Design Philosophy for User Friendly Parameter Handler

Author: Angie Angarita Soto
ABSTRACT

Date: 15 November 2012
Carried out at: Bombardier Transportation Sweden AB
Advisor at MDH: Radu Dobrin
Advisors at Bombardier: Niclas Evestedt and Erik Simonson
Examinator: Ivica Crnković, and Henry Muccini

DCU2 (Drive Control Unit 2) is an important control system used in applications for train systems that are configured by a set of parameters. Traditionally, parameterization is conducted by using an excel workbook during the software development. The parameters are set up and further export the parameters to the compilation step. Such approach has a number of disadvantages, e.g., delays on the validation and verification steps, system configuration overhead, and suboptimal system reliability generated by the parameter configurations.

To improve the parameterization process, this thesis implements a model-based software architecture approach and automotive industry standards via rapid prototyping by using scrum methodology. We do this by using Matlab/Simulink, TDL (Time Description Language) and UML (Unified Modeling Language) architectural description languages to enable different views of the software architecture. We then develop different prototypes that implement ASAM (Association for Standardization of Automation and Measuring Systems) standards like XCP protocol over Ethernet (code ASAM MCD-1 XCP V1.1.0) and ASAP2 (code ASAM MCD-2 MC) in every scrum sprint. An evaluation then shows that the thesis successfully implements previously defined standards that use commercial tools from e.g., Vector, proving that the parameter’s unit control can be handled via online calibration and measurement, leading to a significant improvement in Bombardier’s software development process in a distributed development environment.

Keywords: Drive Unit Control, Engine Unit Control, ASAM, online calibration and measurement, model based development.
This master thesis was provided by Bombardier Transportation Sweden AB and conducted at Mälardalen University in the school of Innovation, Design and Engineering.

First of all, I would like to express my gratitude to have the opportunity to do my master thesis at Bombardier Transportation Sweden AB since. I strongly believe that the time that I have spent into PPC/ TESEC Control Product department as a thesis worker has significantly enhanced my personal and professional background. I would like also to express my sincere appreciation to my supervisor, Niclas Evestedt and department’s manager Erik Simonson from Bombardier Transportation, for their consistent follow up, valuable feedback, smooth guidance and encouragements to achieve the desired results and project milestones. I would like to extend my gratitude to my colleagues from Bombardier Transportation Max Johansson, Björn H. Andersson, Martin Torelm and Bjarne Jensen, and from Vector Informatik Andreas Patzer, Christoph Heller and Wojciech Przystas, for their valuable technical support during this project development.

I also feel appreciated for the support I got from my university’s supervisor Dr. Radu Dobrin. His work has greatly favoured my thesis development by a good combination of academic research feedbacks and service development inputs for committing the goals that this thesis pursues to obtain. I would also like to give special acknowledgements to my program’s coordinators and thesis examiners Dr. Henry Muccini and Dr. Ivica Crnković for their valuable guidance during my study time in University of L’aquila (Italy) and Mälardalen University (Sweden) respectively, and for providing me the opportunity to join Global Software Engineering European Master program as well.

Finally, I would like thank God and to appreciate my father, my mother, my sister my boyfriend and my friends for their constant support on achieving my professional and personal goals during my Master’s studies.

Angie Angarita Soto
## CONVENTIONS

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>DCU2</td>
<td>Drive Control Unit 2</td>
</tr>
<tr>
<td>ASAM</td>
<td>Association for Standardization of Automation and Measuring Systems</td>
</tr>
<tr>
<td>ECU</td>
<td>Engine Control Unit</td>
</tr>
<tr>
<td>CTO</td>
<td>Command Transfer Object</td>
</tr>
<tr>
<td>DTO</td>
<td>Data Transfer Object</td>
</tr>
<tr>
<td>DAQ</td>
<td>Data Acquisition List</td>
</tr>
<tr>
<td>STIM</td>
<td>Data Stimulation List</td>
</tr>
<tr>
<td>SiL</td>
<td>Software in Loop</td>
</tr>
<tr>
<td>HiL</td>
<td>Hardware in Loop</td>
</tr>
<tr>
<td>SUT</td>
<td>Test Benches</td>
</tr>
<tr>
<td>LET</td>
<td>Logical Execution Time</td>
</tr>
<tr>
<td>TDL</td>
<td>Timing Definition Languages</td>
</tr>
<tr>
<td>XCP</td>
<td>ASAM standard protocol for calibration and measurement</td>
</tr>
<tr>
<td>DTS</td>
<td>Data Set</td>
</tr>
<tr>
<td>IDTS</td>
<td>Integrated Data Set</td>
</tr>
<tr>
<td>AC</td>
<td>Alternate Current</td>
</tr>
<tr>
<td>DC</td>
<td>Discrete Current</td>
</tr>
<tr>
<td>DLL</td>
<td>Dynamic Link Library</td>
</tr>
<tr>
<td>RTOS</td>
<td>Real Time Operating System</td>
</tr>
</tbody>
</table>

Table 1: Acronyms and name conventions
## Kapitel / Chapter 1  INTRODUCTION

1.1 ... Background ............................................................................................................. 7
1.2 ... Purpose .................................................................................................................. 7
1.3 ... Problem Formulation .......................................................................................... 8
1.4 ... Justification ........................................................................................................... 8
1.5 ... Delimitation .......................................................................................................... 9
1.6 ... Thesis Contributions ......................................................................................... 10

## Kapitel / Chapter 2  RELATED WORK

2.1 ... Academic Related Work ..................................................................................... 11
   - Model Based Development of Automotive Control Systems .................................. 11
   - Architectural Description Languages for Real Time Systems .................................. 12
   - Validation and Verification of Automotive Control Systems and model based testing ... 12
2.2 ... Commercial Standards ....................................................................................... 13
   - Basic Definitions ......................................................................................................... 13
   - ASAM Standards ......................................................................................................... 13
   - Calibration Tools common architecture ..................................................................... 14
   - XCP Protocol ............................................................................................................... 14
   - Transport layer over Ethernet ..................................................................................... 16
   - DAQ-List- Data Acquisition Lists ............................................................................... 16
   - STIM-Lists- Data Stimulation Lists ............................................................................. 17
   - Event Channel Module: .............................................................................................. 18
2.3 ... Thesis Approach ................................................................................................. 18

## Kapitel / Chapter 3  MODEL/METHOD

3.1 ... Research Protocol Definition .......................................................................... 19
   - Research Strategy ....................................................................................................... 19
   - Research Keywords .................................................................................................... 20
3.2 ... Software Development Methodology ................................................................. 20
   - Agile ............................................................................................................................. 21
   - Scrum .......................................................................................................................... 21
   - Scrum roles ................................................................................................................. 21
   - Scrum Steps ................................................................................................................. 22
   - Related concepts: ....................................................................................................... 22
3.3 ... Assumptions and Limitations Concerning to the Thesis Development .......... 23
   - Environmental ........................................................................................................... 23
   - Technologies .............................................................................................................. 23
   - Work Distribution ..................................................................................................... 24
   - Time and location ....................................................................................................... 24
   - Prototyping ................................................................................................................ 24
   - Initial Costs ............................................................................................................... 25
   - Quality Assurance Plan ............................................................................................. 26
   - Risks ........................................................................................................................... 26

## Kapitel / Chapter 4  SOLUTION

4.1 ... Commercial Tools .............................................................................................. 28
   - Vector: ....................................................................................................................... 28
   - CANape 10.0: ............................................................................................................ 28
<table>
<thead>
<tr>
<th>Commercial Tool’s Solution Selection</th>
<th>31</th>
</tr>
</thead>
<tbody>
<tr>
<td>Motivation for Tool Selection</td>
<td>31</td>
</tr>
<tr>
<td>Evaluation of the Commercial Tools for Parameterization</td>
<td>32</td>
</tr>
<tr>
<td>CANape, eASEE from Vector</td>
<td>32</td>
</tr>
<tr>
<td>INCA from ETAS</td>
<td>33</td>
</tr>
<tr>
<td>Control Desk Next Generation (CDNG) from dSPACE</td>
<td>33</td>
</tr>
</tbody>
</table>

| General Design Description of Developing Prototypes | 34 |

| Use Cases organization per prototype | 37 |

| State Machine Description | 40 |

<table>
<thead>
<tr>
<th>Design Decisions</th>
<th>41</th>
</tr>
</thead>
<tbody>
<tr>
<td>Concern: Which method shall we use for checking memory and resource availability?</td>
<td>42</td>
</tr>
<tr>
<td>Criteria: Functions developed in operative system, complexity, time effort</td>
<td>42</td>
</tr>
<tr>
<td>Concern: Which persistent memory must we use for storing parameters when DCU2 is off or disconnected?</td>
<td>42</td>
</tr>
<tr>
<td>Criteria: security, maximum memory storage capacity, back up resources, memory space availability, and memory overload risk</td>
<td>42</td>
</tr>
<tr>
<td>Discussion about requirements fulfilment during XCP-Train Device Integration</td>
<td>43</td>
</tr>
</tbody>
</table>

## Kapitel / Chapter 5 RESULTS/EVALUATION

5.1 Results discussion of different prototypes | 46
5.2 Recommendations | 58

## Kapitel / Chapter 6 CONCLUSIONS AND FUTURE WORK

## Kapitel / Chapter 7 REFERENCES

APPENDIX A - RISK MATRIX | 63
APPENDIX B - SOFTWARE ARCHITECTURE DOCUMENT | 70
1.1 Background

DCU2 is an important control system used in several applications for train systems, and it is configured according to the system requirements and characteristics during software development and maintenance processes. The set of train applications is configured before compiling and downloading software to the DCU2 controller. A commonly used process in this step is saving this configuration into an excel workbook which allows to set up and export the parameters in the compilation step. However, this mechanism represents a drawback for the software integration process since it cause delays, overhead in configuring controllers due to the lack of documentation, lack of expertise, reliability at the data generated for enabling a good system performance, etc.

