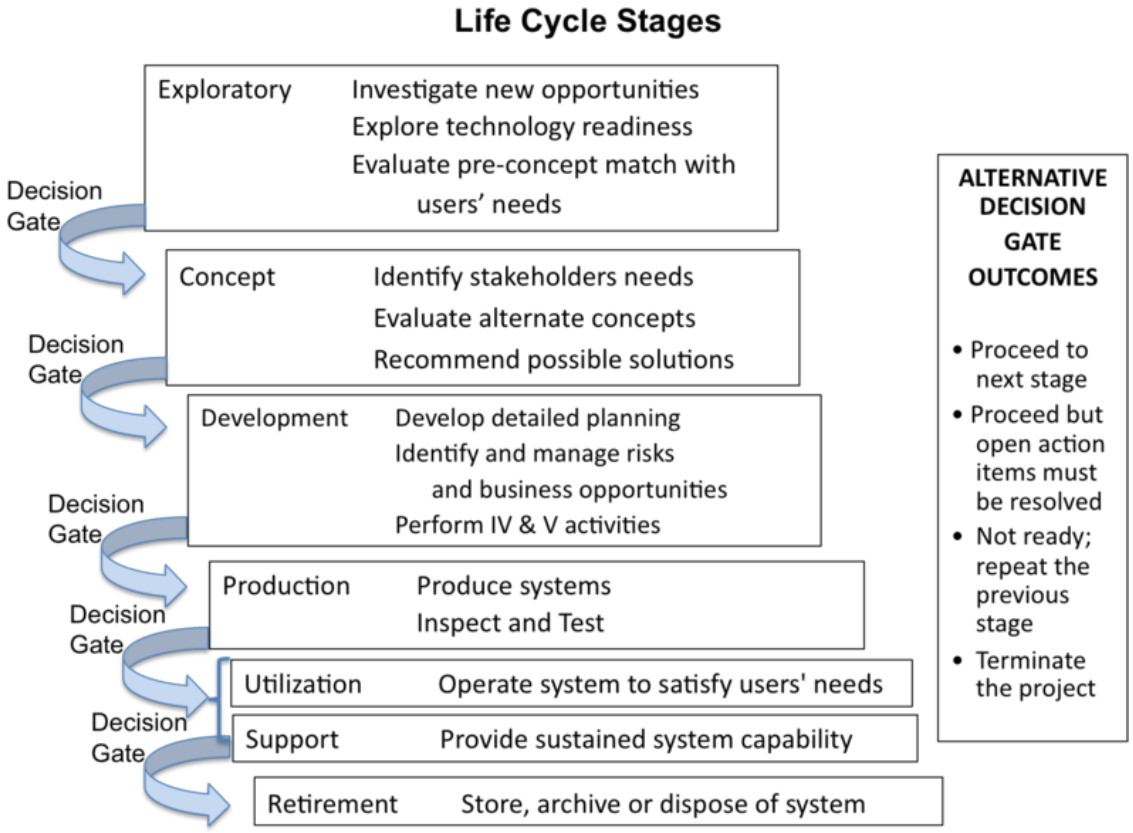

Figure 2.2: Life Cycle Stages [12].

2.2 Document Based System Engineering

2.2.1 Overview

In the traditional system engineering method, which was depicted in figure, the outcome of each stage of the SE process will be a set of documents. These documents are an output of design activity and analysis which will be the input for next stage in the product development process. Each stage is validated by conducting a review of the documents at each stage. The requirement statements are the only connection between the different phase of the system engineering process[14].

2.2.2 Limitations

According to the various studies and researches, document-based system engineering approach encounter several issues in the Product development processes. Since DBSE focusses on dealing with the textual statements, it becomes a tedious process to identify the impacted items if a change request arises. The limited traceability between the parts and design specification makes it difficult to estimate the time and cost to fulfil the changes requests. Communication between the teams is one of the major factors for the success of any organisation. The document publication and release ensure as the medium of communication between the engineering teams. Even though the documents are managed with certain rules and regulations, certain part of the documents are difficult to interpret by other teams which forces the architects to build additional drawings to support communication. Also, the widespread of various documents, it makes it difficult for the engineers to identify the correct document at the right time [15].

2.3 Model Based Systems Engineering (MBSE)

Model based systems engineering is a Systems Engineering methodology that uses domain models as a primary means to exchange the information between the systems engineers. MBSE formalizes the practice of systems engineering through the use of models. MBSE captures the system engineering information in a single model database. In MBSE, each of the system elements and their relationships are established in the form of information model. MBSE also supports by generating documents in the form of reports from the data in the model repository. These documents could be used for further evaluation or validation purpose [16].

According to INCOSE SE Vision 2020, "Model-based systems engineering (MBSE) is the formalized application of modelling to support system requirements, design, analysis, verification and validation activities beginning in the conceptual design phase and continuing throughout development and later life cycle phases".

Model Based Systems Engineering (MBSE) integrates and coordinates the tools, processes and people from diverse disciplines such as mechanical, electrical and software. MBSE allows the system architects to capture the needs of the customer as stakeholder requirements. It then supports the design, manufacturing and validation and verification process. MBSE also facilitates change and configuration management [17].

MBSE provides several advantages over the document-based systems engineering such as improved communications amongst different domains, end to end requirement traceability, improved ability to manage system complexity. By implementing MBSE in an organisation, a system architect would be able to understand and identify the traceability between requirements, functions and its related component.

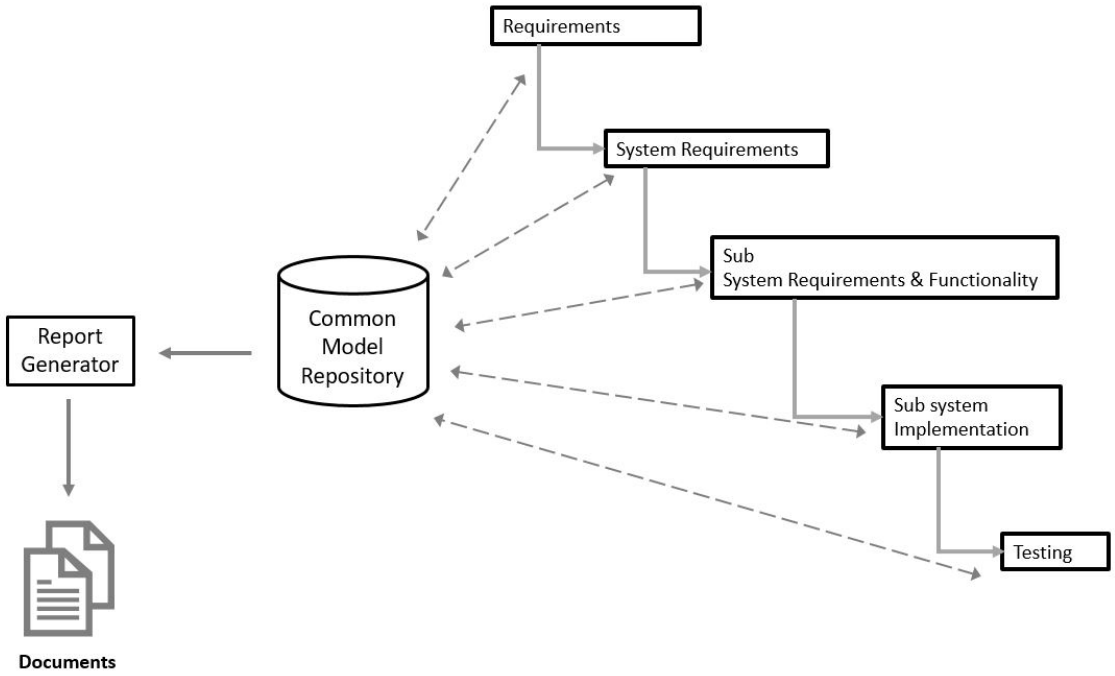

Figure 2.3: Model Based Systems Engineering

Requirement traceability is a necessity tool which helps in identifying the parts in the case of requirement change. According to INCOSE [18], the models supports three aspects of system engineering namely requirement traceability, system architecture and organisational process flows.

According to INCOSE [19], modelling is all about reducing the complexity of the system, understanding of the system, improving communication and providing reusable elements. Capturing a lot of different aspects like requirements, functions, interfaces, behaviour of the complex systems leads to a large model. Therefore, a well structured model is crucial for controlling the complexity of the product.

There are several MBSE methodologies and tools available in practise. The MBSE methodology available in practise are based on functional decomposition and object-oriented decomposition. Irrespective of the tool, the methodology should help the system engineers to understand the problem and to communicate with others about the problem.

According to [19], there is no common thread that runs through the different methodologies except that it should clearly defined to support model development.

2.4 Arcadia Architecture

Architecture Analysis and Design Integrated Approach (ARCADIA) is a model-based system engineering approach based on models which is used for defining and verifying the architectures of a complex system. ARCADIA promotes collaborative way of working among all the key players, right from the engineering phase till their

Independent verification and validation (IVV). The general principles of the ARCADIA are briefly described in the following sections. All the information produced by the engineering team describing technical requirements and solutions are grouped within a single engineering model which will be shared by various actors involved. This promotes all the stakeholders in the company to share the same information in the form of shared model elements[20] .

The approach also supports collaboration and co-engineering between different engineering levels which allows for joint model development [10]. Different engineering teams could be mechanical, electrical, electronics and software. Each set of constraints associated to one of the above specialized domains is formalized in a dedicated perspective which captures the expectations of the system. And each solution variants from the domains are subjected to verification in the early design through the analysis of the models.

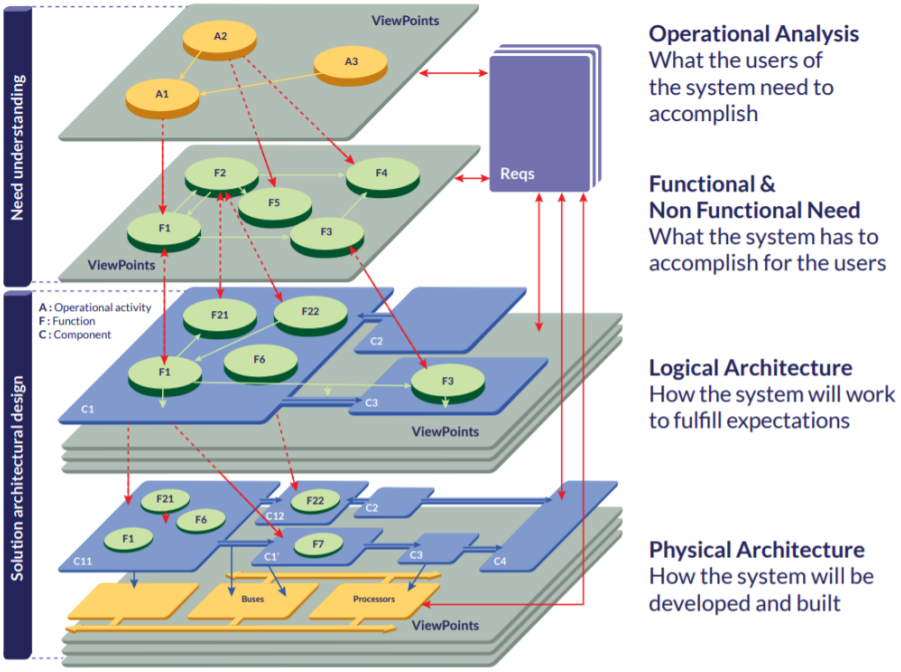

Figure 2.4: Main Views & Perspectives structuring the Arcadia Approach [20]

In Arcadia Architecture, there are four different phases. Each level represents different operations. The four stages of the architecture are Operational analysis, system analysis, Logical Architecture, Physical architecture. The first two levels of the architecture focuses on the gathering the stakeholder needs, interpret the needs and transforming the needs into functions. The logical and physical architecture helps in identifying the design solution for the functions and stakeholder needs defined in the previous stage. All the models in different engineering teams are connected to each other.

- Operational Analysis

- System Analysis
- Logical Architecture
- Physical Architecture

In order to utilize the potential of the MBSE, a methodology consisting of appropriate methods, tools and process is a key necessity. Capella is an MBSE tool which utilizes ARCADIA architecture. The more information about the capabilities of the Capella and its offerings will be discussed in detail in upcoming chapters.

Even though there are existing MBSE tools in the market which are capable of representing any system through viewpoints, it is always a challenge to realize the full potential of the MBSE. In the coming chapters, we will discuss in detail about the different phases of the ARCADIA architecture and the utilization of ARCADIA in Capella. The architecture development capabilities of the ARCADIA – Capella will be discussed and evaluated in later stages with the help of use cases. Each phase of the architecture will be explained in detail in the following sections.

2.4.1 Operational Analysis

This phase focuses on achieving and analyzing the stakeholder needs and translating them into stakeholder technical specifications. Before identifying the stakeholder needs and technical requirements, mission statement defined by the management must be considered based on the defined goals and objectives. The perspective also identifies the operational actors that are interacting with the system, their goals, activities and constraints.The actors and entities identified must be linked to their respective capability. After careful consideration of the technical and management goals, stakeholder needs and technical specifications are identified. According to the paper [10], Operational needs analysis is highly promoted by the architecture frameworks such as DoDAF, TOGAF, NAF.

2.4.2 System Analysis

This phase of the architecture builds an external functional analysis based on the stakeholder requirements and operational activities. It identifies the system functions and non-functional requirements.

System architecture is a description of the system in functional terms. The functional architecture shows the functional interactions of the system components without identifying the actual components. For each capability, functional chain is formed which is the sequence of functions for a particular capability. Functional analysis is one of the most important techniques in SE methodologies. Ignoring any functions at this stage might have serious impact in the later stages of solution development.

2.4.3 Logical Architecture

The Logical Architecture is an abstract representation of the system components. This is the first major solution development stage. The solution identified at this stage are represented as logical components and its respective functions are defined as logical functions. Logical level function decomposition is a necessary step which helps in identifying the sub functions of the sub systems which satisfies the higher level functions. The identified logical elements defines the logical architecture that implements the functions defined in the system analysis phase. The solution development must not be constrained to a specific solution rather the architecture should have solution for different system variants.

2.4.4 Physical Architecture

This is the detailed solution development phase. The objectives of this phase is similar to the previous phase except that the solution defined at this stage are the finalised architecture. The physical components must be identified at this stage for a corresponding logical component. Therefore, a system function satisfied by one logical component may also require one or more physical components.The components interfaces must be identified and the connections to the relevant components must be established.

2.5 Polarsys Capella

Capella is an open source application for model based systems engineering. It is a comprehensive, extensible and field-proven MBSE tool and method to successfully design systems architecture. It provides graphical modeling of systems, hardware or software architectures in accordance with the principles of ARCADIA method. Capella supports ARCADIA method by providing seven characterized types of diagrams that are: data flow diagrams, entity scenario diagrams, architecture diagrams, mode and state diagrams, product breakdown diagrams, class diagrams and capability diagrams.[10]

2.6 System Model Workbench

System Model Workbench (SMW), is an integrated system engineering solution provided by Siemens to incorporate the model-based system engineering concepts to the entire product development process. System Modeling workbench (GUI) is an implementation of Capella with some additional integration features to Teamcenter. With the Teamcenter integration, system engineers can establish tracelinks between the technical requirements and the product models. As mentioned in the earlier chapters, the requirements could be integrated at any stage of the ARCADIA architecture. SMW also provides option to export the product models in SysML format. And it is also possible to import the SysML models in Teamcenter SMW. All the product models are exported and saved in Teamcenter database. And the product

models must be checked out to access and edit the models. [21].

3

Methodology

This chapter presents the systematic approach followed throughout the project. This section also discusses about the theory about the methods that are used in the research work. The initial part of the project focuses on exploring about Systems engineering (SE), Model based system engineering (MBSE) and tools available in the market. Various literature about MBSE are gathered in identifying the advantages of utilizing MBSE in automotive industry. The second phase of the research work focused on exploring about the MBSE tool. Capella and System Modeling workbench are the MBSE tool that will be used in this thesis work. Capella is utilising the MBSE technique called ARCADIA. So, a deep study is carried out in investigating and exploring about the ARCADIA methodology. On the other hand, the implementation of ARCADIA in Capella is also studied by exploring the tool. The third phase is to understand the existing product development process in practise. A number of interviews and meetings has been organised to understand the product development activities followed.

In order to apply the methods in practise, an appropriate sub system which has components from diverse domains namely electrical, electronics, mechanical etc must be chosen. After verifying many sub systems, Rear view Door mirror has been chosen for the study. The modelling of the Door mirror sub system is performed in MBSE tool. At this same time, technical requirements for the Rear-view door mirror are created in Teamcenter. Last stage is the validation part. A list of product development activities which are considered as prime importance are formulated in the form of use case. And the Capella / SMW models for the Door mirror is validated against the use case. In the upcoming paragraphs, different methods used in the different stages of the thesis work will be discussed.

3.1 Identifying Problem statement

Several Structured interview sessions has been conducted with the system engineers and Solution Architects at CEVT to comprehend the underlying problems in the existing product development in practise. A set of qualitative questions has been formulated to understand the change management processes, requirements management and parts reuse.

The methodology followed for the thesis work is shown in the form of flow chart in the figure 3.1.

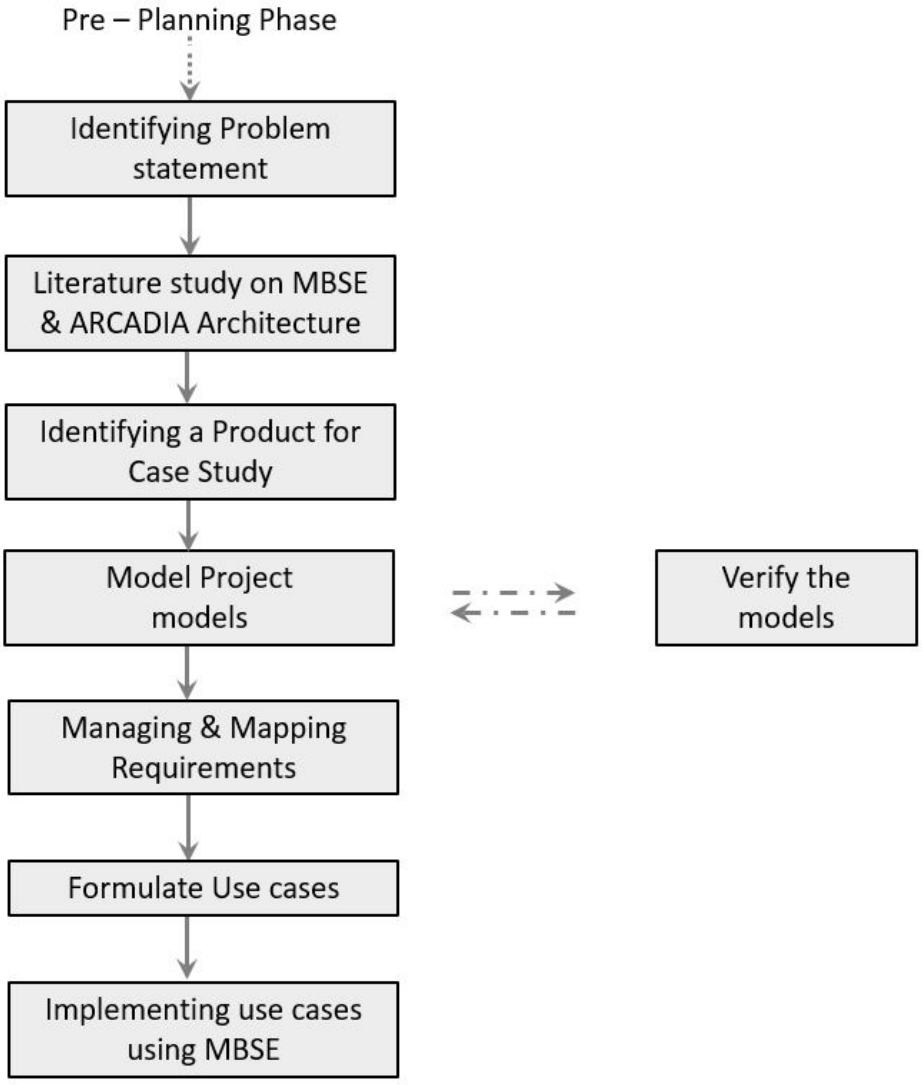

Figure 3.1: Research Methodology Chart

3.2 Literature study on MBSE & ARCADIA Architecture

In this phase, a research was carried out to get an understanding about MBSE and ARCADIA. Various literature and journal papers has been referred on MBSE. The system engineering handbook by INCOSE was exclusively used to gain knowledge on the system engineering and MBSE processes.

The second phase of the literature study is performed on ARCADIA methodology. The Model Based systems Architecture Engineering book by Jean Luc Voirin was used through out the thesis work to understand, implement and verify the models

using ARCADIA methodology. The journal papers on other MBSE methodologies are also studied to gain understanding, relations and differences. But the study on other tools and methodologies was not carried out so intense since the work focused only on ARCADIA.

3.3 Identifying a Product for Case Study

This phase of the thesis involves selecting an appropriate product for modelling the system architecture. Choosing a right product is critical considering the time span and the product complexity. It is also made sure that the selected product has subsystems from different domains which is one of the important criteria in product selection. For instance - Components from mechanical, electrical, electronics, etc. A set of products namely Gear box, Rear view Door mirror has been shortlisted for product modelling. After having discussions with the Solution architects, system engineers and carefully considering the criteria, Rear view mirror is chosen as a product for modelling architecture.

3.4 System Modelling

After gathering all the product information, the system and its sub systems are modelled using an MBSE tool. The modelling process carried out is greatly influenced by the modelling strategies provided by Jean Luc Voirin [20]. According to the modelling practise mentioned by Jean Luc voirin, the product is modelled using a top down approach. But it is also stated in the book that the modelling can be performed either by using a top- down approach or bottom-up approach. The methodology is also flexible enough to start modelling at any phases.

The modelling is carried out based on a top down approach which means the modelling started with the operational analysis followed by system analysis, logical architecture and physical architecture. Even though the chosen product is the existing one, the modelling is carried out considering that the product doesn't exist on market. This helped in documenting and modelling all the minor details in the development process.

3.5 Managing & Mapping Requirements in PLM System

This phase involves managing the technical requirements in a PDM platform. Later the technical requirements are mapped to its relevant product models. According to ARCADIA methodology, requirements could be integrated to any stage of the architecture. This requires requirements to be managed in the same application or in an application which provides integration with the MBSE tool. In the existing product development practise, technical requirements are managed in SystemWeaver©, a standalone application doesn't provide integration to MBSE platform. Therefore,

the technical requirements are imported in Teamcenter in order to establish the tracelinks with the product models. More information on the requirement mapping will be discussed in the upcoming chapters.

3.6 Use Case Implementation

A set of product development activities are formulated in the form of use cases. Several interview sessions has been conducted with the product development team in CEVT to comprehend the product development activities and the importance of it. After careful consideration, several activities are formulated and the implementation is carried out using MBSE. Post interactive sessions with the system engineers has been conducted to review and optimize the implementation.

4

Case Study

4.1 Case Study: Description

This chapter introduces a system that is used for the study using ARCADIA method. The purpose of the case study is to determine the potential, modelling strategies and constraints of the ARCADIA. The product chosen for the study is Rear View Door Mirror of a car. The product is chosen after careful consideration and made sure that the system has components from varied domains say mechanical, electrical, electronics, etc. The model project of Door Mirror is developed by considering Door Mirror as the system of interest and other systems in the vehicle say Door, engine etc as an external systems. The System is modelled using Capella & System Modeling workbench to a considerable level of detail which means all the distinct parts are included but not in detail. For example - Hardware and software parts are included for study but not all the hardware components are considered in detail.

4.1.1 Product Chosen

The Rear View Door mirror in a car is used to view the objects present in the blind spot of the driver. Even though the system looks less complex compared to other major systems in car say Engine and Transmission. In the past couple of decades, rear view mirror has evolved by incorporating automatic defogging, parking assist system in the system. The variety of components in a system makes it perfectly suitable for modelling. The Rear View Mirror has four major sub assemblies which are shown in the table 4.1.,

The Door Mirror system is modelled right from gathering the operational capabilities. The Operational capabilities are transformed into system functions followed by identifying the solution variants for the functions during concept development. The Rear View mirror is modelled by involving all the phases of the ARCADIA methodology. In order to track the model elements precisely, only one solution variant is considered. On the other hand, technical requirements of the Door mirror are gathered and documented in a PLM tool -Teamcenter. The technical requirements are modelled in the form of requirement structure using in Requirements management platform. The technical requirements are mapped to its relevant product models in various phases of product development.

S.No	Sub System Name
1	Camera -360 degrees Panoramic View
2	Rear View Mirror Skull Cap LH
3	Rear View Mirror Glass Assy Heat BSD LH
4	Rear View Mirror WO Cap Glass Heat BSD LH

Table 4.1: Rear View Mirror Sub Assemblies List

4.2 Case Study: Implementation

4.2.1 Operational Analysis

This phase of the architecture analyses the issue of the operational users. It enables to capture the stakeholder needs and actors interacting with the system.

The first step is to determine the objective or goal of the activity which is normally defined as a mission statement. Accordance to the mission statement, identify and define a set of operational capabilities which are necessary to achieve a mission statement successful. This leads to a fundamental question, what is Operational capabilities? Operational Capabilities are nothing but the stakeholder needs. The list of stakeholder needs for the rear view door mirror are as follows.,

- Should be able view the rear vehicles
- Should have auto dim option
- Should have defogging option

The challenges of the Operational users are analysed by identifying the operational actors and operational entities after capability definition. Who are Operational Actors? According to Jean Luc Voirin, an Operational actor is an external entity that is interacting with the system. In simple terms, Operational actors and entities are the actors interacting with the system. Considering our system of reference - Door mirror, the operational actors are Road Vehicles, Environment, Pedestrians, Driver, etc. For all the Operational actors defined, their corresponding operational activities and their interactions must be established. The main objective of identifying the activities and interactions between the key players is to capture the conditions for implementing the capabilities. The actors, entities, activities and information exchanged between actors are identified more precisely. All the above information should be gathered for each of the operational capabilities defined in order to accomplish the mission statement.The Figure 4.1 explains the different operational actors and their interactions with the corresponding operational capability [20].

The following Operational architecture diagram explains the different operational

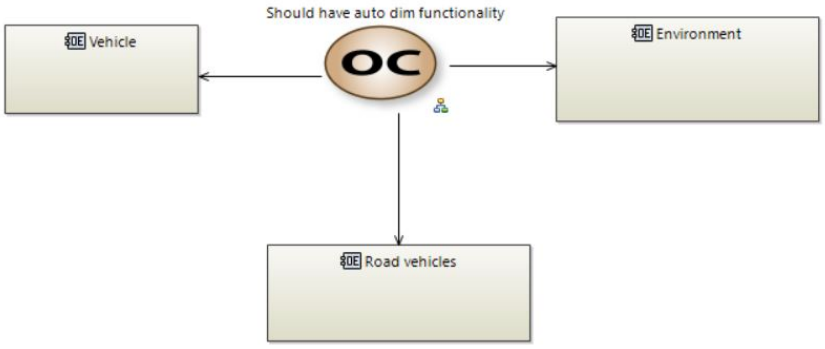

Figure 4.1: Operational Capability diagram for 'Auto Dim'

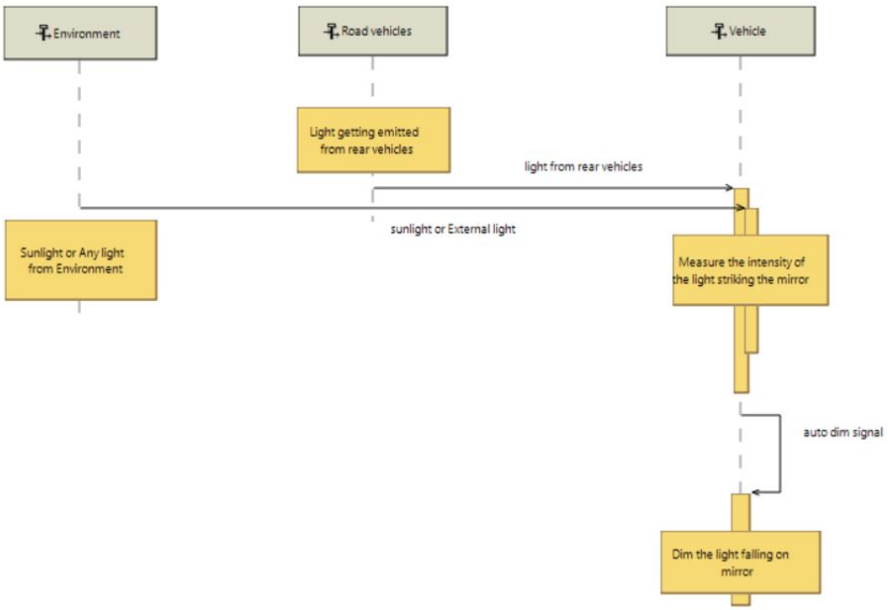

Figure 4.2: Operational Entity Scenario diagram for 'Auto Dim'

actors and the interaction between the different activities. The Operational exchanges are the information exchanges between the actors or operational entities. The exchanges are also defined as the result of an activity performed in the previous phase. Note: It is critical to identify all the capabilities, actors and their activities at this stage. Missing an capabilities will result in losing a physical part in later stages. And traceability must be ensured with all the model elements. It is essential to identify all the actors at this stages in order to capture all the stakeholder and their needs.

4.2.2 System Analysis

This phase of the architecture analyses the operational capabilities and stakeholder requirements defined in the operational analysis phase and performs a functional

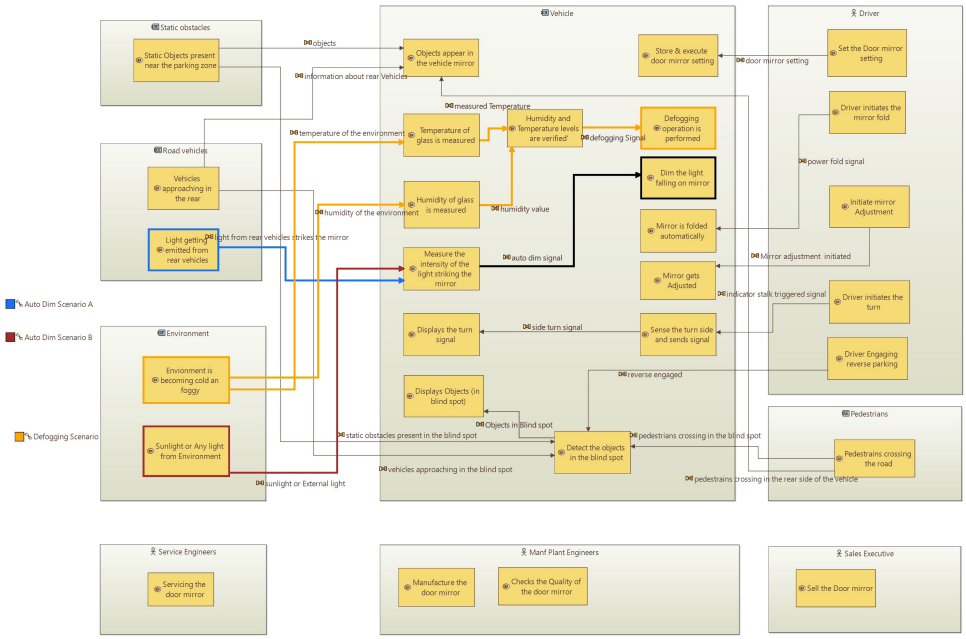

Figure 4.3: Operational Architecture diagram

analysis to identify the system functions and system actors. The sole purpose of the system needs analysis is to define the outcome expected from the system to satisfy the stakeholder needs mentioned in the operational analysis phase.

All the activities and the actors defined in the operational analysis phase will be realized as system function and system actors in system analysis after performing a functional transition. This means, all the operational components will be inherited as system level component after performing a functional transition.

The first activity at this stage will be identifying the system capabilities for the corresponding operational capabilities defined in the previous phase. In our case, operational capabilities are same as system capabilities. But Operational capabilities and system capabilities can also be different. Considering this example, Customer need might be to “reduce the reflection of light falling on the mirror”. In this scenario, a system capability could be “System should have auto dim functionality”. And system Engineers should identify a system function for the need. In our case its ‘Enable Auto Dim’.

The expectations of the system are formalized in the form of functions in the functional needs analysis which the system must satisfy for different capabilities. Performing a functional transition also creates a system function for all the operational activities defined in the previous stage. The system engineers can also add an additional system function in the system architecture diagram. The realization link must be established manually for the newly created system functions and other system models in this phase. All the system functions must be identified by considering the system as a black box.

The non-functional requirements are captured at this phase in the form of constraints. Identifying constraints at this stage is critical since it plays a vital role during solution development in logical architecture. The constraints are vital in ruling out the weak solutions.

Like the operational exchanges defined in the previous phase, the interactions between the functions are captured in the form of functional exchanges. Each functional exchange is an output from a function which will be fed as an input for another function. All the green lines in the system architecture diagram depicts the functional exchange. For instance – Vehicle light, static obstacle, auto dim signal, road vehicle. It is essential to limit the activities performed in the functional analysis stage, since it must not include any implementation details.

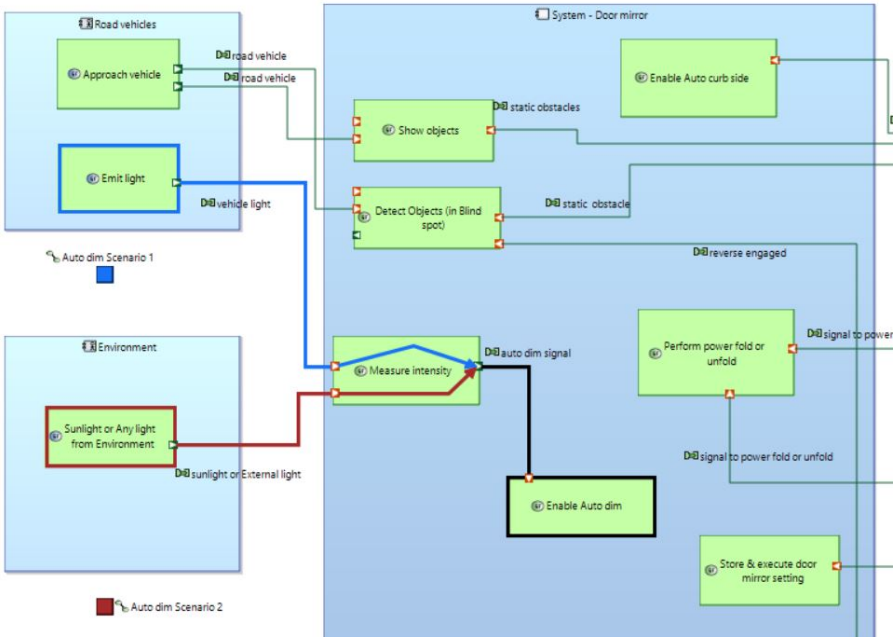

Figure 4.4: System Architecture diagram

After establishing the connections between the functions and model elements in the system architecture diagram, functional chain could be used to distinguish different functions and its associated actors. The System Architecture diagram shown in the figure 4.4 depicts two different functional chain for Auto Dim functionality. The first Functional chain (one in blue) referred as Auto Dim Scenario 1 provides vehicle light as an input to the function 'Measure intensity'. The second functional chain (one in brown) referred as Auto dim scenario 2 has sunlight or other environment light as an input to the main function. Likewise, multiple functional chain diagram could be used to demonstrate the functional behaviour of the system. The usage of the functional chain will be explained in later chapters[20].