In addition, one of the inconveniences that causes delays is that the excel file takes a large amount of time to be opened since its size is extremely large. Another disadvantage is that there are sets of critical parameters’ value that fulfil safety critical properties that have certain dependencies among them that are not validated. Hence, this introduces several dependency conflicts during DCU2 configuration process. Consequently, this can cause unexpected results during the testing phase because a set of parameters is not valid, or are set up with an inaccurate value. Furthermore, the excel workbook is not user friendly, and it does not have a help site either. As the consequence, this adds unnecessary complexity during software development.

1.2 Purpose

The purpose of this thesis is to solve the previous explained situation by designing a new philosophy to manage train controller parameters based on a model-based architecture that implements commercial tools via fast prototyping. Thus, we will perform a scientific review about commercial tools, trends, and techniques that might offer a set of solutions and functionalities to solve the previously mentioned situation. This review also analyses to determine tool’s advantages, disadvantages, and business feasibility. This provide necessary inputs to us for purposing benchmarking solutions, project scope, contacting suppliers for commercial tools, and resources. In addition, we will describe a formal architecture and design a new parameter handling philosophy to be integrated with commercial tools into the DCU2. The thesis also develops some prototypes to check whether proposed design and the tool fulfil the system requirements. Proposed prototypes will contain solutions in high level and low level to handle parameters in DCU2. The thesis objectives are:

- Perform a systematic research about calibration tools, handling parameter techniques, and automation calibration tools.
• Analyze results to propose best solutions based on benchmarking.
• Discuss research results.
• Design software architecture and parameter classification to adapt commercial tools into current situation.
• Develop prototypes to perform validation and verification from proposed architecture.

1.3 Problem Formulation

The main problem that Bombardier has to face every time that DCU2 parameterization is performed is having project delays, lack of parameter traceability and configuration management. Software developers and application engineers have to overwork the process because they have to run the whole configuration process every time that a single parameter is changed. This is due to the lack of parameter traceability that the current tool lacks. Therefore, we will study the following research question;

“How to implement a new philosophy for handling parameters that allows engineers to handle distributed development by decreasing redundancy and providing team collaboration, controlled versioning, configuration management and a life cycle management at Bombardier Transportation?”.

In order to answer this research question, we will perform a scientific review based on scientific and industrial papers, articles, and related automotive industry’s standards. This will provide us the necessary background in order to state a feasible solution. Afterwards, we will design and model a software architecture that implements the suggested solution and validate it via fast prototyping. Hence, we have divided the main research question as follows:

• Which commercial tools exist for handling parameters in the automotive industry? How good are those for our purposes?
• Which are new trends and standards for optimizing controllers’ parameter configuration?
• Which are technical and functional details about parameter calibration commercial tools?
• Which are the software design patterns and model based techniques that can help us to design the new parameter handling philosophy?
• How should we develop the different prototypes for validating and verifying the purposed philosophy?

1.4 Justification

Nowadays, Bombardier Transportation is looking for a set of tools, methodologies, process and others useful ways to increase productivity in their product development. Thus, PPC/TESEC Control Product department has identified a critical process that has been generating delays at every DCU2 product development project. In particular, a current excel workbook that handles parameters causes much overwork at any project development processes such as software testing, and software validation and verification. This is because the developers have to spend much time and effort to execute manual tests for every single parameter several times

Hence, we shall design a new independent, scalable, reliable philosophy in order to end up with a feasible solution to cope with the described inconvenient. As we will implement a new tool, delays in project development, lack of parameters’ reliability, overwork during
software development and configuration will be significantly reduced. Consequently, this philosophy will improve project metrics, productivity during product development. Additionally, the new philosophy will enhance main Bombardier’s goal about competing in the market with new tendencies in software development.

In addition, this thesis provides an internal value that is related to compete with other companies in providing safe and reliable products to generate great revenue for the organization. Regarding to productivity increment, we will provide a solution for handling parameters that will save time and overhead in projects at PPC/TESEC Control Products department, by enhancing team collaboration and distributed development. Thus, we will provide a more efficient process workflow to get results on time. In addition, the thesis presents an external value that will be beneficial to other departments at Bombardier to deliver more reliable configuration references for further projects in a short amount of time. This will bring beneficial results at company’s metrics by reducing costs and time effort.

1.5 Delimitation

This thesis will require some services and materials in order to develop a new and complete parameter handling philosophy. Mälardalen University (MDH) and Bombardier Transportation will provide most of services in a close collaboration among them. MDH will offer a thesis worker who will perform the research an academic supervisor. Bombardier Transportation will provide all physical and economical resources to design and develop the philosophy. In particular, PPC/TESEC control products department will provide industrial supervision to collaborate with the technical knowledge and the voice of the customer.

We will require some materials during the thesis development that refers to hardware and software pre-requisites and procurement process. In this case, PPC/TESEC department will request to IT division some materials like application server, database server, personal computers, developing and testing environments, modelling tools, development IDE and IT support. Furthermore, Bombardier’s administration department will handle all procurement process required to get these materials. Furthermore, we will not consider a set of software development activities to be executed due to the thesis time limitations. Thus, a list that explains every excluded activity it is stated below:

- **Testing**: we do not include testing activities since they are time consuming, and they might represent an obstacle for developing the prototypes. Thus, validation and verification steps will be performed in order to check system functionality and the architecture. Therefore, Bombardier’s staff may perform testing activities in case they decide to implement the solution.
- **Implementation**: due to the nature of a safety critical system, we consider that it is difficult to run full implementation of commercial products in Bombardier in a short time without considering testing phases and the IT infrastructure support.  
- **Pilot programs**: they are considered as a future work only if Bombardier decides to implement the solution.  
- **Work overflow design**: we will consider this step an assumption provided by a supplier because it might introduce several changes into Bombardier’s organization that represent an impact into any project lifecycle.  
- **Parameters classification**: This suggests including a deep study into all possible options for classifying parameters that is time consuming. Therefore, this point may be subject to a further work. Instead, we will provide constrains and rules to adapt existing parameters into different commercial tools by showing some project examples.
1.6 Thesis Contributions

In short, the following list outlines the major contributions of this thesis:

- Scientific review and literature survey concerning tools for parameterization in the automotive industry
- Analysis and evaluation of the commercial tools for parameterization process at Bombardier
- Justification of the tool selection, methodology and standards for the new parameter handler philosophy
- Implementation of standards and commercial tools by developing prototypes
- Evaluation of the thesis results
Kapitel / Chapter 2

RELATED WORK

2.1 Academic Related Work

Model Based Development of Automotive Control Systems

Nowadays, the usage of distributed architectures in real time automotive control systems significantly reduces costs and maintenance activities [1]. This approach can be obtained through automating different software development processes according to different engineering domains. In particular, flexible and comprehensible modular systems can be designed and developed via model-based architecture in order to achieve flexible, reusable, function oriented and scalable software components. Thus, some paradigms that have been currently used among different automotive industries include system orientation, functional orientation, and systems engineering. System orientation refers to develop a system integration based on components architectures. In contrast, functional orientation describes a systematic requirement engineering and architecture/function development based on feature modeling. Likewise, system engineering makes emphasis on requirement management, architecture, and integration phases. Furthermore, there is the need of describing different architectural viewpoints that shall be composed of conceptual and technical views. Conceptual view states hierarchical functional views about local subsystems interaction whereas technical view describes code architecture and runtime behaviour.

It has been claimed that using model driven design methods may enhance different modeling tasks [2]. One idea is to develop macroscopic and microscopic models. The concept about macroscopic model suggests to generate some graphical or textual models that can represent control software behaviour, algorithms and parameter descriptions to run a simulation based on these models. Microscopic models describes a generated code via model to model transformation that represent a description about low level components and their different connections that allows parameters, signals and values passing as shown in Figure 1.
Architectural Description Languages for Real Time Systems

Modern approaches to software design like model driven engineering (MDE) provides different solutions to provide timing properties preservation in real time systems from high level models to low level models [1, 25]. Some synchronous languages such Logical Execution Time (LET) and Timing Definition Languages (TDL) describe different real time system timing properties. In particular, there is a commercial tool suitable to design graphical and textual models like VisualCreatorTDL tool for TDL. Then, this tool provides integration with Matlab/Simulink where other models can be generated via model-to-model transformation tool options.