Function Name	Function Description
Enable Auto dim	Should dim the light falling on the Door Mirror
Enable Auto adjust	Mirror should be able to adjust automatically
Enable Defogging	Defogging operation should be performed automatically
Enable Electrical folding	Mirror should be folded automatically
Enable Memory function	System should store the user defined setting
Display Turn signal	Turning indicator should be displayed
Measure temperature	Temperature should be measured
Monitor objects	Able to monitor 360 view around the car
Detection Objects	Should be able to identify Objects in Blind spot

Table 4.2: List of Rear View Mirror Functions

4.2.3 Logical Architecture

In response to the stakeholder needs and system functions defined in the previous two phases, it enables the system engineer to develop a first level of solution design. Logical functions are developed for the corresponding system functions defined in the previous phase. The Logical components which implements the functions are also modelled in the logical architecture.

The Logical architecture is maintained at the coarse-grained level i.e., in other words any detailed design will be described later in the physical architecture. This makes the solution to be easily understood by hiding the complexity of the solution. The level of abstraction at this stage helps the marketing team, needs analyst, engineers to make an important decision that govern the design of the system who does not require detailed information to decide.

There are several reasons to justify the interest of a first level of solution architecture. First reason is that it is less expensive and easier to define multiple alternatives of potential solution, which allows the engineering team to eliminate the non-viable solution in the early stage. It also offers the possibility to hide variants that are not primary importance. The abstract level of architecture also provides the engineers with the option to search for an existing component from other system to satisfy the need.

4.2.3.1 Definition of the behaviour principles of the system

The major objective of the phase is to define the principles of the desired behaviour of the system and considering all the constraints (non-functional requirements) defined in the system analysis. All the solutions defined in this phase must respond to the constraints under certain operational conditions.

The first task is to identify the behavioural response to the needs identified in the previous phase. For example – The system function in the SA stage ‘’ must be satisfied by the function in the LA stage. There could be cases where two functions in the SA stage would be described by three functions in the Logical analysis.

Finally, all the functions and the components must remain in a description level to highlight the major design choices. For example – In the figure, a component Sensor has been used to describe the function ‘Measure temperature’. But it doesn’t explain the sensor type and configuration of the sensor that is been used. Therefore, the detailed specification of the sensor will be handled in Physical architecture. Logical architecture doesn’t hold the details and specifications of the sub system or a part[20].

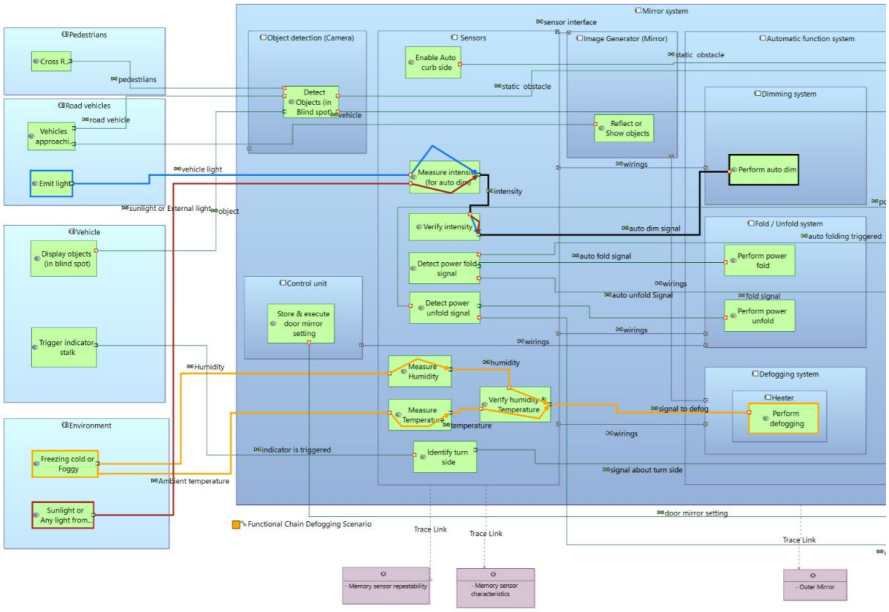

Figure 4.5: Logical Architecture diagram

4.2.4 Physical Architecture

Physical architecture deals with the detailed solution development process. The solutions defined at this phase should be detailed enough for the development teams to implement the solution and carry out the validation and verification process in further phases.

The design approach for PA holds the same objective as the logical architecture except that this will be the finalised architecture of the product or a system. The functions that are intended to be satisfied by the components are mentioned in the respective behavioural components. The constraints defined in the previous phases must be considered at this phase in order to eliminate few solution variants and to build an efficient solution. For instance –Corrosion resistance, Impact resistance,

etc. Other constraints related to the manufacturing and implementation should be introduced and tracelinks must be established to its corresponding model elements.

According to Jean Luc Voirin [20], the main tasks that should be carried out at this phase are as follows,

- Defining the structuring principles of the architecture and behaviour
- To detail and finalize an expected system behaviour
- To build and finalize one or more finalized system architecture
- To select, complete and justify the retained system architecture

4.2.4.1 Defining the Principles of the Architecture

One of the major objectives of the PA is to minimize the complexity through rationalizing. One of the common means is to identify the solution having similarities within each other. This allows the engineers to reuse the components as often as possible in different contexts or in the case of redundancy. For instance – Considering there are two different functions (One for measuring the temperature and other for measuring the humidity level) that needs to be performed. The system engineers can identify two different solutions to implement two different functions. On the other hand, the same solution could serve as a component choice for the two different functions.

Another way to reduce the complexity is to separate the functions inside the part containment within the PA architecture. For instance – Consider the case of temperature sensor implementation for defogging functionality. In this case, the sensor should measure the temperature and verify the temperature against the nominal value. It is a best practise method to model two separate functions representing the operations inside the same behavioural component which corresponds to the physical component. The same concept could also be implemented in the case of separating the functional implementation part of the sensors from data processing part.

Another criterion is to consider the industrial point of view, the components could be separated based on the nature. For instance – Hardware components, hydraulic and electrical sub systems. All the above-mentioned components belong to a specialized domain, so it is essential to create each domain components as a separate subsystem, so that the respective teams can work distinctly without any delay or dependencies. Therefore, all the sub systems can be consumed in the main system. The concept of subsystem will be explained in a detailed way in chapter 5.

4.2.4.2 Defining the Physical Architecture

As mentioned earlier, the objective of the architecture is to define the expected behaviour of the system in detail so that the implementation could be carried out without any interruption. All the component identified at this stage should comply with the constraint definition mainly non–functional constraints. The development approach is quite similar to the approaches carried out in the previous phases.

Note: The Physical architecture is not just the refinement of the logical architecture, which means it should be detailed enough in explaining about the deployment conditions of the manufacturing techniques or the technologies. The interfaces i.e., physical links should be defined accurately since manufacturing engineers rely on the details for further development or supply in case of ordering a component from external suppliers. The nature and the data transferred from one function or component to another are captured with the help of Exchange items. Considering a functional exchange wire harness, this characteristic of the exchange could be defined as shown in the figure 4.6.,

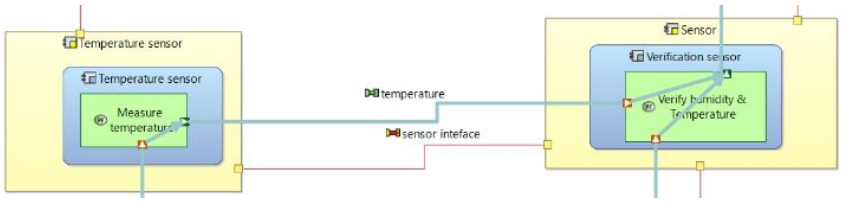

Figure 4.6: Functional Exchange and Physical Link

The concept of physical components in Capella is represented by two elements., behavioural components (blocks in blue) and node physical components (blocks in yellow). All the physical functions should be modelled in the behavioural components. Capella provides an option for an engineer to define the type of component. A behaviour or node component could be classified as any of the below component type [22].

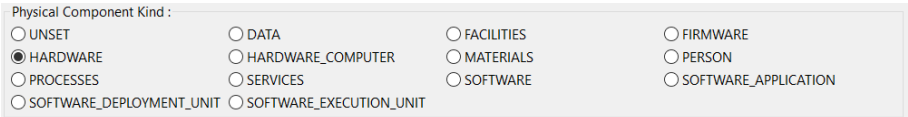

Figure 4.7: Component classification Types available in Capella

The node components are linked using an element called Physical link. This physical link supports as a communication element between the two components and it also serves an interface in Physical Architecture(PA). The figure below shows a physical link -wire harness between the components glass and steel plate. Another important aspect of the PA is the product breakdown structure, where the complete product components and its hierarchy can be defined. The complete PA diagram for door

mirror is attached in appendix.

5

Use case Implementation

The Chapter contains a list of use cases that has been formalised in order to validate the MBSE and SMW.

5.1 Use Case 1: Create Product Model

5.1.1 Description

In this use case, the system engineer defines the stakeholder requirements and other product development activities in the form of project model using ARCADIA methodology. The tracelinks must be established between the models defined in various phases, All the model elements created must be published to the Teamcenter.

5.1.2 Purpose

The main purpose of the use case is to enhance clearer communication and transparency amongst product development teams. Although the product development activities mentioned in the textbook and articles are sequential and systematic. The actual process carried out in the industry is not always same. This is mainly because of the constraints that exists in company. The major constraints are cost and time. This forces the engineers to fast track the concept development processes and finding the readily available solution in the market at that point of time. Although this helps the business to arrive with the easy and cheapest solution, there are serious consequences in missing certain phases of the development activities. For instance - Chances of missing an innovative next generation solution. On other hand, in some cases even though the business ends up in finding the right solution, the traceability between the different activities of product development are not established. This creates a serious impact in a product which has components from cross domain platforms. Another drawback in the system is due to conflicting requirements. For instance – Considering a system which has technical requirements - Fuel efficiency and Performance. In this scenario, it would be contradictory to have requirements stating high performance and high fuel efficiency. These Conflicting requirements must be identified in the early stages of product development. But these redundancies are unnoticed due to missing tracelinks between requirements and components.

Identifying these errors in the later stages is time consuming and costly. The product models will help in identifying these redundancies in the early phases.

5.1.3 Expected outcome of the Use case

The outcome of the use case should be project model elements. The interfaces between the system and its sub-system should be clear and distinct. The functional, component and interface realization should be properly established. The models should be self-explanatory for the readers and the engineers in the relevant department.

5.1.4 Use case Implementation using MBSE

Below is the list of activities that are performed in SMW/Capella as part of the creating project models. First and foremost activity in product development is to identify the customer and stakeholder needs. After collecting the needs, it should be stored in the form Operational Capabilities in SMW. Also, the Operational actors associated to each capability must be identified and established in the form of Operational Entities and Operational actors. Once the actors and entities related to the capabilities are identified and related, the next step is to identify the activities for the Operational Actors and the entities. For each Operational Capabilities, the sequence of activities is identified, and the tracelinks are established in the form Operational chain in Operational Architecture diagram.

After completing all the operations in the Operational Analysis stage, a functional transition must be performed to system analysis. This Operation should create system function for all the operational activities created in the Operational Analysis stage. All the capabilities, entities and actors defined in the Operational analysis stage should create a system capabilities, entities and actors after performing functional transition. The user should be able to add new system actors, capabilities and functions. Similar to the Operational chain created at the Operational stage, the user should be able to create functional chains at this stage. The non-functional requirements must be created as constraints. If the Operation is performed in System Modeling Workbench, then the non-functional requirements are created as a requirement in Teamcenter and it must be mapped to the relevant level.

After creating all the model objects, a functional transition must be performed. This transition must create the logical objects relevant to the system objects. The functional transition must create the logical objects automatically without any external user action. The user should be able to create logical component to define the high-level solution. And the last level is Physical architecture. At this stage, the user should be able to create new physical component and should be able to utilize the existing physical component created at the time transition. The interfacing components connecting the physical components are created in the form of

physical links. The requirement must be associated to any stages of the architecture depends on the placement. System Modeling workbench provides options to save the project models in the workspace as well as in Teamcenter. Publishing the models to Teamcenter enhances the information flow between the different domains. The saved project model items in Teamcenter can be viewed and validated through workflow process.[23] [20].

5.2 Use Case 2: Identify Requirements and relate them to correct model elements

5.2.1 Description

After analysing all the stakeholder requirements, it must be transformed into a technical requirement or specifications. These technical requirements must be mapped with their respective models. This use case demonstrates about managing the technical requirements and mapping it to the product models. The system engineer should be able to understand the technical requirements and its impacted models and related them together. Ex Functional requirements should be related to model functions.

5.2.2 Purpose

As discussed in the previous chapters, the broken link between the requirements and its relevant components creates a major impact in a product. Considering a product is encountered with a design flaw. In that case, a requirement changes forces the engineers to identify the part impacted by the requirements. In the existing document-based systems engineering, identifying parts corresponding to a requirement is purely based on experienced professionals. In this case, there is a chance for human error. Even though the engineer identifies all the relevant parts associated to the requirement, it would be a time-consuming process. This creates a need to have a solution which provides integration between the technical requirements and its parts.

5.2.3 Prerequisites

To evaluate and perform the use case, it should be made sure that the requirements are available in Teamcenter. Teamcenter Active workspace (AWC)is the recommended interface to establish tracelinks. All the model elements must be created in SMW application.

5.2.4 Use case Implementation

After creating and publishing the model elements in SMW, the next step would be mapping the requirements with the model elements. The user must read and

NOTE-TREG OUTER MIRROR	000110	A
1 Legal requirements	000111	A
1.1 Outer Mirror	REQ-000068	A
1.2 Direction Indicators	REQ-000069	A
1.3 Lamp functions	REQ-000070	A
1.4 Certification / Homologation	REQ-000071	A
2 Functional environment	000112	A
2.1 Functions	REQ-000078	A
3 Environmental Requirements	000113	A
4 Property Requirements	000114	A
4.1 Percieved Quality	REQ-000007	A
4.2 EMC requirements	000124	A
4.3 Scratch resistance	REQ-000013	A
4.4 Impact strength at low temperature	REQ-000014	A
4.5 Resistance to Chemical	REQ-000015	A
4.6 UV resistance	REQ-000016	A
4.7 Short term heat test, exterior	REQ-000017	A
4.8 Long term heat ageing test, exterior	REQ-000018	A
4.9 Resistance to climate variation	REQ-000019	A
4.10 Humidity	REQ-000020	A

Figure 5.1: Requirements structure in Teamcenter Active workspace

understand the specified technical requirements. The user must analyse the model elements that are impacted by the requirement. For instance – A functional requirement must be mapped to the relevant function in the system Analysis. The Figure 4.2 shows an example which shows a function defogging and its corresponding functional requirements[23] [20].

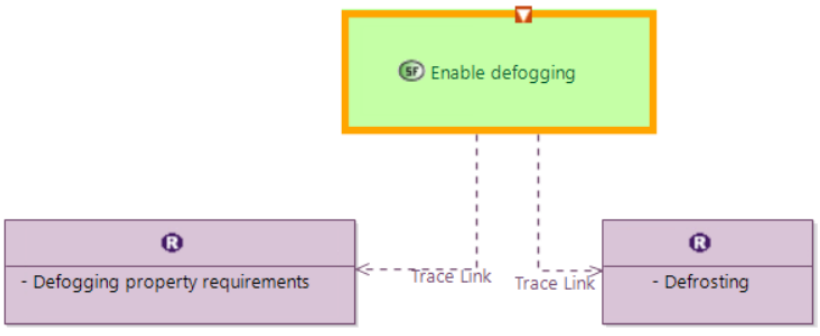

Figure 5.2: Functional Requirement mapping

The Product specific requirement should be mapped in Physical architecture. In other words, any requirement which is associated to the solutions of the stakeholder needs. Considering the Rear-view Mirror, turning indicators is one of the components. The requirements associated to the directional indicator must be mapped in

the physical architecture level.

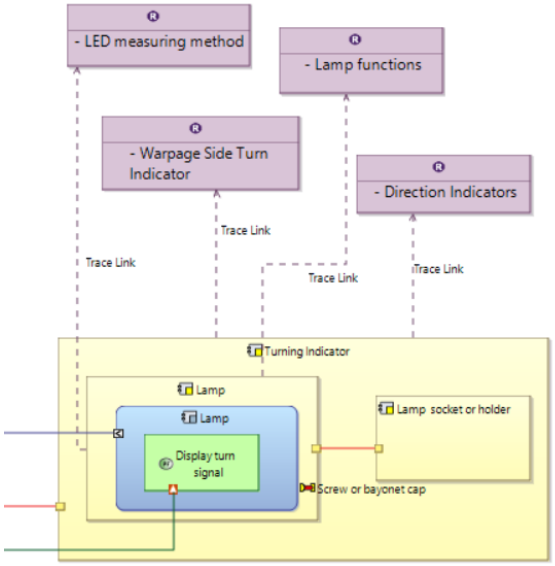

Figure 5.3: Product specific Requirement mapping

In the figure – PA, another important detail to note is the connection of the turning indicator with the other components in rear view mirror. The Red Line shows there is an interfacial component. The Green line is the functional exchange from an external function to Display turn signal function. The functional exchange could be any form of information exchange from the source.

Similar to above two requirements, non-functional requirement which is also known as constraints must be mapped to its respective system or sub system. For example – In our system of reference (rear-view mirror), constraints are corrosion resistant and impact resistant. The constraints should be mapped to the models in the system level architecture. Mapping the constraints or non-functional requirements at this level promotes the system engineer to arrive in a precise solution.

In order to map the requirements to the model elements, Active workspace window must be opened in SMW interface. The Active workspace view could be opened using window-> Show View option. This opens Active workspace window adjacent to the SMW window.

The user can drag and drop the requirement to the relevant model elements and perform mapping. The trace link is established between the requirements and product models after drag and drop. Trace links are a directional relationship between any two objects. The objects could be requirements, models in functional architecture, logical architecture and physical architecture phases. One Source object can connect to multiple target objects. The concept behind trace link is that, the source object

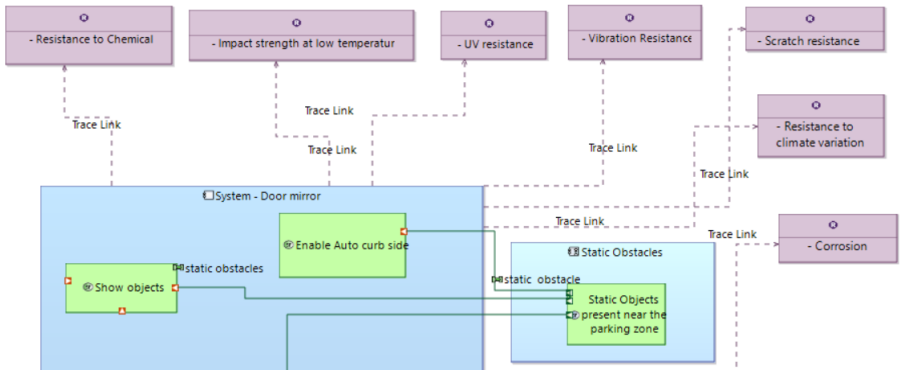

Figure 5.4: Non-functional Requirements mapping

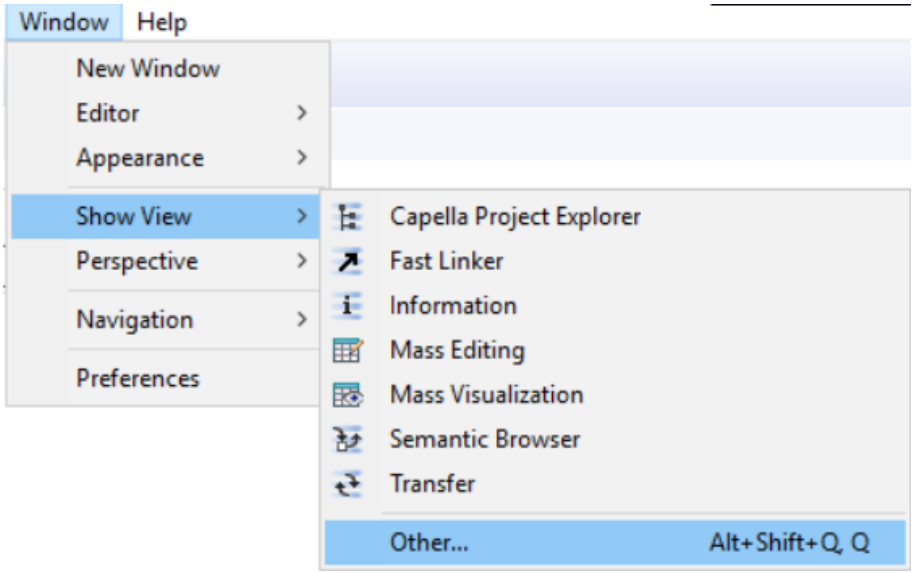

Figure 5.5: Show View Option

must comply the conditions defined in the target objects. In the figure, System - Door Mirror is the source object and all the requirements are the target objects it must comply with. If the user tries to map the requirement that has been already mapped to another requirement. In that case, the drag and drop option feature will not support mapping. The PLM Requirement Link option which is available in palette must be used to map the requirements.

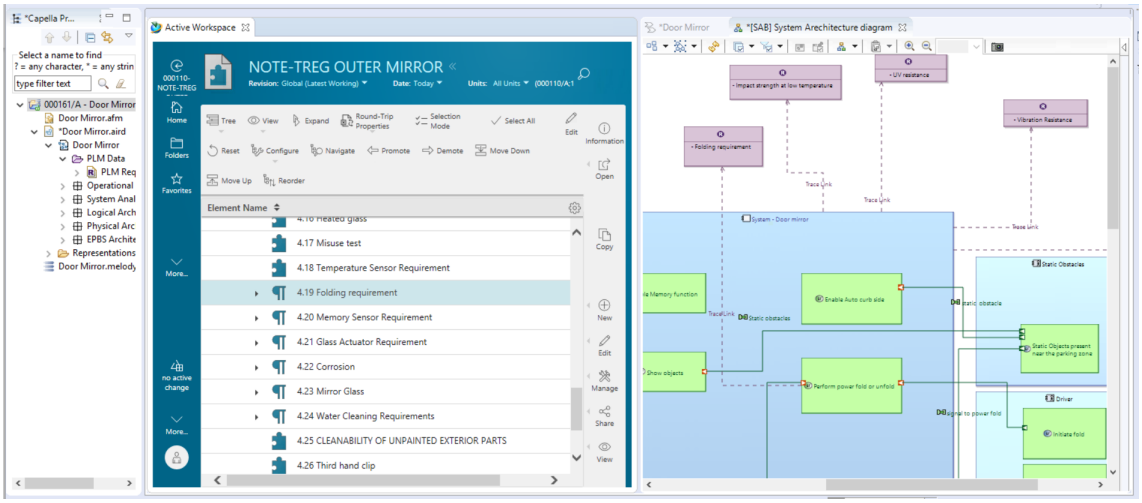

Figure 5.6: AWC view in SMW

5.3 Use Case 3: System to Sub System Transition

5.3.1 Description

The use case promotes modelling sub system as individual model projects and establish a connection between the system and sub system.

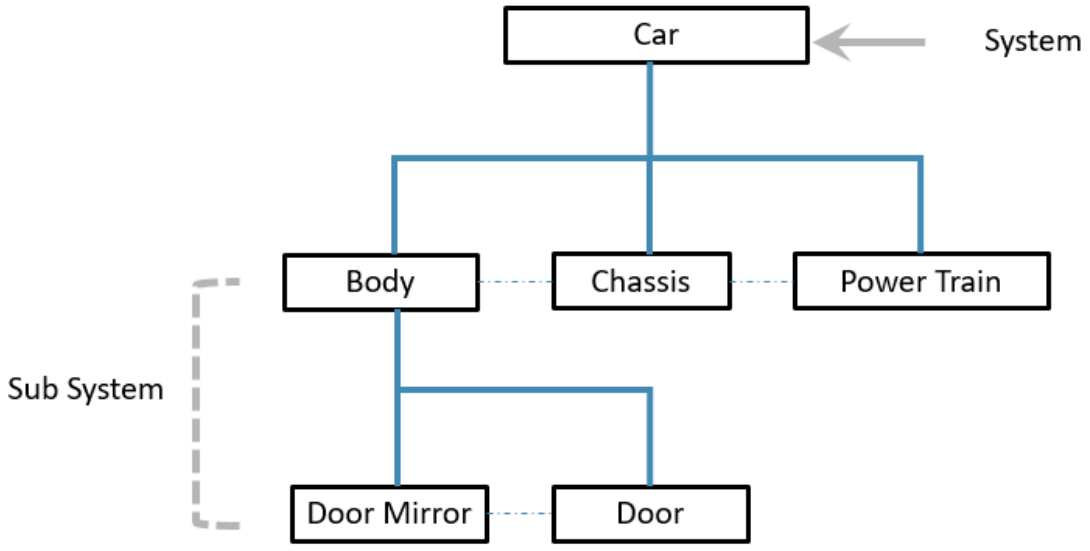

Figure 5.7: Vehicle System- Sub system Structure

5.3.2 Purpose

A product is a combination of many sub system and components. Consider modelling a product in an Automotive industry, it is a tedious task to model all the sub

systems in a single model. So, there arises a need to model each sub system separately to reduce the complexity. Even though the primary reason is the complexity, there are other reasons which forces to divide a system into multiple subsystem and model projects.

One of the reasons is the varied domains. There are different domains in the industry that works independently. In that case, modelling a product as a single system will create dependencies amongst various teams. These dependencies eventually increase the lead time of the product.

Another factor is component or sub system reuse. The same sub system or component is utilized by different variants of the product. In this case, creating sub system model project for each variant is unnecessary and time consuming. The easiest way is to create a single sub system model projects and to consume the subsystem in many different product variants (system).

Another factor is time and complexity. If the whole system (eg -car) is modelled as a single project model, the architecture becomes complex and a time-consuming process. It also restricts different departments to work simultaneously due to certain access restrictions that exists inside the company.

5.3.3 Prerequisites

All the sub systems must be modelled as a separate library project. The actors and other model elements must follow a standard naming convention. For instance – Considering an actor – Road Vehicles. All the sub system utilizing this actor must use the same naming conventions. The sub system and system projects must be available in the same working directory.

5.3.4 Limitations

The limitations mentioned here corresponds to SMW version. Only part of the functionality of the use case is available in SMW version 4.0. So, in order to achieve the entire functionality in SMW 4.0, minor customization must be performed in the project file and melody modeller file. The customization changes the Capella project to a library project. More information about the library project will be discussed in the next section.

5.3.5 Working Principle

All the project models are created as Capella projects irrespective of a sub system or system. If the sub system is reusable in different variants of the product, then the

sub system should be modelled as a Library Project. The Capella project can be converted into library project by clicking an option in SMW. But this transforming option is not available in SMW version 4.0. The Red colour icon in the Project name denotes that the project is a library project.

Once all the projects and libraries are made available in the workspace, the desired Model project of the system must be opened, and the sub system libraries should be linked in the references. After adding the libraries to the references of the project system, the sub system is ready to be consumed in system.

For all the subsystem library project, REC must be created, and the access restrictions must be established. A REC is a definition of an element which could be used in multiple contexts or configuration models. According to Jean-Luc Voirin [20], a Replicable element collection (REC) is a set of model elements, identified as being a pattern for building Replicas (RPLs).

The sub system transition could be implemented in two different scenarios.
Scenario 1: Door Mirror Sub system in a Car Assembly.

In this case, Car is considered as a main system and rear-view mirror, chassis and power train are considered as sub systems. All the sub systems must be created as a library projects and the main system should be modelled as a Capella project. The sub system libraries should be able to be consumed in the main system as a reference.

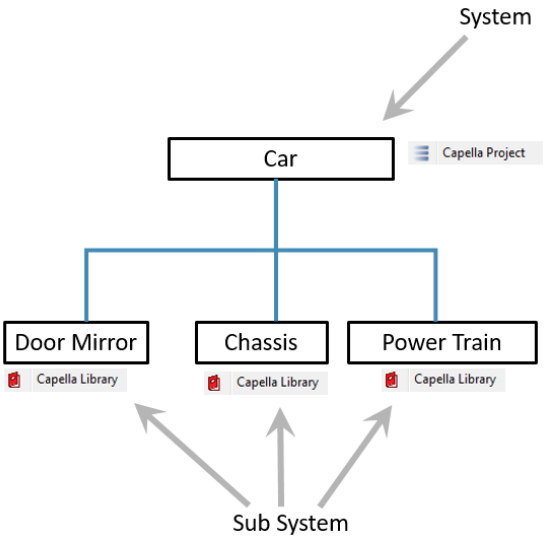

Figure 5.8: Sub systems in an Assembly

Scenario 2: Sub system consumed in Multiple Assembly. In this scenario, the same subsystem (Door mirror) is used in multiple systems or product.

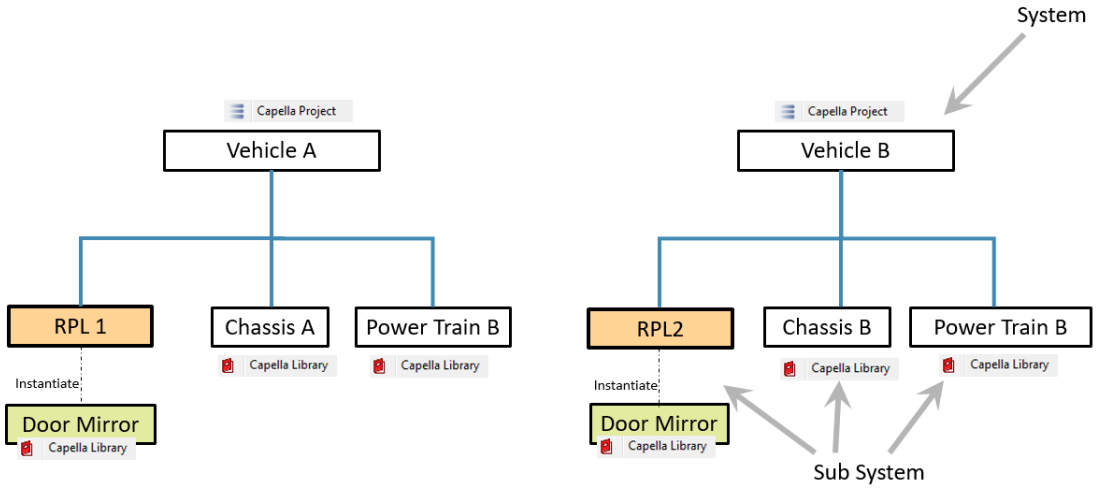

Figure 5.9: Sub system used in multiple systems

Multiple instances of REC with different access restrictions should be able to be created. Each instance of the Replicable Elements collection a given context or configuration model is known as Replica (RPL). The instance of the sub system (RFL) should be able to be utilized in other systems now. To consume the sub system, a replica (RFL) of REC must be instantiated. This could be done using an option 'Instantiate RPL from REC' by right clicking on the physical architecture diagram (PAB) of the sub system or system. Once the RPL is instantiated, the sub system is imported by using Node PC option.

The figure 5.9 shows Vehicle A and Vehicle B are using the same Door mirror sub system, but different replicas. It shows that Vehicle A and B has not consumed all the model elements from Door mirror but only certain model elements which are necessary for the respective system.

5.4 Use Case 4: Handling Requirement Change

5.4.1 Description

In certain cases, original requirements may not be able to fulfil the product needs due to design restrictions. This creates a requirement change and the corresponding design change must be performed on the parts impacted by the requirement. There arises a complexity in identifying the parts impacted by a requirement change. In a complex product, it becomes a tedious task to handle a requirement change without traceability.

5.4.2 Purpose

One of the important process of the requirements engineering department is requirements management which is responsible for managing the changes to existing system requirements. The new requirements or change in the existing requirements emerge due to errors in the system or due to change in external circumstances. The changes in the requirements must be documented and controlled formally in order to track the impacted parts. But there are chances of missing the primary and secondary parts impacted by the requirements.

Based on an article on Expressen.se [24], Volvo cars recalled around 750,000 vehicles after discovering a sensor failure in all Volvo car variants. The reason for the sensor failure was the missing software code which has been failed to be updated. So, during this situation, it is a necessity to have a traceability between parts and requirements, to identify the hardware belonging to the software component or vice versa. On other hand, this failure could have been avoided, if there is a traceability was ensured between the hardware, software and its requirements.

5.4.3 Prerequisites

To evaluate and perform the use case, there are certain prerequisites that must be fulfilled. The requirements should be managed in Requirements management module in Teamcenter. The corresponding model elements of a product must be available in SMW interface. The Teamcenter must have integrated Active workspace environment.

5.4.4 Use case Implementation

The use case helps in identifying the parts impacted by a specific requirement. The prerequisites must be satisfied before executing the use case. There are two ways of identifying the parts impacted by a requirement change. One way is using a semantic browser window in SMW and other method is using a traceability matrix.

5.4.4.1 Steps Followed to identify the Impacted Parts

- Identify the Function corresponds to the Requirement (Eg-Auto Dim)
- Identify all the requirements linked.
- Identify the parts associated to the requirements using Semantic Browser or Trace link Matrix.

5.4.4.2 Using Semantic Browser

Semantic browser is an effective tool in identifying the tracelinks between the model elements at any phase of the architecture. In this case, a requirement related to Defogging functionality has been chosen.
Step 1: The relevant functions correspond to the defogging must be identified.

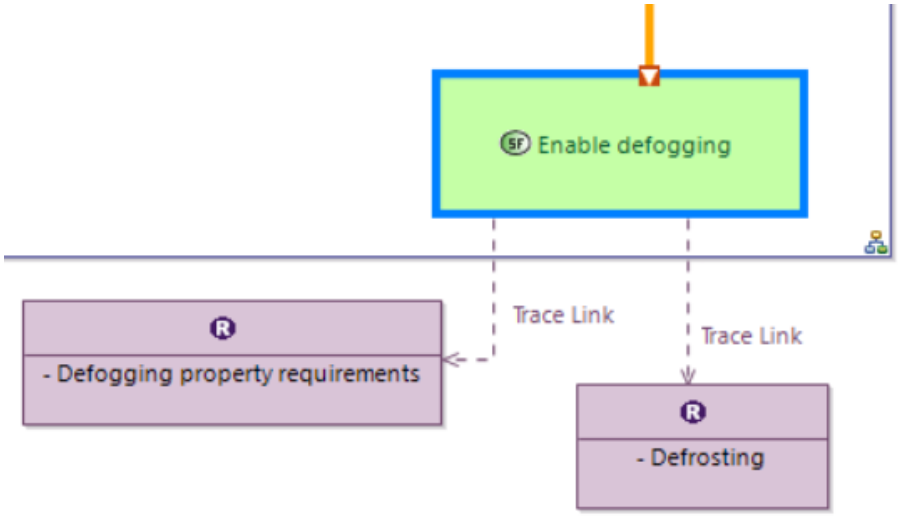

Figure 5.10: Defogging Functions and Requirements

Step 2: Select the Function and go to the Semantic browser to identify all the realized functions and components. From the semantic browser, the realized logical function and component should be able to be identified.