Validation and Verification of Automotive Control Systems and model based testing

Some approaches have been using recently to run validation verification at automotive control systems [1]. One is software in the loop (SiL) and another is hardware in loop (HiL). HiL enhances simulations by configuring a hardware environment to check whether a functional or timing property is satisfied, whereas, SiL is mainly used for checking software functional properties. In addition, there are other important testing types that are used as model based testing and mutation testing. Model based testing requires a test infrastructure that involves under test (SUT). This requires a HW/SW support from test bench that may include HiL or SiL. In contrast, mutation testing quantifies a test suit by measuring how many faults can a mutant version from original software may fail that represent common programmer’s typing mistakes at the code. Consequently, all of previous mentioned approaches are designed and executed by Matlab/Simulink, dSpace tools, or other resources.
2.2 Commercial Standards

In this section, we will describe some basic definitions used in commercial tools and standards in the automotive industry. We will give the detailed information as follows:

Basic Definitions

This section will outline some basic terms that current commercial tools uses to apply different approaches based on section 2.1 in automotive control system software development [3]. The terms belong to “Measuring and Calibrating” process at controllers’ development at V-Model in the right side (validation and verification steps) as it is shown in Figure 2.

![Development of an ECU (V-Model)](image)

- **Calibration**: it is a term that describes different requirements to perform measuring, adjusting parameter/signal values, flash programming, and other related tasks in a rapid prototyping environment.
- **Measuring**: this is a technique that it is used to visualize control algorithm’s behaviour, record and display I/O parameters, environmental data, internal variables, etc. This also shows different time-correlated possible variables that can be considered in offline or online evaluation.
- **Calibrating**: it is a process that optimize closed loop controlled system’s behaviour to determine whether a parameter or records affect control system process. This task is usually performed by the application engineers according to project requirement and specifications.
- **Flash Programming**: this kind of programstyle calibrates ECU or other controller’s parameters by reprogramming it flash memory.

ASAM Standards

ASAM is for short Association for Standardization of Automation and Measuring Systems that provides different solutions to provide efficient and professional project management [4]. The association develops different standards and protocols to assist automotive industries to develop their controller products. Some interfaces that ASAM includes are described below:

- **ASAM MCD-1**: [3, 4] this defines connection of an ECU to a computer or
measurement device. This connection is handled via transportation layer. This also includes interfaces for managing internal run-time variables and calibration parameters. In particular, XCP protocol is defined in this layer.

- **ASAM MCD-2**: [3, 4] this states a set of standards for ECU access and network data transmission that are enabled through some file exchange conventions like ASAP2 or ASAP3. Besides, file descriptions contains memory address, data type, data format and some conversion functions to transform internal parameter to physical parameters.

- **ASAM MCD-3**: [3, 4] this standard states and object-oriented API between measurement, calibration and diagnostics functions. Therefore, this allows having test automation in client server systems between controllers, PC or other supplier. It also suggests different solutions to run measurement and calibration process in a high-level automation system in order to acquire measurement data and perform parameter calibration.

- **A2L file**: [4] it is a description format for parameters and measurements of an Electronic Control Unit that also describes communication and interface descriptions. This kind of file also contains the required information to transform internal data into physical one and vice versa. A common market name is ASAP2 or ASAP3 (that refers a newer version for standard ASAM MCD-3).

**Calibration Tools common architecture**

A common ASAM architecture [5, 16] defines communication between measurement and calibration systems and ECU. It also provides a set of interfaces that allows communications between automation interfaces for test benches, protocol and transport layers. This set of interfaces communicates directly to A2L description file of the memory content. This can be shown in Figure 3.

![Figure 3: ASAM Interface model](image)

**XCP Protocol**

XCP protocol [7, 17, and 16] refers to “Universal Measurement and Calibration Protocol”. It serves to enable different interfaces in a measurement and calibration system. It also allows communication over other standard protocols like CAN, Ethernet, USB, etc. This protocol is defined as two-layer protocol since it separates protocol layer from transport
layer, and it takes Single-Master/Multi-Slave communication style. Thus, a single master system describes the connection between PC to multiple slaves that run in embedded systems via request and response messages passing as it is described in Figures 4 and 5. This allows getting a complete view of any automotive control system where this protocol is executed. This mechanism provides the flashing programming feature since it allows modification of persistent memory to be replaced by a firmware or calibration parameters. This means that the master can perform a complete replacement of the code that runs in ECU via firmware once this has been uploaded. This feature prevents boot loader component at ECU, so XCP master may have a complete control over ECU during calibration tasks. Moreover, XCP can be protected with a Seed and Key architecture that prevents control units from tampering and password sniffing over transport layer.
In particular, message passing between master and slave contains distinctions to handle different commands [6, 17]. Thus, an important distinction is made between Command Transfer object (CTO) and Data Transfer Object (DTO) to handle a synchronous data acquisition from slave’s memory. CTO contains the following commands: CMD (command), RES (Response), ERR (Error), EV (Event), and SERV (Service Request Processor). In order to get required data from slave, DAQ (Data Acquisition) and STIM (Stimulation) commands are executed to transfer objects from event-driven reading of measurement variables. This structure is described in Figure 6.

**Transport layer over Ethernet**

XCP over Ethernet [7, 17] message frame is done through packets. This is composed of a header and a tail. XCP header contains packet number and length values whereas XCP packet is composed of data information such as DAQ, TIMESTAMP, etc as it is explained in Figure 7. This allows to TCP/IP protocol to impose a limitation of packet size where DTO/CTO is described to enhance network performance.

**DAQ-List- Data Acquisition Lists**

A DAQ list [7, 17] allows sending a large amount of data in a short period with low bandwidth. Each data list contains a number of Object Descriptor Tables (ODTs) and Object Descriptor Table Entries (ODT Entries). ODT contains address and length from parameter
description. When DAQ-list is processed, each list data is copied into the corresponding address from each ODT as it is stated in Figures 8 and 9. The list may contain static and dynamic configuration. Static configuration may be used when parameter descriptions shall not be modified. In contrast, Dynamic configuration can be used during calibration task since the master can have a direct access to change parameter descriptions.

Figure 8: DAQ- ODT list number identification (PID)

Figure 9: DAQ-list memory allocation

STIM-Lists- Data Stimulation Lists
STIM-lists [7, 17] have similar DAQ’s-list features, but they allow master to stimulate data in a controlled environment. This is because master can write its last buffered data in the slave until STIM list is executed. This permits to get a data copy into a specific ECU memory address. Consequently, this avoids the need of implementing control loops mechanism by decreasing redundancy.
**Event Channel Module:**

In order to handle events, XCP provides an event channel module [8, 17]. This permits the DAQ list to be simultaneously active to trigger events at the slave component. When an event has to be triggered, this channel builds a generic signal source that allows to determine whether a data may be transferred simultaneously or not in a given period. This channel also describes frequency of event execution.

### 2.3 Thesis Approach

We will implement a theoretical framework based on different software architecture approaches and automotive industry standards in order to arrive into a feasible solution. In this sense, we will use some approaches from model-based development of control systems for designing a solution based on different architectural representations. Then, some basic concepts from model driven engineering will be used to generate code in order to run validation and verification process via Matlab/Simulink and TDL languages. This is for generating a close simulation to real-time behaviour. Therefore, we can use a model checking approach for checking whether state machines that handle previous behaviour may meet safety critical properties. In addition, we will implement ASAM standards for developing a set of prototypes since they have been implemented successfully in other automotive industries to solve related issues and challenges.
Kapitel / Chapter 3
MODEL/METHOD

3.1 Research Protocol Definition

In this section, we will define the strategy to perform a scientific and industrial review for finding empirical evidence to the research questions outlined below. This strategy describes the data extraction techniques to assure that the extracted information will be relevant to the research questions.

Research Strategy

We will use an industrial oriented research strategy. We will also use different kinds of resources, libraries, and keywords from stakeholder references. The research libraries that we will use are Science Direct, SpringerLink, ASAM download center, IEEE libraries, Google scholar or web, automotive magazines, supplier’s product download center, etc. Moreover, we will use sources and references from stakeholder’s suggestions, and supplier’s contact information. Other useful resources are located at Bombardier’s knowledge database that can provide enough documentation about DCU2, train systems and other related devices. Furthermore, we are expected to run two research iterations in order to gather preliminary results and refine keywords to filter results respectfully. Additionally, we will run a set of interviews with the related stakeholders in PPC/TESE as a complement of the collected information to understand the current situation and system requirements. In this sense, some questions that we will answer in this research are:

• Which are commercial tools for handling parameters in automotive industry? Are there any commercial tools that can offer solutions to calibrate parameters in automotive industry? How good are those for our purposes?
• Which are current study cases that claim benefits of using these commercial tools?
• Which are new trends about optimizing controllers’ parameter configuration?
• Which are technical and functional details about parameter calibration commercial tools?

In addition, we have considered different techniques as a data extraction strategy from the scientific and industrial review to optimize research time, analysis, and the thesis discussion. This will allow us to filter keywords and refining the review results. Some of them are:

• Read abstracts and other general overview at each found article to determine whether it is useful or not.
• Highlight keywords and main ideas from articles.
• Make a contact list about stakeholders, suppliers, and other key people interested in project.
• Include solutions and describe them at the benchmarking document.
• Save the minutes of meetings (MoM) into a document to organize and analyse stakeholder references and suggestions.
• Review at Bombardier’s document references to reach more information about their products.
• Extract resources according to publication date and location. This will help us to determine whether the article presents a proved theory or not. Thus, if the article presents a non-proved theory, we will only extract a key idea or basic concept. Otherwise, we will consider further information such as solution details, use cases and other useful information.