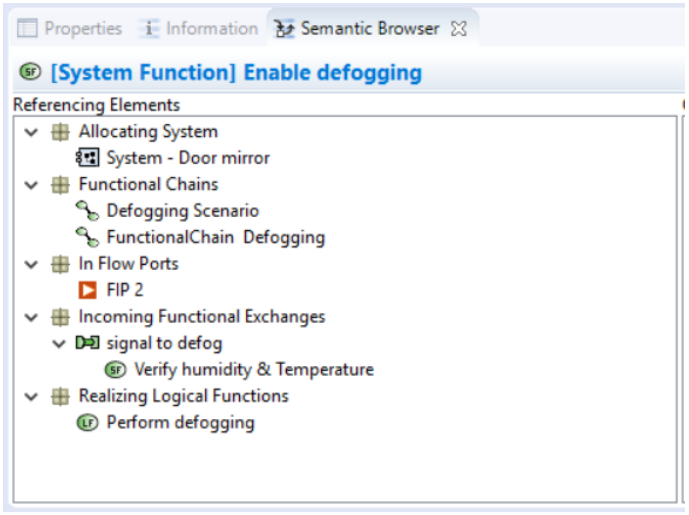

Figure 5.11: Semantic Browser

Step 3: Select the identified logical component to find its corresponding physical components.

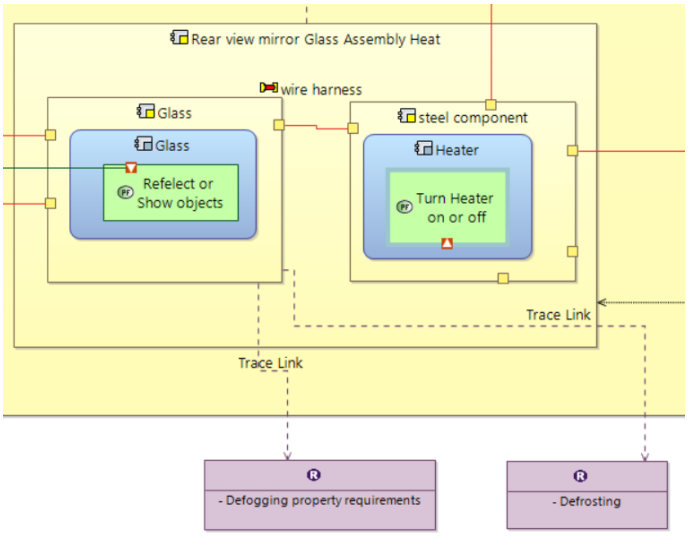

Figure 5.12: Defogging functionality - Physical components and Requirements

5.4.4.3 Using Trace link matrix

Trace link matrix is one of the easiest ways to identify the impacted components. The first step followed is exactly similar to the previous method i.e., identifying the functions and its associated requirements. After identifying the requirements, a trace link matrix should be generated between the requirements and the system block revision. Here comes a basic question. What is System Block Revision? System blocks are the items in Teamcenter. Also, System blocks are the replica of the physical architecture model elements in Teamcenter. The trace link matrix shown in the figure, explains the requirements and its associated sub systems. The figure explains the tracelinks between the technical requirements of the rear-view mirror and its components. From the figure 5.13, it is interpreted that technical requirements (color tolerance , gloss tolerance and Grain requirements) are mapped to Rear view mirror Skull cap and Rear-view mirror Wo Cap sub system.

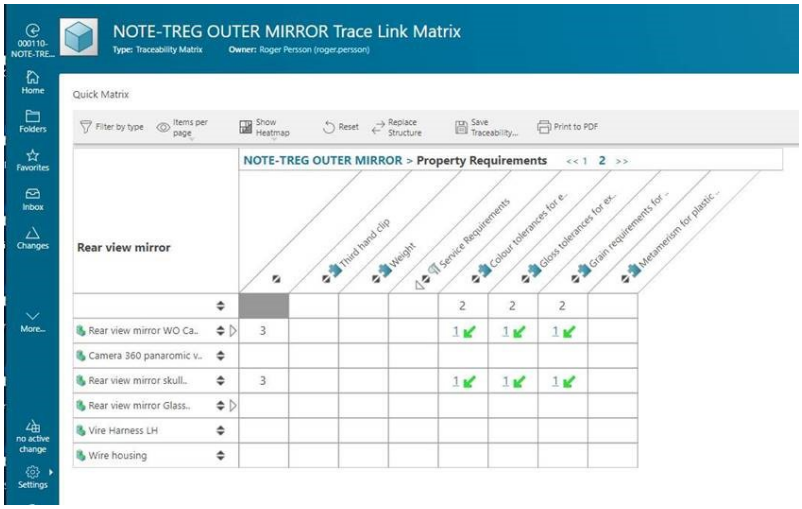

Figure 5.13: Trace link Matrix

5.5 Use Case 5: Parameter management in TC and Models

5.5.1 Description

Most of the requirements has a parameter which is a measurable attribute. Parameter management provides options to manage the parameters associated to the requirements. The use case explains about defining parameters for the requirements and relating the parameter objects with the SMW & Capella models.

5.5.2 Purpose

The main purpose of managing the requirement parameters as a property is to eradicate documenting the parameter values. In the exiting product development processes, all the parameters are documented in the form of pdfs. Defining the parameters as models also reduces the retrieval time of the information. Also, parameters plays a major role during Validation and Verification. Considering a case where Engine has an attribute Fuel Efficiency. During System development, the product will be developed in such a way that the system meets a fuel efficiency target.

The results from Validation and verification are stored in the form of documents and the results are unnoticed until the simulation engineer sends the result to the respective team. With MBSE, storing a parameters as properties helps in faster interpretation of results. It also helps the respective teams to analyse the simulation values with the achievable value or target value. Thus, parameter management enhances faster communication & collaboration between teams. It ensures simple and secure order control which allows flexible job management.

5.5.3 Limitations

The limitation mentioned here corresponds to SMW. The entire functionality of parameter management is not supported in SMW version 4.0. The use case could be executed only in the higher version of SMW i.e., SMW 5.0 and above. The SMW version 4.0 supports creation of parameter objects in Active workspace but the integration between SMW objects cannot be established.

5.5.4 Prerequisites

A Requirement which consists of different set of parameters must be available. All the requirements must be managed in Teamcenter.

5.5.5 Use case Implementation

Since the current SMW version 4.0 doesn't support the entire functionality. Only the working principle will be demonstrated in this section.

First step is to create the parameter objects as part of the Capella projects. The corresponding model element for the parameters in SMW are known as Exchange items. Second step is to define the parameter definition in Teamcenter. The properties (Exchange items) defined in SMW will be created as a parameter in Teamcenter if the property complies the criterion of the parameter definition. In the figure 5.14, Defogging functionality is considered and the parameters for the defogging are modelled as exchange items in SMW. The parameters are temperature and humidity and time. The unit for the parameters can also be defined at this stage.

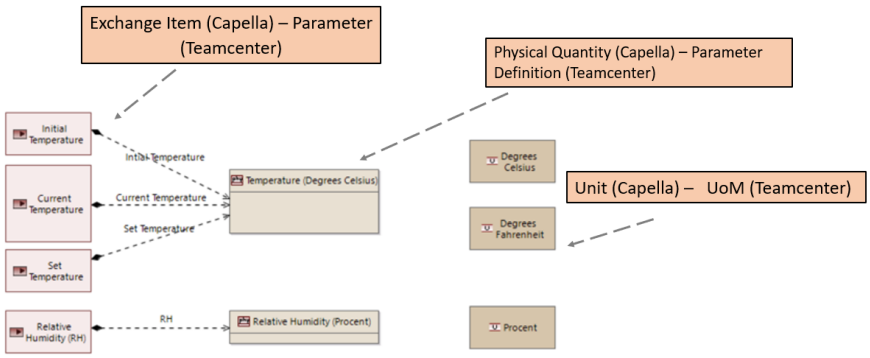

Figure 5.14: Parameter objects in Teamcenter and Capella

As mentioned in the purpose, the parameters play a major role in validation and verification. Considering the case of validating the Defogging functionality. A Verification plan is created to validate defogging functionality. All the requirements and test methods associated to the function should be linked to the verification plan. The test methods for Defogging states that mirror should be cleared from fog in 3 minutes. So, this achievable time is captured as a target value. The testing result is captured in a parameter after validating the functionality. This parameter values helps the simulation engineer to verify whether the system meets the testing requirements.

5.6 Use Case 6: Identify Conflicting requirements

5.6.1 Description

This use case describes on identifying the conflicting requirements in the projects.

5.6.2 Prerequisites

In order verify the working principle, parameter management must be installed in Teamcenter and SMW.

5.6.3 Working Principle

This section only describes about the working principle since this use case cannot be verified completely without parameter management module. Considering there are two conflicting technical requirements, Fuel efficiency and Horsepower. In this case, in order to achieve high horsepower, fuel efficiency must be balanced. Identifying this trade off in the early stage is crucial in multi domain products. Upon implementation of the parameter management, an individual set of parameters define the requirement description. Linking these parameters with the respective requirements provides a clear view to the system engineers. Since associating the requirements to the models enables all the development teams to view and verify the parameters. The transparency in the models and requirements with parameter trace links helps in identifying the conflicting requirements early in the development cycle.

5.7 Use Case 7: Requirement Fulfilment Dashboard

5.7.1 Description

This use case describes about showing the requirements completion in the form of dashboard view.

5.7.2 Purpose

During product development it is needed to get a view of current status and progress. How well product components fulfill specified and agreed requirements must be actual and visible all time.

5.7.3 Limitations

Requirement fulfilment dashboard view is not available in SMW or Capella. But a dashboard view can be generated in Teamcenter. This requires a separate module to be installed in Teamcenter. The module is called Teamcenter Reporting and Analytics (TCRA).This functionality requires Requirements to be managed in Teamcenter.

5.7.4 Working Principle

All the technical requirements must be defined or imported in Teamcenter.A milestones or a deadline must be set for the requirement completion. And the next step is to associate all the test methods to the respective technical requirements. Once the tracelinks are established between requirements and test methods, verification request is created to validate the requirements. Based on the status of the verification request revision, the requirement completion status will be updated. The

Figure 5.15: Working Principle

requirement completion will get updated in the form of bar or pie charts. By viewing the dashboard, the development teams can get a quick update on requirements.

Associate the relevant test methods and Verification plan to the technical requirements. Upon test completion, based on the status of the analysis request revision, the requirement completion status will be modified. For instance- If the verification request revision has been created and workflow initiated for defogging functionality,

5.8 Use Case 8: Automated test and Validation of Requirements

5.8.1 Description

Validation is one of the key elements in assuring the quality of the product. The use case explains about the validation of the product in early stages of product development. This validation helps the system engineer to ensure that the components developed fulfils the requirements defined.

5.8.2 Purpose

In most businesses, design verification or testing is performed in the final phases of product development. Even though the approach followed is traditional, some components goes unnoticed while testing due to the missing traceability between the requirements and parts. This leads to the costly changes in the product development. Due to these reasons, design verification must be performed in the early stages of product development to avoid costly changes in the end.

5.8.3 Prerequisites

The model elements corresponding to the product structure must be available in the SMW. Trace links must be established between the model elements and requirements.The Operational process must be modelled to the corresponding functional

chain that has been desired to be modelled.

5.8.4 Use case Implementation

Functional Chain is an efficient tool which is used to describe the product behaviour for a particular functionality. Functional chains must be created for each system capability for validating the requirement fulfilments. Functional chain could be modelled in any stages of the Arcadia architecture except Operational Analysis. After modelling functions, functional exchanges and other components, the user must select the capability for which the functional chain must be created. Once the Capability is chosen, identify the functions and other model elements corresponding to the capability. For instance – Considering Auto Dim functionality for the purpose of demonstration.

The system engineer must identify the sequence of functions in the System Analysis stage and create a functional chain. After creating a functional chain in each stage, it must be linked to realized functional chain. In SA stage, the realization functional chain must be the functional chain in Logical Architecture. The same process should be followed in creating functional chain in other phases.

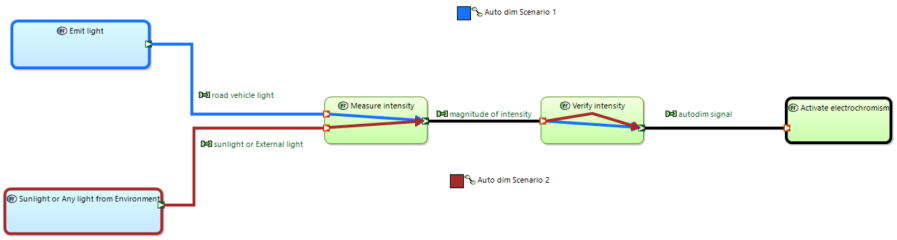

Figure 5.16: System Architecture - Functional chain Diagram of Auto Dim

For the above functional chain in System architecture diagram, the corresponding physical architecture diagram representing physical components is shown below,

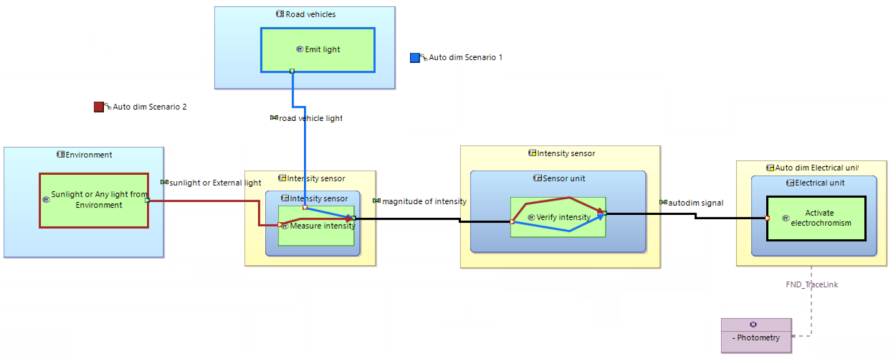

Figure 5.17: Physical Architecture - Functional chain Diagram of Auto Dim

In order to perform verification or validation for the chosen functionality i.e, Auto dim, the system engineer must verify the functional chain for requirement fulfillment, design completion and part release. From the functional chain, the system

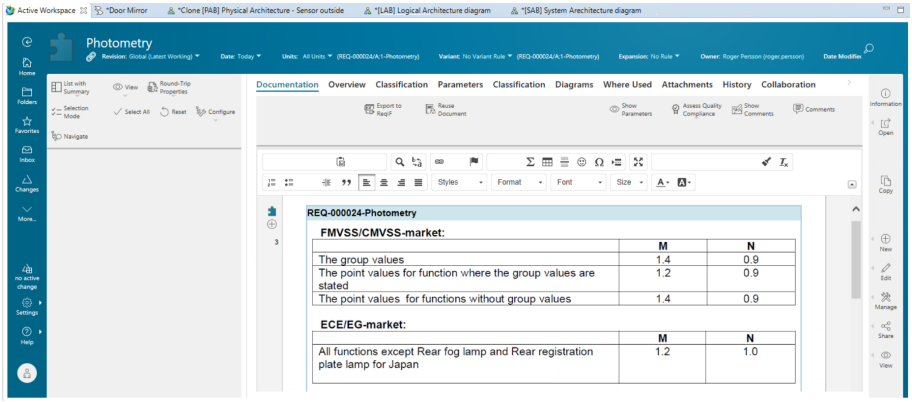

Figure 5.18: Requirement Definition

engineer must be able to understand all the components associated to the functionality. Further, the components could be verified for requirement fulfillment by viewing the requirements description in Teamcenter AWC. The requirements can be viewed in Teamcenter using 'View in Active workspace option'. This opens the AWC window with the requirement description and parameters. The figure 5.18 shows the description and parameters for a requirement 'photometry'.

5.9 Use Case 9: Solution Variant as Subsystem SMW Project

5.9.1 Description

Increasing Customer expectations drives the product manufacturers to offer more variants of the product. As the product variants increases, so does the complexity. The use case focuses on implementation of variant management on model-based systems engineering models.

5.9.2 Prerequisites

All the model elements must be available in SMW before executing the use case. The models must be exported to Teamcenter and the corresponding system blocks must be properly available.

5.9.3 Limitations

The variant management functionality is not supported in the existing version of SMW i.e., 4.0. The functionality is expected to be available in the upcoming versions of SMW.

5.9.4 Purpose

Variant management is one of the key aspects in the automotive industry due to the increasing variability in the products. Increasing diversity and complexity of the product makes the business process more challenging, which makes it difficult to manage the product variants. The variant management functionality is available in PLM systems. This creates a need to configure the product variants in MBSE platform.

5.9.5 Use case Implementation

The outcome of a use case will be a configured model from different product variants. Since the current version of SMW doesn't support this feature, this section will only discuss about the working principle. The use case is executed by following steps.

- After publishing the product models in Teamcenter. Systems blocks are created for all the product models in SMW.
- System engineers can define the variant conditions for the system blocks in Teamcenter.
- Variant rule for the system block elements can be created.
- After defining the variant conditions to all the model elements, the models could be configured by applying the Variant rule.
- After applying the Variant rule to the model, it configures the system blocks in Teamcenter.

This variant configuration configures the system blocks in Teamcenter as well as the product models in System Modeling workbench.

5.10 Use Case 10: Validate SMW models and manage divergency

5.10.1 Description

Mistakes are bound to happen during modelling project models in SMW. Even though the product is modelled and the tracelinks are established, the models and the associated links must be validated to ensure there are no dummy models or duplicate model elements. In order to overcome the errors, a validation of product models must be performed. The use case explains about validating the product models to identify the conflicts and divergency between functions, requirements and other model elements

5.10.2 Prerequisites

All the model elements should be available in SMW before executing the use case. The requirements must be linked with the model elements depends on the validation the user is performing,

5.10.3 Implementation

The outcome of the use case is to determine the missing models and the relation between the model elements. After completing the modelling, the user must select the phase that needs to be validated. For instance - Right click on the System Analysis and select Validate Model. This validates the model and its relations with the parent and the child model. Validating the System analysis will validate the system models, requirement integration as well as the tracelinks established from operational and logical architecture to system analysis. If any of the relation elements are missing, that will be thrown as an error or warning with a description. Other Error scenarios will be discussed separately in upcoming chapters.

The system engineers can also define business specific validation rules. For example – The naming conventions of the Actor or the system functions should start with Upper case letter. Any models that doesn't satisfy the rules will be captured in validation.

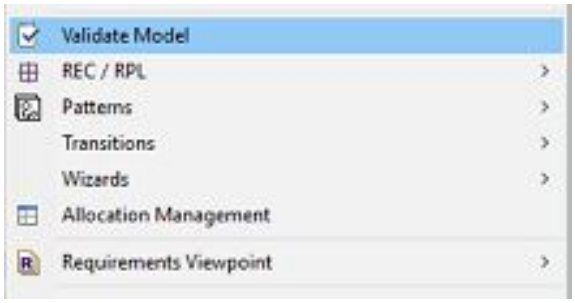

Figure 5.19: Validate option in SMW

5.11 Use Case 11: Joint construction of architecture and product variability

5.11.1 Description

In this digital era, companies face challenges to shorten design cycles, reduce development cost, improve productivity and increase product quality. Most of the companies accomplish these objectives by adopting the reuse strategies. Reuse strategies accelerate the product designs and drive cost reduction. The use case explains about reusing the project model in a new project.

5.11.2 Prerequisites

A project model should be available in the environment before executing the use case.

5.11.3 Purpose

Reusing the existing resources is followed in all platforms. Several studies shows reusing the knowledge minimizes the risk, increase the effectiveness and save time and money. Creating a project models from the scratch is a tedious and a time-consuming work. This creates a need to utilize the pre-existing templates or the existing model project. These model project provides a base platform to work and it also helps the users by reducing the workload by neglecting the rework.

5.11.4 Implementation

To perform a cloning or revise operation on a model project, system engineer must ensure that the base model project is same as the current project. In order to model a similar project with additional models, the engineer can clone the base project model. The new model will be created in PLM platform with the newly generated ID. Another scenario would be to revise the existing released project, then it could be performed using revise operations in SMW. This revises the model project. The revise operation carries forward requirements from the previous revisions, the system engineer must undergo a change process to edit and add requirements, functions and other model elements. After performing changes, the models must be published.

6

Recommendation

In this chapter, list of recommendations are discussed that must be followed upon implementation of MBSE in practice. The chapter also lists out several best practise methods that needs to be followed while modelling the components in Capella / SMW. The chapter is divided in two sections namely., Modelling recommendation and Teamcenter & Integration recommendations.

Create modeling guidelines and conventions for all system aspects, hierarchy levels, and views. Build a comprehensive model, which serves as the basis for providing different views to different engineering aspects and subsequent activities;

6.1 Teamcenter & Integration Recommendations

In order to utilize the complete potential of MBSE and its offerings from Capella / SMW, below are the things that should be considered.

- The version of Teamcenter installed in the environment should be 12.5 & above.
- System Modeling Workbench platform must be installed in client machine.
- The version of System Modeling Workbench should be 4.0 or higher
- SMW version should be compatible with the Teamcenter version.
- Requirements management module must be installed in the Teamcenter environment.
- All the relevant technical requirements must be available in Teamcenter Requirements management in the form of requirements and requirements specification.

6.2 Modelling Recommendations - MBSE

Here is the list of best practise methods that is recommended while working with MBSE.

- The Naming conventions of Function should start with the Upper-case letter. For example - Enable Defogging.
- The Naming conventions of the Functional exchanges must be in lowercase. For example - temperature.
- The Naming convention of the function should be in the format (Verb + noun). It is possible to rename as per business standards. But it is not advised to name the function name in a sentence. For instance - Measure temperature, display turn signal.
- The colour coding of the functional blocks and exchanges could be modified from the default colour coding. But it must be made sure that the colour coding followed for one model type should be followed in all the phases of an architecture. For instance – If the colour coding for the functional block (system functions) is blue, the same colour coding should be followed throughout the project.
- Deletion and Addition of same requirement to a same function or model elements at the same time will throw an error. It means deletion and addition are not allowed at the same time for the same objects. For instance – If the user deletes a function in a project, it must be saved and published to Teamcenter. And then the function can be added back, saved and published to Teamcenter. If the user tries to delete a function and add the same function again, it will throw an error while publishing the models to Teamcenter.
- When modelling a component, the system engineer must verify whether the component could be used in another product structure (or in future products). If yes, it is a best practise to model the component as a separate sub system in order to enable easy consumption. Assuming it is not modelled as a separate sub system, the engineer might be forced to create a same sub system again for a different product. This not only creates duplicate models but also increases the complexity and time.
- Also, when modelling a new component, it is a best practise way to search for an existing sub system or component which could be reused in the project models.
- As mentioned in the previous chapter, Teams for Capella add-on can be used to enhance collaborative working environment.

6.3 Implementing MBSE in a new Project

This section will explain a list of recommendations that should be considered while starting a new project.

Even though a minimum level of training is necessary for the system engineers

to enable collaborative modelling in Capella, modelling of the whole system is a tedious and time consuming process. This creates a need for a template project. A template projects are start point models which could be cloned to create new similar project models. The template project helps the system engineers to perform modelling efficiently. The templates helps the system engineers to comprehend the concepts and the model easily, ensuring the knowledge base required for modelling a system is delivered. More information on process related changes after implementing MBSE from existing product development practise will be discussed in detail in the conclusion chapter.

Modelling a system can start at any phase of the product architecture, which means that the system engineer can start defining the physical components in physical architecture and follow a bottom up approach to define the logical components and system functions. But, It is a best practise method to start with the Operational phase followed by other phases. It means that all the stakeholder needs must be defined in operational stage and its corresponding functions must be identified in system architecture phase, followed by its respective phases. If this modelling method is followed, it must be ensured that all the entities are modelled before moving to the next phase. Another best practise method is to model based on one stakeholder need at a time. It means that system engineer starts modelling with a single stakeholder need and define system functions, logical components and physical components. The same process is be carried out for other stakeholder needs. This ensures that all the stakeholder needs and its corresponding entities are modelled correctly and the tracelinks are ensured.

7

Discussion

Upon having discussions and interviews with several CEVT professionals, it is observed that it is an important criterion to have access restrictions for the product models in Capella / SMW. In the current version of SMW 4.0, the product models can be accessed by only one person which is a drawback. It means that, the product models exported by one user will not be accessed by another person at the same time. Capella provides an add-on called "Teams for Capella" which promotes team collaboration by accessing the product models by several persons at a time. Even though it allows multiple person accessing the product models. It restricts several persons accessing the same content of the product model at the same time. For instance- If System engineer A is modifying the physical architecture, System engineer B will not be able to edit the physical architecture in parallel, but it is possible to work on other models namely models in system architecture or other phases.

8

Conclusion

This chapter of the research work concludes the work with some justification of the choices made regarding the methodology used and MBSE tool utilized. It also presents the benefits of implementing MBSE in practise and also discusses few limitations that should be considered.

The thesis investigated the functional modelling approach using ARCADIA methodology in Capella / System Modeling workbench. The thesis followed most of the modelling techniques and recommendation of Jean Luc Voirin. The multi perspective evaluation was implemented to evaluate the ARCADIA methodology and SMW which includes the overall methodology and the ability to support the key product development issues.

In the beginning of the thesis work, a research question was framed. The thesis work progressed with the focus of answering the research question. The research question focused on investigating MBSE on addressing the barriers found in the existing product development process. The list of product development activities formulated as use case helped in the validation of MBSE implementation and assists in arriving in a conclusion whether implementing MBSE solves the barriers encountered. Upon investigating MBSE and Capella/SMW implementation in practise, several barriers could be solved. Barrier1 - The Relation between the requirements and components are established. Implementing MBSE promotes integration between requirements and the product models which plays a major role in identifying the components associated to each requirement. Barrier2 - Requirement change - As mentioned in the use case implementation part, MBSE helps easy identification of the part associated to the requirement especially during requirement change. MBSE implementation also saves a lot of time in data retrieval which was one of the major problems in the industries nowadays. This also ensures all the components related to the requirement are retrieved and addressed during concept development or requirement change, without any leaving any loop holes which ends up in recalls. Barrier3 - Relation between hardware and software components are established since all the product models are connected and tracelinks are established. This ensures all the hardware and software component are properly validated and promotes work collaboration between two domains. Barrier4- Parts Reuse, which is another major concern in the industries. Identification of the existing part or sub system for the new requirement is difficult in the existing system due to the lack of traceability. MBSE solves this problem by ensuring the traceability between the requirements and parts.

8.1 Benefits of implementing Model based systems engineering

This section lists other set of benefits that can be achieved upon implementing MBSE over traditional methods in practise.

8.1.1 Master product change

MBSE enables the product development team to understand and realize the impact made by each product decisions and design or requirement change across the whole system.

8.1.2 Digital Thread

Implementing MBSE integrates all the major players in the product development ensuring digital thread. The figure 7.1 explains how each elements of the product development are integrated. The digital Thread give an instant view of dependencies and provides a platform to work with an agile way of working in product development.

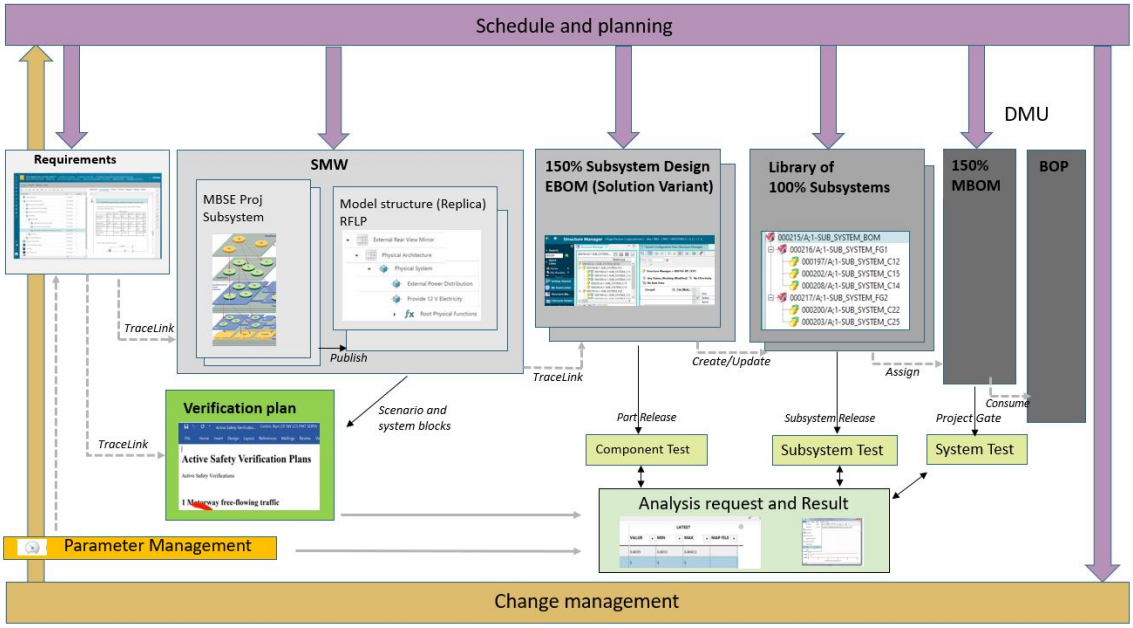

Figure 8.1: Digital Thread

8.1.3 Transparency

MBSE promotes transparency by integrating all the platforms together. There by enabling all the member of the team to continuously track and validate the design items against the specific requirements.

8.1.4 Time to market

MBSE enables collaboration among the development teams and helps in identifying the issues or problems earlier in the product development cycle. Therefore, it minimizes the rework and increases the time to market.

Bibliography

[1] Andreas Wortmann, Benoit Combemale, and Olivier Barais. "A systematic mapping study on modeling for industry 4.0". In: *2017 ACM/IEEE 20th International Conference on Model Driven Engineering Languages and Systems (MODELS)*. IEEE. 2017, pp. 281–291 (cit. on p. iv).

[2] Rajeshwari Hegde, Geetishree Mishra, and Gurumurthy Kargal. "Software and Hardware Design Challenges in Automotive Embedded System". In: *International Journal of VLSI Design Communication Systems* 2 (Sept. 2011) (cit. on p. 3).

[3] NHTSA. *Report*. Retrieved on July 03, 2020. From `https://static.nhtsa.gov/odi/rcl/2020/RCLRPT-20V144-9786.PDF`. 2020 (cit. on p. 3).

[4] Dag Henrik Bergsjö. *Engineering challenges of intrafirm technology reuse*. Retrieved on July 03, 2020. From . 2018 (cit. on p. 4).

[5] R. Prieto-Diaz and P. Freeman. "Classifying Software for Reusability". In: *IEEE Software* 4.1 (1987), pp. 6–16 (cit. on p. 4).

[6] S. M. Duffy, A.H.B. Duffy, and K.J. MacCallum. "A design reuse model". English. In: *Proceedings of the International Conference on Engineering Design (ICED 95)*. Aug. 1995, pp. 490–495 (cit. on p. 4).

[7] STA Van den Houdt. "Identifying and Managing the Success Factors behind the implementation of Systems Engineering". In: (2013) (cit. on pp. 4, 6).

[8] Cecilia Haskins, Kevin Forsberg, Michael Krueger, D Walden, and D Hamelin. "Systems engineering handbook". In: *INCOSE*. Vol. 9. 2006, pp. 13–16 (cit. on pp. 6–8).

[9] incose. *systems-engineering*. Retrieved on July 03, 2020. From `https://www.incose.org/about-systems-engineering/`. 2020 (cit. on p. 6).

[10] Shashank P Alai. "Evaluating ARCADIA/Capella vs. OOSEM/SysML for System Architecture Development". PhD thesis. Purdue University Graduate School, 2019 (cit. on pp. 6, 11–13).

[11] ISO/IEC/IEEE 15288. "Systems and software engineering–System life cycle processes". In: (2015) (cit. on p. 6).

[12] Vmodel. *Vmodel*. Retrieved on July 03, 2020. From `https://www.sciencedirect.com/topics/engineering/v-model`. 2020 (cit. on pp. 7, 8).

[13] Systems$_T$*hinking*. *Systems$_T$hinking*. Retrieved on July 03, 2020. From `https://incoseuk.org/Normal_Files/WhatIs/Systems_Thinking`. 2020 (cit. on p. 7).

[14] Morayo Adedjouma, Thibaud Thomas, Chokri Mraidha, Sebastien Gerard, and Guillaume Zeller. "From Document-Based to Model-Based System and Software Engineering". In: *Joint Proceedings of EduSymp and OSS4MDE 2016* 1835 (2016), p. 27 (cit. on p. 8).

[15] Paul Logan, David Harvey, and Daniel Spencer. "Documents are an essential part of model based systems engineering". In: *INCOSE International Symposium*. Vol. 22. 1. Wiley Online Library. 2012, pp. 1899–1913 (cit. on p. 9).

[16] vmcse. *mbse*. Retrieved on July 03, 2020. From `https://vmcse.com/category/mbse/`. 2020 (cit. on p. 9).

[17] Laura E Hart. "Introduction to model-based system engineering (MBSE) and SysML". In: *Delaware Valley INCOSE Chapter Meeting, Ramblewood Country Club, Mount Laurel, New Jersey*. 2015 (cit. on p. 9).

[18] incose. *mbse*. Retrieved on July 03, 2020. From `https://www.nasa.gov/consortium/ModelBasedSystems`. 2020 (cit. on p. 10).

[19] Jeff Estefan. "MBSE methodology survey". In: *Insight* 12.4 (2009), pp. 16–18 (cit. on p. 10).

[20] Jean-Luc Voirin. *Model-based System and Architecture Engineering with the Arcadia Method*. Elsevier, 2017 (cit. on pp. 11, 17, 20, 23, 25, 26, 31, 32, 37).

[21] Siemens. *SMW*. Retrieved on July 03, 2020. From `https://www.plm.automation.siemens.com/global/en/products/collaboration/product-architecture.html`. 2020 (cit. on p. 14).

[22] Capella. *Capella*. Retrieved on July 03, 2020. From `https://esd.sutd.edu.sg/40014-capella-tutorial/index.html`. 2020 (cit. on p. 27).

[23] Pascal Roques. *Systems Architecture Modeling with the Arcadia Method: A Practical Guide to Capella*. Elsevier, 2017 (cit. on pp. 31, 32).

[24] Expressen. *Volvo Call Recalls*. Retrieved on July 03, 2020. From `https://www.expressen.se/motor/volvo-aterkallar-750-000-bilar-med-bromsproblem/`. 2020 (cit. on p. 39).