Research Keywords
• ECU Calibration Tool
• ECU Calibration optimizer
• ECU Calibration parameter
• DCU2
• Train propulsion systems
• ASAP2 file
• ASAP3 file
• ASAM MDC-3 V2.2.0
• ASAM MDC-2 MC
• XCP Protocol
• Automation calibration
• MCU Signals Parameters
• Calibration tool architecture
• Calibration tool requirements/pre requisites
• Automotive software parameter configuration
• Automotive software calibration Tool
• CANape
• INCA
• dSpace
• CalDesk
• ControlDesk Next Generation (CDNG)
• ASAM AE Standards
• VxWorks
• PowerPC hardware Architecture

3.2 Software Development Methodology

In order to develop a solution that solves the research question, we have selected a set of approaches from the section 2.1 and the agile methodology with the Scrum method. One of the selected approaches refers to model based development in the automotive industry that we will use for designing the software architecture. Then, we will validate and verify the architecture by applying model checking techniques, model simulation, and fast prototyping based on SiL and HiL as we have described them into the section 2.1. We will also use the described techniques for executing and verifying the DCU2 parameterization process. Therefore, we will apply an iterative and incremental methodology such Agile based on Scrum methods to allow us to develop the different prototypes that will implement the previous selected approaches. Thus, we have stated brief description about Agile and Scrum
Agile

Agile software development [14] is a group of software development methods based on iterative and incremental development, where requirements and solutions evolve through collaboration between self-organizing, cross-functional teams. It promotes adaptive planning, evolutionary development, and delivery, a time-boxed iterative approach, and encourages rapid and flexible response to change. A conceptual framework promotes foreseen interactions throughout the development cycle. Furthermore, it is very effective where Client frequently changes his requirement because it involves more client interaction and testing effort. This ensures bugs are caught and eliminated in the development cycle, and the product is double tested again after the first bug elimination. Another Agile method’s advantage is that it allows for specification changes as per end-users requirements, spelling customer satisfaction. Hence, there are two methods by which this methodology can be implemented as scrum and extreme are programming. Consequently, we have selected scrum methodology for developing this thesis and we describe it as following:

Scrum

Scrum [15] is an iterative and incremental agile software development method for managing software projects and product or application development. This is composed of roles, project lifecycle, release, and sprints. Some of these concepts are described below:

Scrum roles

Scrum [15] is a process skeleton that contains sets of practices and predefined roles. The main roles in Scrum are Product owner, Scrum Master, and Team Member that are described below:

- **Product Owner:** The Product Owner represents the voice of the customer and is accountable for ensuring that the Team delivers value to the business.
- **Team Member:** a Team member is any person that belongs to the development team that is responsible for delivering any product or artefact.
- **Scrum Master:** Scrum is facilitated by a Scrum Master, also written as Scrum Master, who is accountable for removing impediments to the ability of the team to deliver the sprint goal/deliverables. The Scrum Master ensures that the Scrum process is used as intended. The Scrum Master is the enforcer of rules. A key part of the Scrum Master’s role is to protect the team and keep them focused on the tasks.
Scrum Steps

An important feature from the iterative process is that it contains sequential steps and a lifecycle that ensures goal commitment [15]. This is shown in figure 10, and some of these steps are:

- A product owner creates a prioritized wish list called a product backlog.
- During sprint planning, the team pulls a small chunk from the top of that wish list, a sprint backlog, and decides how to implement those pieces.
- The team has a certain amount of time, a sprint, to complete its work - usually two to four weeks – but they meet each day to assess its progress (daily scrum), which we will do according to our needs and time.
- Along the way, the Scrum Master keeps the team focused on its goal.
- At the end of the sprint, work will be presented on our milestones as a final product for that phase.
- The sprint ends with a sprint review and retrospective on which all project members will discuss what has been done, and what has not.
- As the next sprint begins, the team chooses another chunk of the product backlog and begins working again.
- The cycle repeats until enough items in the product backlog have been completed, or a deadline arrives. Which of these milestones marks the end of the work is entirely specific to the project, but in our case, goal is to provide solutions to every requirement. No matter which impetus stops work, Scrum ensures that the most valuable work has been completed when the project ends.

Related concepts:

- Product backlog: [15] may be dynamic where Items may be deleted or added at any time during the project. It can also get prioritized items with the highest priority are completed first. Therefore, it will be progressively refined where lower priority items are intentionally coarse-grained.
- Sprint backlog: [15] a sprint backlog is a negotiated set of items from the product backlog that a team commits to complete during the time box of a sprint. Items in the sprint backlog are broken into detailed tasks for the team members to complete. The team works collaboratively to complete the items in the sprint backlog, meeting each day (during a daily scrum) to share struggles and progress and update the sprint backlog and burn down chart accordingly.
• *Potentially Shippable:* [15] after every sprint product is considered potentially shippable. In that phase, one part of the project is considered done no matter which requirements are met since we will not have time to consider reviewing them. There is a possibility that couple of requirements will be used in two different sprints. This is not very popular among Scrum process at all, but we would like to ensure ourselves to have a back door if something goes wrong in one sprint. The product owner makes the decision about when there will be a release of any functionality or deliverable.

### 3.3 Assumptions and Limitations Concerning to the Thesis Development

#### Environmental

We will develop this project in three different servers that will stay physically between 2 computers. This schema allows developing fast prototypes regardless procurement time limitations. Thus, a computer may contain two servers: one for developing database and other for configuring application database and configuration management. The other computer may serve as client for application server environment and as virtual server for handling Scrum management tool (Ice Scrum). Besides, we will get a DCU2 sample board to test low-level solutions. Once that prototypes have been finished and validation and verification steps have been completed, project sponsor may decide about running the implementation to request all needed hardware.

#### Technologies

We will organize this project according to different solutions that implement a combination of system styles. One solution suggests implementing a client server application to manage parameter calibration. Other solution will be based on real time systems event handling that is composed of a layered architecture that describes different protocols and standards in automotive control systems. In this sense, prototypes from both scenarios will be designed, developed and tested by following some model-based architecture, component based architecture and rapid prototyping concepts. Therefore, validation and verification steps will be performed in conjunction with some stakeholders and application engineers. Consequently, some technologies that may be accurate to use to cover different approaches are listed below:

- **Developing Languages:**
  - Java Standard Edition
  - C/C++
  - Matlab/Simulink

- **Architectural Description languages:**
  - UML
  - Matlab/Simulink
  - TDL

- **Development IDE:**
  - Matlab/Simulink
  - Mitrac Tools

- **Modeling tools:**
  - Magic Draw
- Matlab/Simulink
- TDL Visual Creator

- Solutions from market:
  - Control Desk Next Generation (CDNG)
  - CANape 10.0
  - INCA tools
  - ASAM standards
  - eASEE Tool
  - IceScrum

- Servers:
  - Apache Tomcat for an application server
  - Oracle

- Database:
  - Oracle

Work Distribution

Since we will implement in this project different software and hardware components that will follow rapid prototyping concepts, it is important to take as a sample a previous project, select a device and team responsibility to perform all different tasks during project development. This means that we will use Scrum method to distribute some punctual task among team members and execute them in parallel or in a sequential way. Hence, periodical task will be assigned and distributed at every sprint according to the corresponding goal. Some task can be shared between suppliers and thesis worker under close collaboration, but they are mutually exclusive. Furthermore, we will execute validation and verification steps between the thesis worker and the selected team engineer or application engineer by giving mutual collaboration to avoid unnecessary time consuming from both parts.

Time and location

A master thesis worker at PPC/TESEC Control products department at Bombardier will develop this thesis. This will be developed under close cooperation and collaboration from the IDT department in Mälardalen University at Vasteras, Sweden. Furthermore, we will develop the thesis in 5 months where we have included validation and verification steps. Then, we have distributed prototyping workload to deliver several prototypes from May 2012 to August 2012. However, there is a burning issue related to procurement process and staff availability because it may delay some important activities related to gather software, requirements, documentation and other related resources or services. Thus, we shall fully follow the Scrum method in order to accomplish each sprint goal. Another assumption is that the master thesis worker expects to work 40 hours per week that includes meetings, courses, mentoring and reporting time. Besides, we also have considered holidays from April to August that may affect on the thesis planning.

Prototyping

In order to develop and deliver every prototype, we will organize the different activities according to the Scrum method. Thus, we have planned to deliver two releases with two sprints each in order to deliver four prototypes. This means that we have planned one release for the solution design and validation and another release for developing run-time prototypes.
In addition, we need to select a finished project, device, and stakeholders. Then, we can select a set of use cases and scenarios to narrow prototype functionalities and development activities according to the Scrum Method. This allows us to have a close collaboration and mentoring between the staff and thesis worker. In this sense, we will develop an initial prototype based on a model to validate concepts and design. Then, a low-level initial prototype will prove the possibility to change parameter values in running application process. This is followed by the packages transmission XCP over Ethernet at the DCU2 to check XCP feasibility. Finally, we will build client server calibration tool prototypes to prove the overall concept between calibration tools and DCU2. Therefore, we have described each prototype as follows:

<table>
<thead>
<tr>
<th>Prototype No.</th>
<th>Release No.</th>
<th>Title</th>
<th>Sprint No.</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>Model Simulation</td>
<td>1</td>
<td>A model will be designed to simulate XCP over Ethernet at DCU. This model will be composed of a behavioural description to be designed and simulated in Matlab/Simulink and TDL VisualCreator, and a state machine that contains XCP states to check properties via Model Checking techniques through UPPAAL tool.</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>Memory Page exchanging</td>
<td>2</td>
<td>Build a C script that will handle memory page exchange via XCP protocol layer functions to allow dynamical parameter changes based on few amount of parameters.</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>Client Server Scenario</td>
<td>3</td>
<td>A scenario to handle parameters via Vector tools will be developed to check workflow process. This will check responsibilities and roles management in a distributed team as well.</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>XCP/DCU2 Scenario</td>
<td>4</td>
<td>A scenario will be loaded at DCU2 to handle signals and few parameters via XCP. This will represent using or selecting a parser provided by the supplier that generates an A2L from a selected small group of rules that DCU2 application contains for handling parameters or signals and files. Therefore, it will run a process to load signals based on A2L files</td>
</tr>
</tbody>
</table>

Table 2: Prototype list definition

**Initial Costs**

Costs are low at the beginning of the thesis development because we can obtain evaluation licences to enables tool adaptation. Then, costs may increase significantly depending on number of floating licences, fixed licences, and different business cases that the implementation may require. Therefore, we consider that costs will be reconsidered and business cases will be re-formulated before implementing tool. In order to have an approximated cost for implementing any commercial tool in a given business case sample.
scenario, we believe that it is possible to consider different licence quotations that each supplier has submitted as a reference.

**Quality Assurance Plan**

Since we will design and develop prototypes during this master thesis, we will execute validation and verification steps in different phases to prove different concepts. Thus, the idea is to run validation and verification based on models, prototypes, and run-time environment. During every phase of validation and verification, there will be continuously validation from customer and stakeholders to check whether requirements are fulfilled. Then, key activities where customers and stakeholders will be involved to run this plan are described below:

- Design software architecture to describe and detail high level and low-level representations.
- Build and execute simulations based on models described into the architecture to run validation and verification steps.
- Select small scenarios, functions, and properties to load them at different prototypes as a pre-requisite to get a run-time environment in a small scale.
- Run prototypes in a real-time environment validation and verification in conjunction with the customer and stakeholders.
- Implement model-checking techniques to prove via mathematical formal methods whether a property from the XCP protocol is satisfied. In particular, a state machine that represents different XCP protocol events will be checked.

Consequently, we will design a behavioural and dynamic model in Matlab/Simulink to run different simulations based on a TDL model in TDLVisualCreator to check timing properties and protocol connections and interaction with the DCU2 application. Then, we will validate the protocol states through a state machine design via UPPAAL tool, which is a graphical model checker based on formal methods of validation and verification. Next, we will obtain DDL model via a C code generation from Matlab/Simulink process (Real-Time Workshop) to perform SiL activities according to each prototype’s needs. Besides, static descriptions will be designed in the software architecture in order to represent file, packages, layers and other software related components. After this, we will refine requirements and previous described architecture at each prototype sprint. Finally, requirement track will be updated continuously according to scrum plan and stories related.

**Risks**

Since this thesis requires that we handle several risks due to safety critical system nature, we will maintain traceability at every sprint between requirements, architecture, prototype, and risk plan. This traceability description will allow tracking most critical risks described in following subsections according to the risk management matrix. (See appendix A that describes a detailed risk matrix).

**Safety critical/Security:**

- Safety critical requirements were not properly defined, and product might not fulfil minimum safety critical properties.
- Improper safety critical function definition or design at tool may lead to Unit control critical failures.

**Software Design/Technical:**

- Inaccurate software design and development may lead into malfunction of any control unit component since XCP protocol permits a complete control over any
unit control.
- Prototype’s scope may be so wide and difficult to complete.
SOLUTION

According to what we have described in chapter two and three, we have selected the described methods and approaches for designing the solution that answer the research questions. This means that we will design a solution that implements ASAM standards such as XCP protocol and ASAP2, model based development techniques, and we will use commercial tools from the automotive industry. This will allow us to provide a solution that enhances team collaboration, control versioning, configuration management, and project life cycle management.

Therefore, we will describe in this chapter the commercial tools analysis, software design and its rationale for selecting the most convenient solution to each design concern. In addition, we will present and illustrate some important features from software architecture that we will design for implementing ASAM standards and commercial tools. We will also present a process workflow for handling the new philosophy and the system integration among protocols and tools.

4.1 Commercial Tools

Vector:

CANape 10.0:

CANape [9] is a tool designed for optimizing and enhancing ECU calibration in an iterative process. This tool assists engineers to perform tasks like rapid prototyping, get test benches or test drives, run parameter calibration, measurement and diagnosis. In particular, there is a physical interface between CANape and the ECU via Ethernet, CAN, Flex Ray and XCP. This allows having online and offline calibration modes. With respect to measurement mode, it offers different views like graphical representation, DAQ list configuration, virtual signals, Matlab/Simulink models, and managing calibrated data. This tool also provides interfaces to handle A2L files, calibration, and data management systems (for instance, database and profiles management system in a client server application) via eASEE. Another supported feature is model-based development that allows transforming Simulink models into data analysis and measurement models. It also provides simulation and scripting mechanism to run, analyze, and verify models through DLL files or XCP server in Simulink. Furthermore, it provides the necessary framework to run software validation and verification, and model base testing in the parameterization process. Thus, an explanation about model based development by using CANape and Matlab/Simulink is at the following section:

CANape in model based development by using Matlab/Simulink:

CANape [3, 18, and 29] also contains some important features that enhance model-based development by using SIL approach. In particular, it is possible to simulate and perform
measurements and calibration tasks in a Simulink model by using an XCP server. This server and other required functionalities belong to a Matlab/Simulink library that is included in CANape via plugin installation. This plugin allows generating a DLL model in Matlab M-files, M-scripts, exporting options, and mapping model objects with A2L of any device in CANape. Then, it also permits adding XCP blocks into any Matlab model and generating A2L files as well. Then, it is possible to reuse them into the calibration and measurement process as it is in figure 11. Furthermore, CANape has the option for visualizing Matlab/Simulink model without executing the Matlab application. The model visualization allows having an iterative process during calibration and measurement activities without modifying controller’s code because only it updates the model independently from the code as it is in figure 12. Therefore, Real-time Workshop generates the corresponding code until this iterative process has reached the final version of the calibration data.

Figure 11: CANape-Simulink Model Interaction

Figure 12: CANape-Simulink general model based development flow
eASEE:

eASEE [10] is a client server application that provides functions for process support in complex calibration projects. The tool provides a schema that enhances collaboration among distributed teams by giving support for roles, responsibilities, and user authorizations. The schema allows handling parameter status, data management models, project documentation, reports and process workflow. Besides, it provides some functions for quality assurance, parameter validations, and A2L file modifications. This tool also contains a parameter editor that assists engineers to perform comparisons and merge data between parameter values and projects.

ASAP2 Editor:

ASAP2 Editor [9] is a product that provides a framework for generating A2L files out of Map files, and it is part of CANape functions. The ASAP2 Toolset performs file generation for ASAP2 standard and contains functions for creating, updating, merging, and comparing the generated A2L files. In particular, it is possible to create A2L files from C-code object files and manage address between A2L file and the system target. The editor contains an interpreter and a parser that handles object files and symbol tables according to the system target description.

ETAS:

INCA base product:

Similar to CANape 10.0, INCA [11] provides a framework for handling measurement and calibration systems. In particular, this tool contains a set of editors like hardware configuration, calibration scenario, variable selection, and memory page management that allows engineers to run the calibration/measurement configuration. Hardware Configuration Editor provides software based on a replication of the target hardware to run calibration and measurement tasks via experiment environment generation. Then, Calibration Scenario Editor describes set of calibration and references variables for setting parameters in a given calibration scenario. Thus, variable selection and experiment configuration editor define variables from different calibration scenarios. Furthermore, the Memory Page Management Editor is responsible for memory page and flash programming management to allow downloading/uploading and working with memory pages (for instance, reference page and working page). In addition, Calibration Data Manager and Measure Data Manager handle the data management from both calibration and measurement. Both managers allow generating local documentation, file exporting, and A2L file management.

dSPACE:

Control Desk Next Generation:

Control Desk Next Generation [12] is software that implements a philosophy to develop ECU software based on experiments that replicates ECU developing for rapid prototyping. This allows development teams to get necessary working environment at all experimenting stages through handling modules for diagnostics, calibration, measurement, software testing, validation and verification as it is shown in Figure 13. In specific, it provides necessary framework for measurement and calibration tasks through Control Desk Next Generation basic version. This allows displaying different layouts and instruments for calibration tasks, managing projects, data sets, signals variables and plotting data. Besides, it contains a tool automation to handle scripts to customize tools according to customer needs. For instance, it is possible to run certain calibration tasks by scripting some options so it is possible to
automate project creation and configuration or filtering data sets according to project functions.

### Module Overview

![Module Overview Diagram](image)

**Variable Editor:**
Similar to ASAP2 editor, Variable Editor [13] provides a framework for editing, visualizing, and creating ECU description file (A2L files). Moreover, it allows importing and exporting A2L variables according to specific requirements or characteristics. It also manages map and hex files to handle address, symbols, and signals because it has address update automation options via command line interface.

### 4.2 Commercial Tool’s Solution Selection

We have made a decision about selecting the most convenient commercial tool that were described in section 4.1 in order to apply calibration and measurement concepts described in chapter two. Thus, the solution selection was based on criteria-concern analysis where the concern corresponds to this question “Which calibration/measurement tool and supplier is the most suitable solution?”, and the studied criteria were scalability, flexibility, security, risks, process design orientation, performance, efficiency, cost and supplier availability and support.