A

Appendix

A.1 [OAB] Operational Architecture Diagram

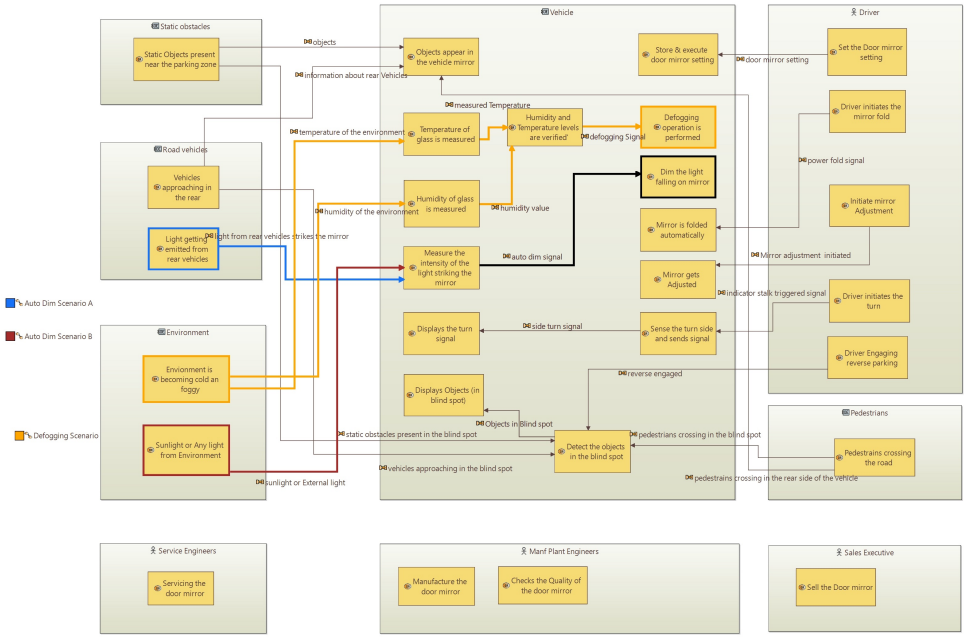

Figure A.1: Operational Architecture for Door Mirror

A.2 [OCB] Operational Capabilities & Operational Actors

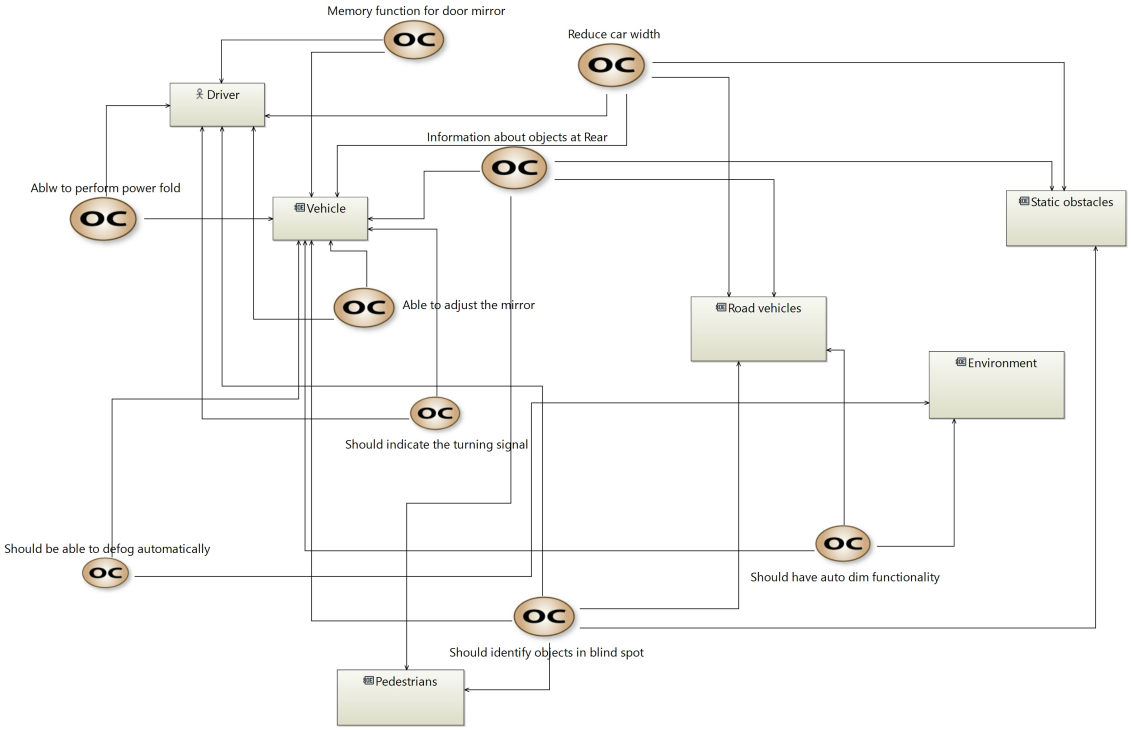

Figure A.2: Operational Capability Diagram

A.3 [OAIB] Operational Activity Interaction - Defogging

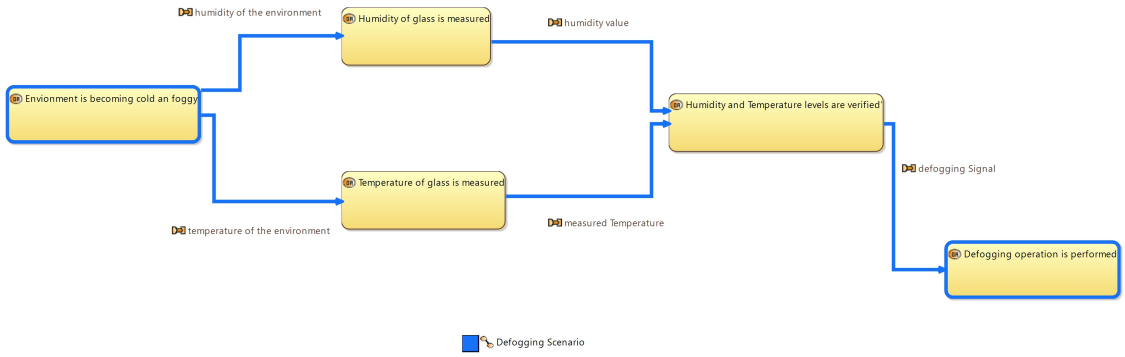

Figure A.3: Operational Activity Interaction [OAIB] - Defogging

A.4 [OAIB] Operational Activity Interaction - Auto Dim

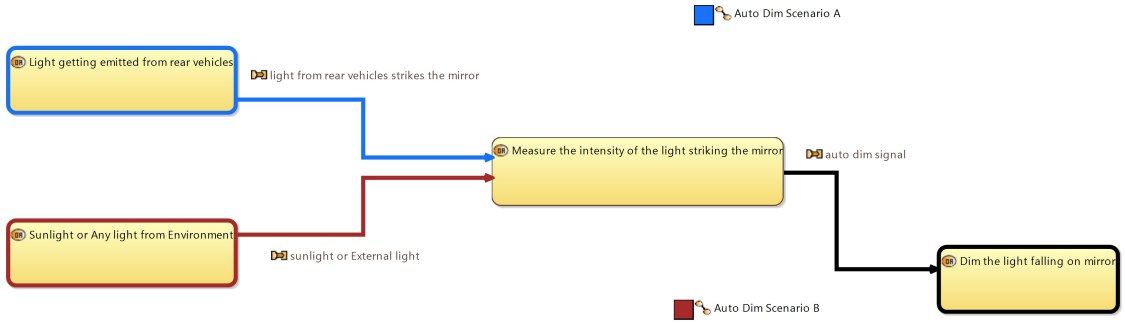

Figure A.4: Operational Activity Interaction [OAIB] - Auto Dim

A.5 [OES] Operational Entity Scenario - Defogging

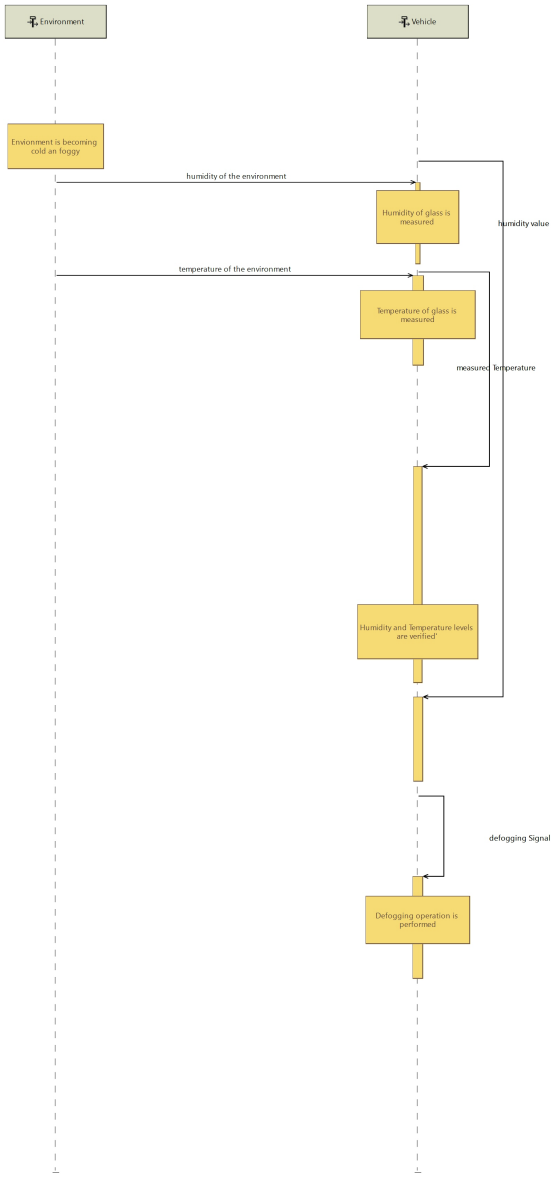

Figure A.5: Operational Entity Scenario - Defogging

A.6 [OES] Operational Entity Scenario - Auto Adjust

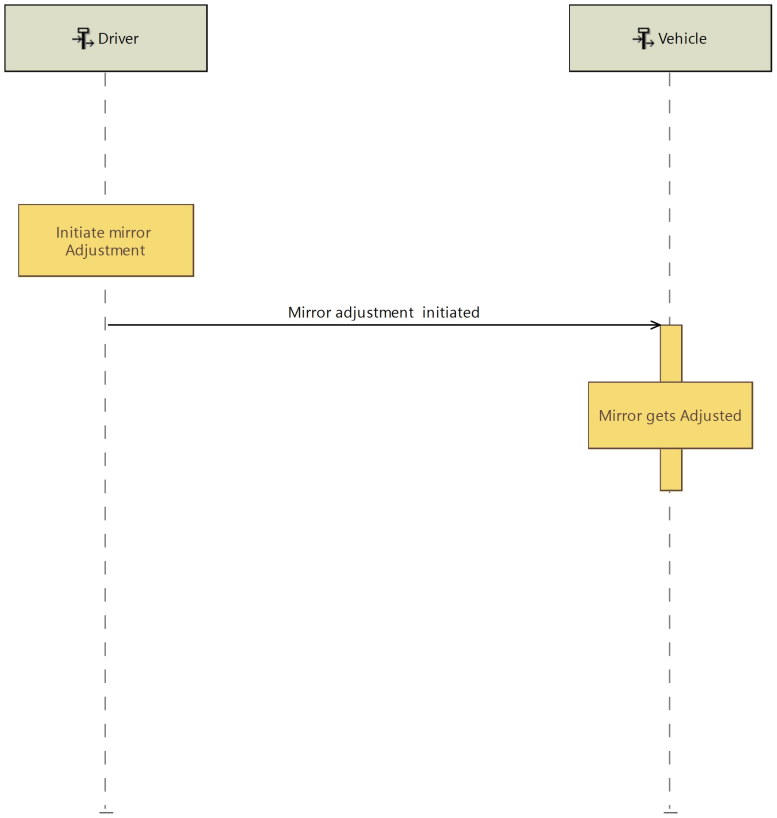

Figure A.6: Operational Entity Scenario - Auto Adjust

A.7 [OES] Operational Entity Scenario - Auto Dim

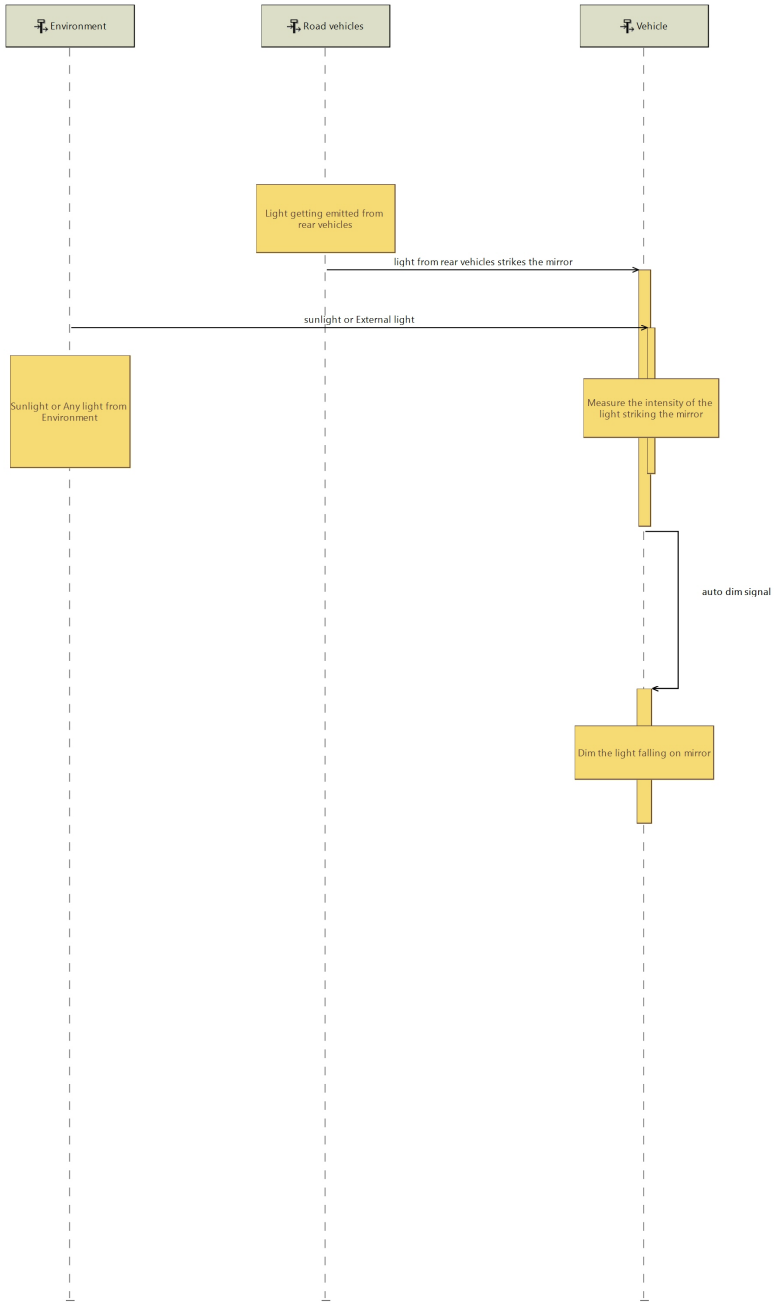

Figure A.7: Operational Entity Scenario - Auto Dim

A.8 [OES] Operational Entity Scenario - Blind spot