**Motivation for Tool Selection**
The solution that we have selected is to use a combination from Vector’s tools that are CANape, eASEE, and ASAP2 editor. This is because Vector’s tools can fulfil the thesis assumptions, provide the required evaluation licences for each product, and give us the required environment for fast prototyping. Additionally, it implements the different
approaches for model-based development described in section 2.1. The following sub-section
describes the rationale from each option:

Evaluation of the Commercial Tools for Parameterization

CANape, eASEE from Vector

**Scalability:** CANape and eASEE are scalable since they offer a component based solution that
enables system reusability and reduces redundancy. Furthermore, it supports model-based
architecture that enables further implementations to improve processes.

**Flexibility:** tools are flexible enough because they provide a set of interfaces based on ASAM
standards that makes this tool compatible with other commercial tools or further in-house
developments. They also have the required framework and process workflow to enhance
distributed development environment.

**Security:** this solution offers authentication methods, profile management, password
encryption, authorization levels, and other mechanisms to enhance security in distributed
development. Furthermore, it provides a methodology based on configuration management
and control version for a calibration project lifecycle according to client server style because
project and parameters information is stored in a database instead of storing them in local
files. Moreover, this methodology enhances a secure eASEE server access from any client by
using authentication methods, network firewall, and other protection against system
intrusion.

**Risks:** it offers different mechanism to execute preventive activities when any risk is present
due to process workflow orientation. Besides, vector offers good quality products since they
are certified and based on ASAM standards, so CANape and eASEE provides a riskless
environment according to risk analysis stated at the risk analysis.

**Process design orientation:** it provides a new philosophy for handling parameter, calibration
tasks and ASAM standards based on process oriented solutions, where eASEE manages
responsibilities, profiles, documentation, approvals, control version, etc for a distributed
development automatically. This enhances productivity by reducing bureaucratic steps that
are time consuming.

**Performance:** performance is high since the designed system is accurate for distributed
teams according to number of clients and controllers to calibrate. This implies that hardware
requirements shall meet Vector´s technical specifications.

**Efficiency:** a client server calibration/measurement tool enhances efficiency in a distributed
team since it offer an accurate process workflow.

**Cost:** costs are low at the beginning of project because we can get evaluation licenses to
enables tool adaptation. Then, costs may increase significantly depending on number of
floating licenses, fixed licenses, and different business cases that the implementation may
require. Therefore, we will need to reconsider costs and business case analysis before
implementing tool in the organization to get costs estimations according to its capacity.

**Supplier availability/support:** vector provides enough support and it is customer oriented.

Conclusion: we have selected this option since it offers the required background to develop a
fast prototype for the thesis project based on assumptions stated at the project scope.
Additionally, the solution offers an evaluation license that allows building the needed
prototyping environment. Moreover, it is flexible enough to provide scripting, model base
testing and process workflow scenarios to cope with parameter handling problem domain.
**INCA from ETAS**

*Scalability:* INCA has a medium scalability since it offers required interfaces to access to other components in a client server structure, but solutions are not component based and every editor are closed and difficult to customize.

*Flexibility:* it is not flexible enough because INCA provides closed and supplier dependant interfaces that can allow us developing a process workflow according to profiles, responsibilities and distributed teams.

*Security:* INCA has a high security with respect ASAM standards and memory pages control version. However, it doesn´t offer authentication mechanism and client server configuration management, so they have to be developed in a separated client server application in conjunction with the database. Besides, control version is local and it doesn´t support distributed control versioning system. This implies implementing this feature at the proprietary client server application.

*Risks:* it offers different mechanism to execute preventive task when any risk is present due to process workflow orientation. Besides, vector offers good quality products since they are certified and based on ASAM standards.

*Process design orientation:* it does not provide a process oriented solution to cope with handling profiles/responsibilities requirement. Therefore, we shall perform an extra workflow analysis and development in order to fulfil this requirement.

*Performance:* performance is high since the system design is corresponding to controllers’ characteristics. This implies that hardware requirements can meet ETAS´s technical specifications.

*Efficiency:* INCA is not efficient enough for supporting distributed development because it is not client server based. Besides, the tool is difficult to adapt into this environment because configuration management and control version is local.

*Cost:* costs are high at the beginning of project because there are not evaluation licenses available to develop tool prototyping. Then, costs may increase significantly depending on number of floating licenses, fixed licenses, and different business cases that the implementation may require. Therefore, we will need to reconsider costs and business case analysis before implementing tool in the organization to get costs estimations according to its capacity. Besides, we shall implement a big effort in order develop client server solutions and parameter handling philosophy that implies a rise on development costs.

*Supplier availability/support:* ETAS provide enough support.

**Conclusion:** we have rejected this option since it does not offer the required background to develop a fast prototype for the thesis project based on assumptions stated at the project scope. Besides, it requires developing client server solution that may be time consuming and expensive for a thesis prototype. Furthermore, this supplier does not offer evaluation license, so initial costs are high with respect to other options.

**Control Desk Next Generation (CDNG) from dSPACE**

*Scalability:* CDNG has a high scalability since it offers the required interfaces to access to other components in a client server structure, and it provides an Automation tool to customize and evolve the software according to what the customer needs.
**Flexibility:** it is flexible because the Automation Tool allows developing applications according to customer’s needs. However, there are not client server solutions, so we shall develop them separately in order to increase flexibility

**Security:** CDNG has a high security with respect ASAM standards and memory pages control version. However, it doesn’t offer authentication mechanism and client server configuration management, so they have to be developed in a separated client server application in conjunction with the database. Besides, control version is local and it doesn’t support distributed control versioning system. This implies implementing this feature at the proprietary client server application.

**Risks:** it offers different mechanism to execute preventive task when any risk is present due to ASAM specifications. Besides, dSPACE offers good quality products since they are certified and based on ASAM standards.

**Process design orientation:** it does not provide a process oriented solution to cope with handling profiles/responsibilities requirement. Therefore, we shall perform an extra analysis and development in order to fulfil this requirement.

**Performance:** performance is high since the system design is corresponding to controllers’ characteristics. This implies that hardware requirements can fulfil dSPACE’s technical specifications.

**Efficiency:** CDNG is not efficient enough for supporting distributed development because it is not client server based. However, we can improve this drawback significantly via scripting developing to support different roles.

**Cost:** costs are low at the beginning of project because we can get evaluation licenses to enables tool adaptation. Then, costs may increase significantly depending on number of floating licenses, fixed licenses, and different business cases that the implementation may require. Therefore, we will need to reconsider costs and business case analysis before implementing tool in the organization to get costs estimations according to its capacity. Besides, we shall implement a big effort in order develop client server solutions and parameter handling philosophy that implies a rise on development costs.

**Supplier availability/support:** dSPACE provide enough support.

**Conclusion:** we have rejected this option since it does not offer the required background to develop a fast prototype for the thesis project based on assumptions stated at the project scope. Moreover, it requires developing client server solution that may be time consuming and expensive for a thesis prototype although this supplier may offer evaluation licenses.

### 4.3 General Design Description of Developing Prototypes

We have designed the solution for implementing a new parameter handling philosophy in this thesis by implementing component-based architecture that decomposes the software system into sub-systems. One sub-system implements a client server application through the eASEE tool to manage parameter calibration. This also contains CANape as the calibration tool that provides the client access via function call, and this is responsible for handling different parameter configuration levels and administrates ASAM standards. Both eASEE and CANape can be on the same terminal or in different terminals. The other sub-system represents a real time systems event handling that is composed of a layered architecture that describes different features from XCP protocol, Vector’s specifications, and DCU2.

In this sense, we have designed a deployment diagram, to represent different physical
and software components that we will deploy for the new philosophy (see Figure 14). This diagram describes three main components that are Calibration Management System (client server component), CANape, and DCU2. The last component is composed of several layers such as application, protocol, transport, and interface. Hence, the component diagram in Figure 15 implements previous component organization into a master /slave philosophy according to the XCP standards [17]. Therefore, the set of ports in this diagram represents different access between one layer to other one and layer’s realization. A master component is composed of eASEE clients (standard client, administrator, and configurator clients) and CANape tool. XCP/DCU2 component and its layers represent the slave component. A more description about the software architecture is explained in the appendix B.

Figure 14: deployment Diagram
Figure 15: Component Diagram
4.4 Use Cases organization per prototype

According to section 3.4 and table 2 that states a list of different prototypes, we have defined a set of initial use cases in order to meet each prototype goal. Thus, we have described a relationship between each prototype, use case short description, and diagrams below:

<table>
<thead>
<tr>
<th>Prototype ID</th>
<th>Use Case ID and diagram number</th>
<th>Short Description/remarks</th>
</tr>
</thead>
</table>
| 3.4          | See Figure 16 and 17           | UC7: Create Calibration Project: this use case represents creating and configuring projects in eASEE system and CANape.  

UC7.1: Configure product attributes: this allows configuring products attributes for a software key related to the calibration project.  

UC8: Modify Calibration Project: this allows to modify the calibration project properties  

UC9: Create Dataset: this represents a container for holding parameter files that corresponds to a set of calibration data. This may be organized by function view or group view.  

UC10: Set up calibration ownership: this permits configuring settings for assigning tasks to calibration engineers and other stakeholders.  

UC11: Handle Historical logs: this allows seeing an historical log about project changes, calibration data, parameter sets and other related information.  

UC12: Handle Reports: this is able to generate reports according to the calibration tasks.  

UC13: Handle baselines: this creates a baseline for calibration projects.  

UC14: Deploy data sets and parameter: this generates the deployable version for a dataset in case of eASEE. For CANape, this case generates a deployable parameter file for the software product.  

UC15: Handle profiles: this allows configuring roles and responsibilities at the eASEE administrator tool.  

UC16: Deliver parameter sets: this allows delivering parameter sets into the parameter file at the eASEE.  

UC17: Login: this allows to eASEE users to logging into the system and this includes Authentication Methods.  

UC17.1: Authentication Methods: this allows defining authentication method kind to different users at the eASEE administrator tool.  

UC18: Handle Calibration data: this use case allows
performing basic calibration configuration, online and offline calibration and measurement.

**UC19: ASAP2 Database configuration:** this permits set up transport layer and devices configuration into the A2L file descriptions. This will enables the physical connection between master and slave under XCP over Ethernet.

**UC20: Generate A2L:** this enables the generation of the A2L file by extracting and handling object files from the software after compilation process in order to extract parameters information.

<table>
<thead>
<tr>
<th>1,2</th>
<th>Figure 18, UC1-</th>
<th><strong>UC1:</strong> XCP Configuration, a set of initial definitions and base configuration has to be set in the DCU2 component in conjunction to commands configuration in order to get the necessary base XCP configuration to execute different XCP layers.</th>
</tr>
</thead>
<tbody>
<tr>
<td>1,2</td>
<td>Figure 18, UC2, UC3</td>
<td><strong>UC2:</strong> Handle Master/Slave connection, different there are different task definitions at the DCU2 to manage master-slave connection over Ethernet. The defined tasks are responsible for setting up a session, synchronization, and initialization. They represent associated used cases respectfully.</td>
</tr>
<tr>
<td>1,2</td>
<td>Figure 18, UC3, UC2, UC1</td>
<td><strong>UC3:</strong> Memory Page, this implements different memory page management functions to control application layer every time that an XCP command is executed to request different associated use cases such as get page, set page, copy page and handle pointers.</td>
</tr>
<tr>
<td>1,2</td>
<td>Figure 18, UC3, UC2, UC1, UC3.1, UC3.2, UC3.3, UC3.4, UC3.5</td>
<td><strong>UC3.1: Get Page,</strong> this use case is responsible of getting the memory page required by a XCP command and it includes check memory status use case (UC3.4). <strong>UC3.2 : Set Page,</strong> this use case is responsible of setting the memory page required by a XCP command and it includes check memory status use case (UC3.4) <strong>UC3.3 : Handle pointer,</strong> this use case is responsible of managing memory pointers when a memory page is get or set and it includes check memory status use case (UC3.4) <strong>UC3.4: Check memory status,</strong> this use case is responsible of checking memory availability and access wherever any command or resource request a memory page. Therefore, it implements mutex property described in section 6. <strong>UC3.5: Handle files,</strong> this use case is responsible of managing files in order to create, read and write parameter files. This includes check memory status use case (UC3.4)</td>
</tr>
<tr>
<td>3,4</td>
<td>See figure 18</td>
<td>This represents use cases UC20 and UC19</td>
</tr>
<tr>
<td>2,3</td>
<td>Figure 18, UC6</td>
<td><strong>UC6: Handle Packages,</strong> the transport layer is responsible</td>
</tr>
</tbody>
</table>
of managing XCP packages that will contain the calibration data. The associated use cases are build package, send package and receive package.

Table 3: Use Case Organisation

Figure 16: General Use Case Diagram
4.5 State Machine Description

As part of prototype 1, we have designed a state machine in order to specify different constrains and states that XCP protocol shall have to perform model-checking technique. This specifies the precedence relations between all protocol states as well. In particular, each state represents a task to be executed by the operative system in a given period. In this sense,
we have represented a reactive behaviour in a given environment handled by commands that came from calibration tasks and XCP configurations according to ASAM specifications [17]. This state machine was designed and checked with UPPAL tool in order to verify the correctness of the proposed protocol previous to development, and this was done by following the stated rules according to “UPPAL in a Nutshell” official white paper [19]. We have described each state below:

- As it shown in Figure 19, this state machine describes a set of modes, states, and required guards to go from one task to other during protocol execution.
- From starting state XCP protocol may go from resume mode to other mode. When resume mode is true, it goes to transferDTO otherwise it goes to disconnected.
- If it is connected, it can execute any command and remain into connected state if only if a command send a response that may have a error level less greater or equal to 2. Else, it goes into a disconnected state.
- If other mode that is different from resume mode is true, then it will remain connected.
- If other mode that is different from resume mode is false, then it will remain disconnected.

![Figure 19: XCP Protocol State Machine](image)

### 4.6 Design Decisions

We have made a set of design decisions about selecting the most convenient option to solve different design issues or questions that represent a significant impact into software development for different prototypes. Thus, we have based our solution selection on criteria-concern analysis where the concern corresponds to some questions and concerns. We have made other important decisions to check whether we can fulfil the requirements or not. The following sub-section describes the rational from each option and the corresponding concern that are according to the RTOS characteristics from VxWorks and Hardware resources in the DCU2 [20, 21]:

41
Concern: Which method shall we use for checking memory and resource availability?

Criteria: Functions developed in operative system, complexity, time effort. Mutex function for RAM and Checksum services for persistent memory

Rationale:

- Functions developed in operative system: there exist all the needed functions for handling mutex semaphore in RAM and checksum services for handling file security.
- Complexity: code development is not so complex because of the function definitions at the API, so we only need to design a flow for calling them into the code.
- Time effort: this provides code reusability since it implements already defined functions and reduces time effort from developer.

Decision: we have selected this option because we have defined and tested functions in the RTOS API. Thus, this decreases time effort from developer.

Checksum services for persistent memory and RAM

Rationale:

- Functions developed in operative system: there exist all the needed functions for handling checksum services for handling file security. However, there are not functions defined at the operative system for handling RAM.
- Complexity: code development may be complex because there are no functions for handling RAM memory.
- Time effort: the developer will spend significant time in defining and developing functions for handling checksum services in RAM.

Decision: we have rejected this option because the developer may need to make significant effort on implementing functions for handling memory in RAM.

Concern: Which persistent memory must we use for storing parameters when DCU2 is off or disconnected?

Criteria: security, maximum memory storage capacity, back up resources, memory space availability, and memory overload risk.

File System

Rationale:

- Security: unexpected power loss may corrupt file information. Then, we must implement checksum control.
- Maximum memory storage capacity: it has limit of 13MB.
Back up resources: it has an extension for back up of 32MB.

Memory space availability: there is availability for parameter storage up to 3MB, and it contains sufficient space for storing big amount of parameters because it is possible to handle around 5000 parameters that may have a size of 2MB in total. (Based on an assumptions and address size specifications from ASAM standards)

Memory overload risk: there is a riskless memory overload since there is sufficient space for storing parameters into an application.

**Decision:** we have selected this option because it offers a riskless memory overload, back up resources and enough memory space capacity in a worst-case scenario. However, it is necessary to consider parameter amount and size as requirement for long scale projects when DCU2 software from the common software design.

**Non-Volatile RAM (NVRAM)**

**Rationale:**

- Security: if there were an unexpected power loss, stored data at the memory would be corrupted. Then, checksum control must be implemented.
- Maximum memory storage capacity: it has limit of 1MB.
- Back up resources: there are not resources for back up.
- Memory space availability: there is a limited space of 500KB
- Memory overload risk: there is a high risk of memory overload in projects where huge amount of parameters may need to be configured.

**Decision:** we have rejected this option because it does not offers a riskless memory overload, back up resources and enough memory space capacity in a worst-case scenario.

**Discussion about requirements fulfilment during XCP-Train Device Integration**

In order to integrate XCP application layer with train devices, we have designed the train device integration based on previous point’s discussion [16, 17, 20, 21]. Thus, an initial schema about this integration states that it is possible to get parameter values via A2L files. The interface layer that can process object/Motorola files from A2L in both modes handles the set of files: online and offline. Then, the transport layer will manage object/Motorola files in order to send and request some needed information to protocol layer. Therefore, the application layer will handle this information by function call to protocol layer and the train application. Hence, any device can get parameter values from A2L information that is at CANape. Besides, there exists offline mode and online mode. In online mode, the interface layer manages A2L file information according to the previously described. In case of offline mode, it is possible to generate C code from some CANape’s templates to handle parameter by function call. Then, this code is added as a resource and downloaded into the target. We consider important to clarify that this step shall be executed once that calibration tasks have been completed. This is described in Figure 20.

In addition, we have defined some software constrains during the development process in order to fulfil ASAP2 standards [16, 17] and the operative system VxWorks requirements [20, 21]:

- There is no need to generate a parser to handle map and object files since CANape database editor provides the required mechanism to extract parameter information. Therefore, Vector has generated a patch for enabling database editor
to read VxWorks object files according to GNU compiler specifications.

- We have defined that offline mode is required for the first time to gather parameter information from object files in order to generate A2L file. Then, we require this process to deploy parameters into product C code once calibration and measurement task were completed and certified.

- We need to compile and merge parameter files into the product code and downloaded into the DCU2 once that we have deployed the parameter’s information into H files Therefore, parameters will be assigned by function call by other devices or application functions.