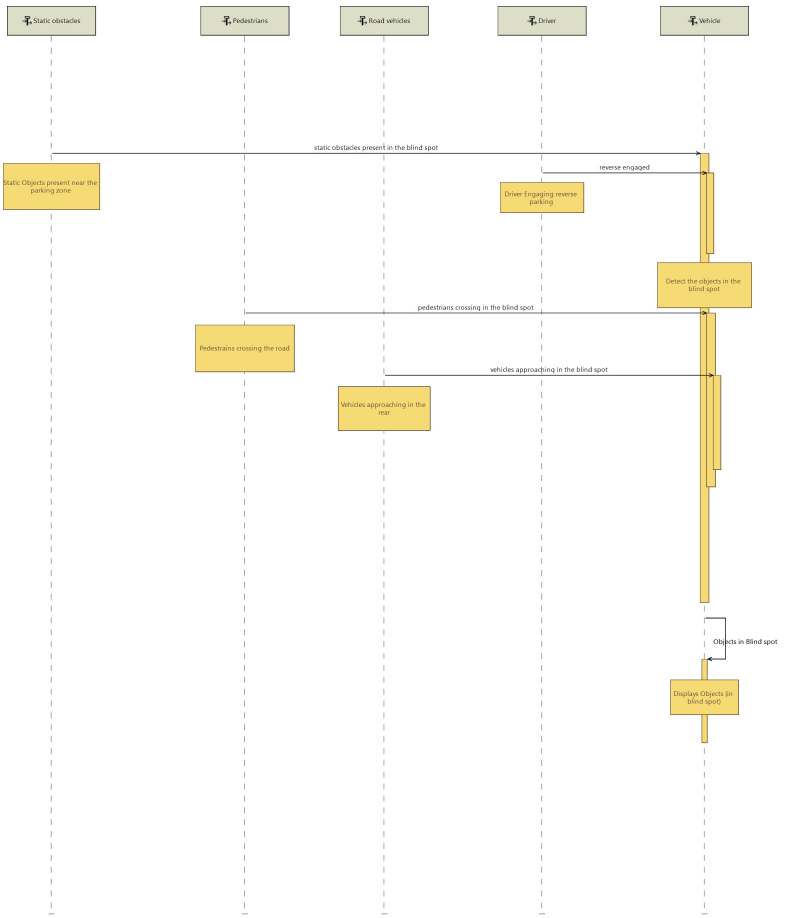

Figure A.8: Operational Entity Scenario - Blind spot

A.9 [OES] Operational Entity Scenario - Memory function

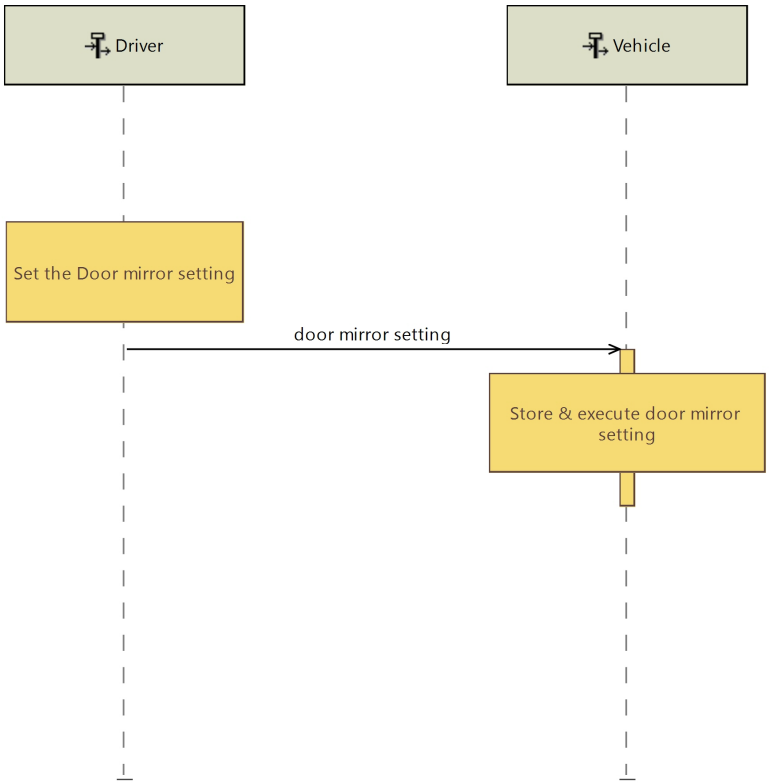

Figure A.9: Operational Entity Scenario - Memory function

A.10 [OES] Operational Entity Scenario - Power fold

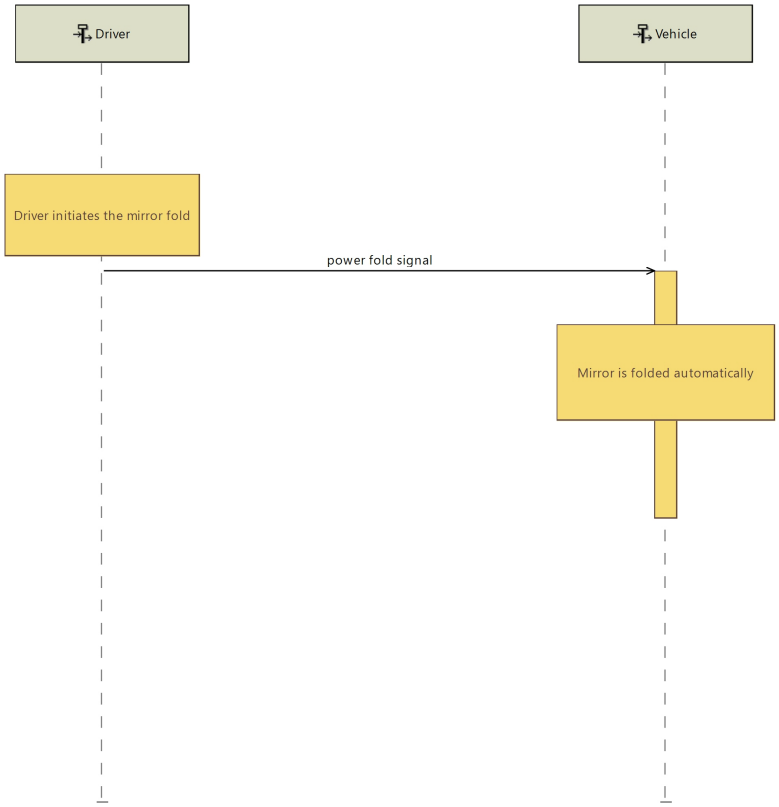

Figure A.10: Operational Entity Scenario - Power fold

A.11 [OES] Operational Entity Scenario - Turning Indicator signal

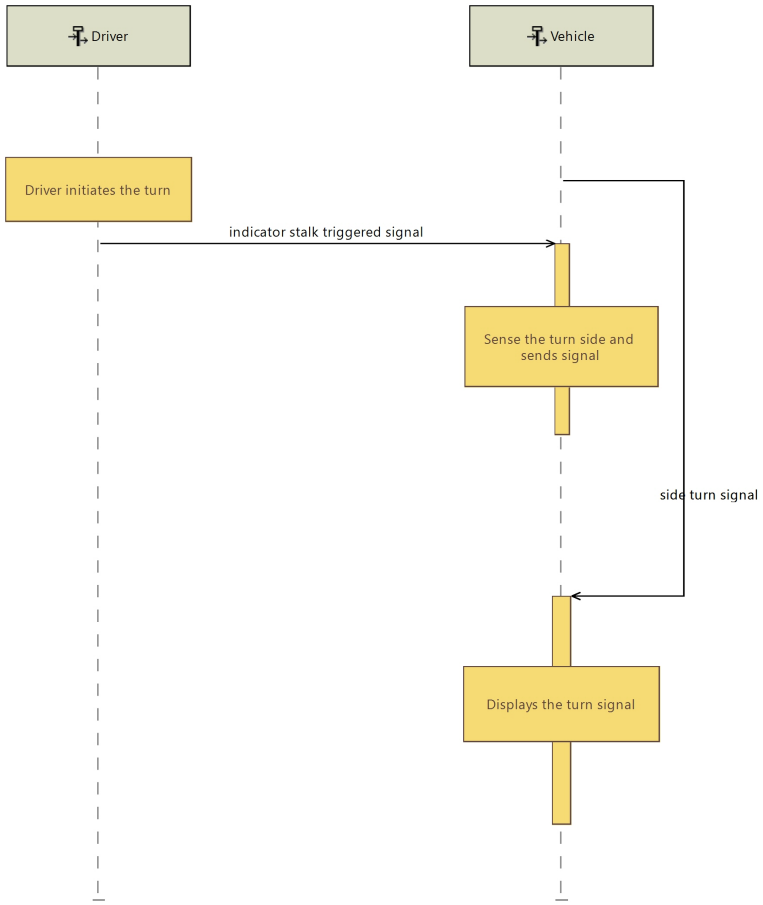

Figure A.11: Operational Entity Scenario - Turning Indicator signal

A.12 [OES] Operational Entity Scenario - Show objects

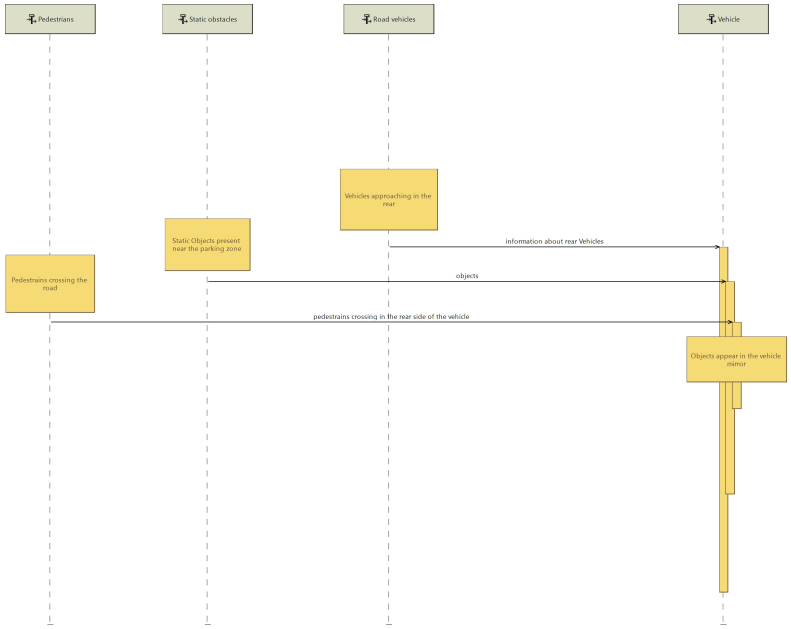

Figure A.12: Operational Entity Scenario - Show objects

A.13 [SAB] System Architecture diagram

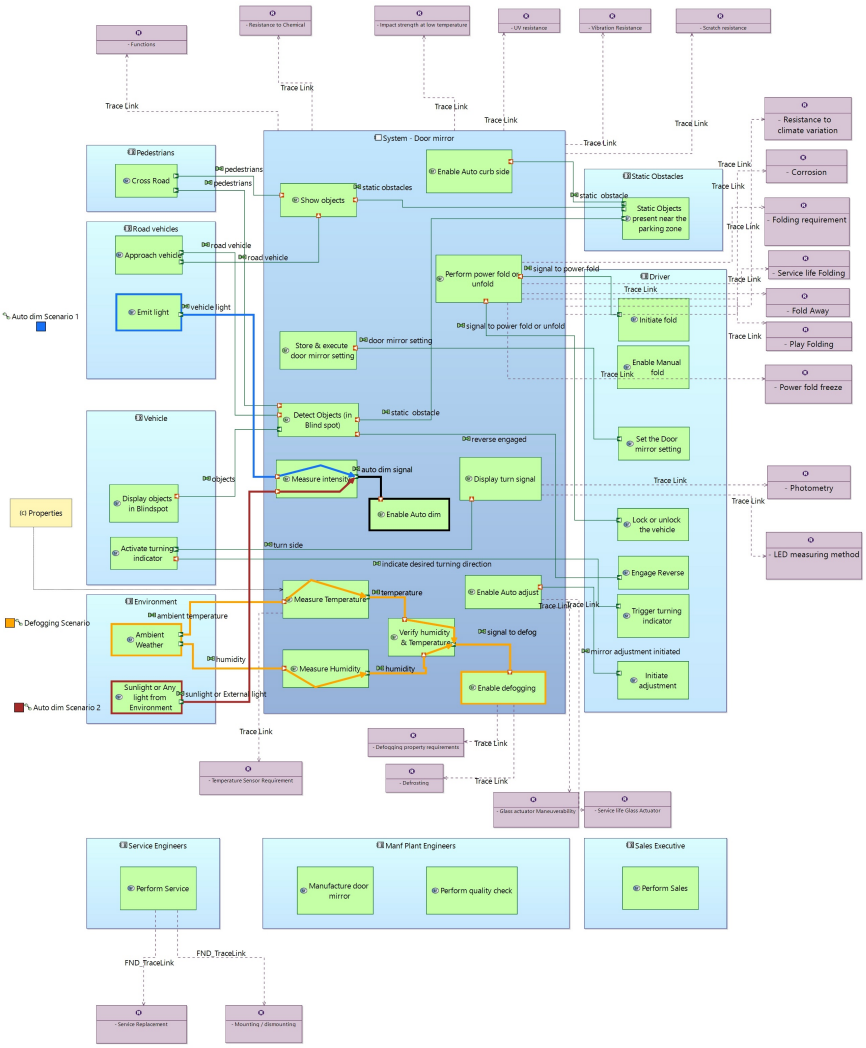

Figure A.13: System Architecture diagram

A.14 [LAB] Logical Architecture diagram

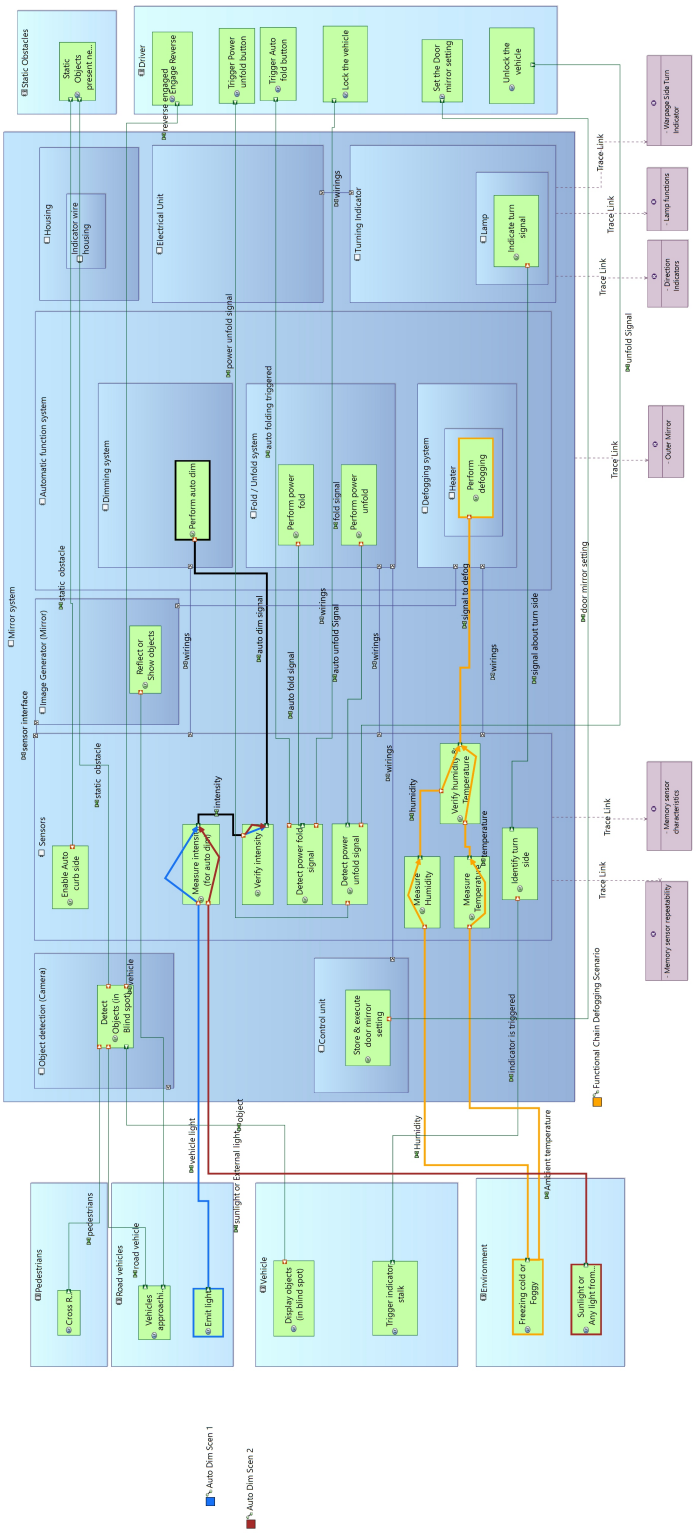

Figure A.14: Logical Architecture diagram

A.15 [PAB] Physical Architecture diagram

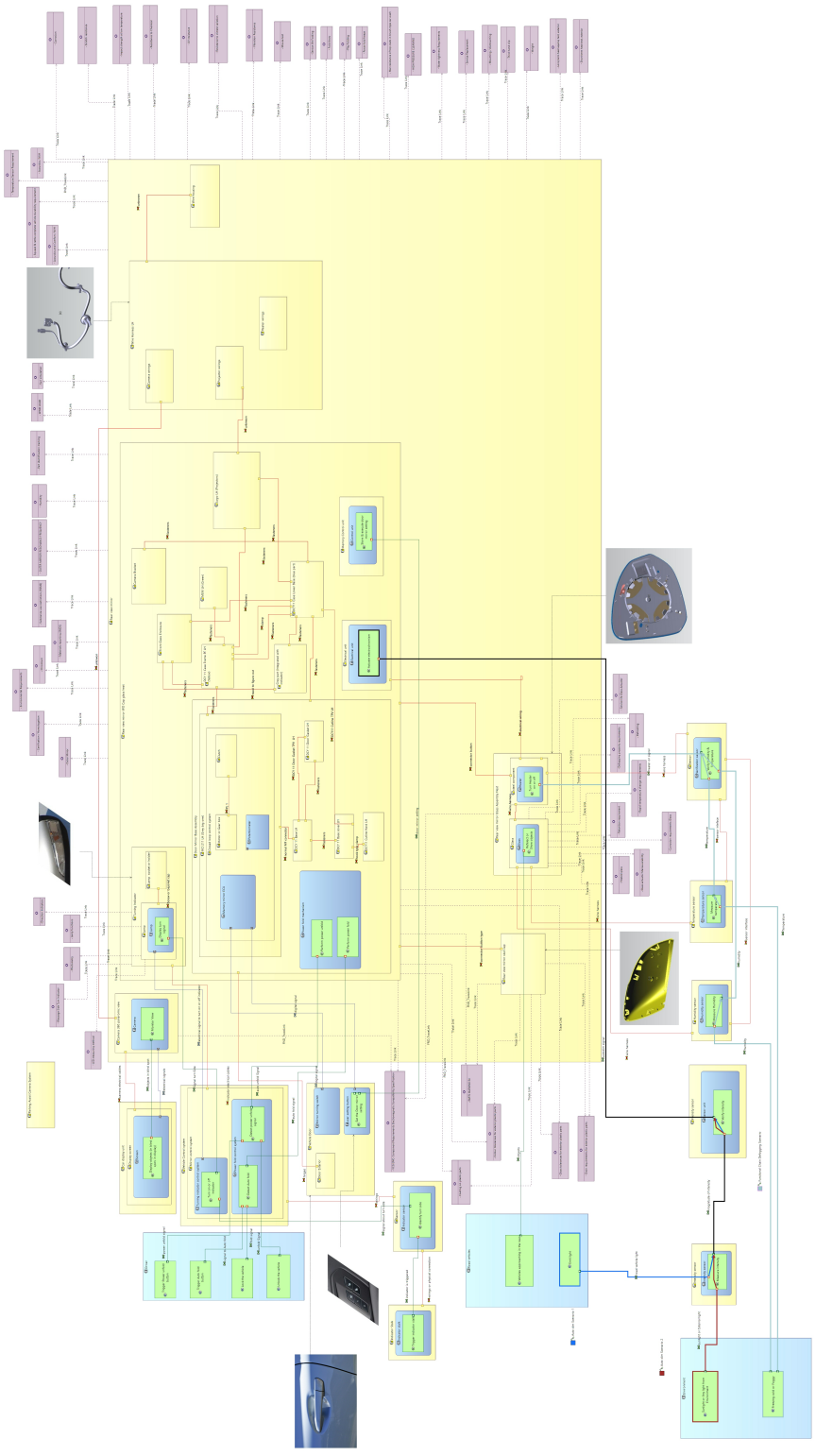

Figure A.15: Physical Architecture diagram