- There is one limitation with respect to this approach because we believe that it is necessary to make changes into software organization and parameter identifiers since the database editor do not support type definition structures in the symbol table for a set of parameters(as it is currently described in the train application code). In addition, the database editor pre-requisite states that parameters shall be defined as a global variable at the first time in order to get the complete symbol table description according to GNU compiler. When parameters will be deployed, they will get static identifier into a header file in order to allow the inclusion of this file at the main c code and the function call process.
Figure 20: Online Calibration proposed workflow process

1. Compile application
2. Generate link and object files
3. Run train application
4. Call transport layer
5. Connect to DCU2
6. Call application by function
7. Deploy parameters into the final product
8. Perform measurement and calibration
9. Run train application
10. Start calibration?
   - No: Download files to the target
   - Yes: Generate Motorola file from debug file
8. Start CALiPse and configure device
9. Read and interpret debug file
10. Configure Motorola file
11. Generate A2L file
12. Online calibration?
   - No: Generate .diz files
   - Yes: Generate .out files
13. This contains compiled and debug information which can be used to generate all files and is called debug files
14. This represents the software artifact to be downloaded to the target
15. Object files may represent map file or downloadable artifact
16. Deploy parameters into the final product
17. Go to production
18. Calibration finished?
   - Yes: Go to production
   - No: Go to online calibration

This is an initial workflow process for calibration and deployment of train application product. This may be adjusted according to the EE requirements for the new controller, but it cannot be modeled for current controllers.
Kapitel / Chapter 5

RESULTS/EVALUATION

According to what we have explained in chapter 4, we have designed a solution that implements ASAM standards, model based development techniques by using Vector’s tools that answer the research question. The designed solution also provides mechanism enhance team collaboration, control versioning, configuration management, and project life cycle management in the new parameter handling philosophy.

Hence, we present in this chapter an evaluation of results from model simulation and code execution in different prototypes development. Besides, we have organized this chapter into a set of sections were we state prototypes results, recommendations. Therefore, each developed prototype is corresponding to prototype matrix described in section 3.4 at table 2 and the use cases described in section 4.4.

5.1 Results discussion of different prototypes

We have found some important results from one prototype to another. This section aims to present them briefly. In this sense, we have designed and executed simulations during prototype 1 in order to validate and verify standard specifications from XCP protocol. Then, we have performed a simulation at the controller in the prototype 2 that is based on a set of sample scripts that describes what was proposed at the simulated models to check implementation feasibility and reduce risks due to the system complexity. After that, we have developed prototype 3 in order to start implementing ASAM standards in an early stage to reduce integration risks. Finally, we have described in prototype 4 a full standard implementation concept based on previous prototype results.

Some important results that we have obtained in prototype 1 have represented an input for upcoming prototypes. In particular, the executed simulations and model checking tasks that we have made at UPPAAL tool during prototype 1 (see Figures 21 and 22) [19]; the XCP protocol state machine is able to go to all requested states without having a deadlock state. Besides, due to the nature of the XCP protocol, it will process several commands at the same time so this state machine has more than one instance at the simulation to check when all different command types might be processed. However, this state machine is not able to control different resources and managing different command request and responses in a synchronous mode. Thus, we have decided to implement polling method initially to run periodical tasks in a given period. Hence, we have considered that we need to implement queue methods and other algorithms to improve performance during measurement tasks. In particular, it is required to handle a list with the different elements and data to be calibrated and measured through XCP protocol according to ASAM specifications [17]. In addition, we have executed simulations via Matlab/Simulink environment to check the logical time execution behaviour in a given period of sample time [22, 23]. In this sense, we have described a TDL model [25] in order check this behaviour. We also have was configured a
time trigger event based on 10 ms as a sample time both XCP protocol and memory page module to examine event driven behaviour in the protocol. An important result that we have obtained from this simulation is that a given task can be managed in a defined period and they can be synchronously executed according to the specifications that are configured at Mitrac Tools. However, we have considered that it is important to state a scheduler to handle different periods according to what function needs. We also realized that it is needed to improve performance during measurement task by using event-triggering policies to enable required interrupts when different events are requested. This is also suggested by ASAM specifications at the corresponding sections.

Figure 21: XCP/DCU2 state machine simulation
With respect to prototype 2, we have developed a memory page prototype in C by following VxWorks operative system functions and some standards stated at Bombardier’s documentation and Operative System API [20, 21]. The C script execute following tasks:

- **Init task**: generate memory segments and pages and handle a preliminary configuration.
- **Special tasks**: They will be responsible to handle transport layer, and mutex protocol for memory page/segments.
- **Cyclic tasks**: to test client connection to the transport layer and waiting and taking states from mutex protocol for memory page and segments.

The result from this set of c scripts can show to us that it handles memory page and segments successfully. Then, it saves this information into a binary file before transportation layer is executed. This procedure simulates memory allocation in both RAM memory and file system. After this, transport layer executes socket handler that is currently processing client messages. This behaviour represents the initial background required to execute and process XCP protocol commands [24].
Figure 23: DCU2 transport layer communication test

Figure 24: Online measurement and calibration based on SIL approach
In order to develop prototype 3, we have selected a set of case scenarios from use case description stated in section 4.4. Therefore, a list is stated below:

- Offline calibration and a2l file generation
- Deployment of calibrated parameters from offline calibration
- Online Measurement and Calibration based on SiL
- DCU2 transport layer communication test
- Loading a sample project in eASEE by standard DTS and group view methodology
- Loading other sample project in eASEE by standard DTS, function based and group view methodology
- Create Baselines for previous sample projects and deploy the project into eASEE configuration management methodology for standard DTS
- Sample case for assigning roles and responsibilities in a calibration project

Therefore, for creating offline calibration and A2L file generation [26], we have developed a set of sample functions in C, and we was updated the application layer from prototype 2 in order to handle some cyclic tasks that run those functions. After this, we have imported object files at the database editor in CANape in order to create the A2L files. At the same time, we have made a connection between CANape and DCU2 via Ethernet to check socket server connections. This is because we had to make a configuration at the device manager in CANape that adds this description at the A2L file. Thus, results from this test can be seen at Figure 23 that shows a successful testing connection and disconnection when offline mode is activated as it is displayed in the figure. In addition, by following previously stated procedures, we have made online mode scenario based on SIL approach and some simple Matlab/Simulink models were defined [29]. Then, a DDL driver (this emulates an ECU environment under PC platform) was generated from Real-Time Workshop code generator from Simulink based on that defined model. Then, an additional init model is generated from this process also in order to fulfil some model requirements from CANape tool [26]. This code generation also produces A2L files based in functions view and DDL. After that, we have made another sample configuration in CANape to run online measurement and calibration via DDL driver. The result is stated in Figure 24 that displays a user friendly and successful online calibration and measurements for model-based development.

Afterwards, in order to run online and measurement calibration for distributed teams we have loaded a set of project examples into eASEE tool [28]. There are a set of constrains and rules to be followed in order to fulfil eASEE methodology pre-requisites into a calibration project development. A calibration project requires one software key. This software key is composed of a product key, product attributes, and a variant. By following that constrain, we have loaded some sample projects in order to deploy previously defined scenarios. Therefore, we have applied each methodology combination from one project to another and they were successfully created and managed by the tool. This is described in Figure 25 and 26.
Figure 25: Parameter classification Conceptual Diagram and Constrains
Figure 26: Sample project definition that reuses parameter definitions and functions.
With respect to prototype 4, we have discussed, designed, and proposed a more advanced model in Matlab/Simulink, parameter organization and classification in a sample eASEE project [29], and we have refined the transport layer from DCU2 component. In this sense, we have described a sample train model in Simulink by incorporating CANape blocks to get parameter and signal definitions. This set of definitions allows CANape to compute parameters automatically by a given reference at this tool by global variable and function definitions. Thus, a measurement configuration [26] was loaded and a .net panel that handles a script in order to create test branches to compute parameters based on a defined mathematical behaviour. For instance, one button from this script generates a step response that computes a torque reference to calculate the velocity of the train, so it is possible to have a faster acceleration into the train by controlling all related parameters. This can be seen in Figures 27 and 28 that state a positive acceleration when it has a positive torque reference.

In addition, we have executed successfully the integration between protocol layer, transport layer, application layer, and interface layer at the DCU2. In this case, we have modified the init task in order to start the protocol layer. After this, we have included transport layer refinements in order to handle packet transmission, acknowledge messages and byte order management according to CPU’s endianness. In this case, the CPU endianness that we have considered was PowerPC big endian (Motorola) according to [27, 20, 21]. This enables the transport layer to build packages and send the correct positive response to be interpreted by the interface layer in CANape with respect to ASAM standards. In addition, we have configured polling methods at CANape in order to enable online calibration measurement based on this schema [16, 26]. Then, CANape was connected to the DCU2 via Ethernet and it resulted into a positive communication that transmits properly acknowledge messages to CANape. This has enabled online successful online calibration and measurement based on simple sin functions in a sample application layer. This is described in Figures 29 and 30. However, this schema represents a low performance into real time measurement because it sends the data once at time. Thus, we believe that we shall implement event channels in a further work in order to enhance performance because it will process huge amount of data into a list that is wrapped at the package and controlled by events.

Subsequently, we have updated previously stated sample projects in prototype 3 into a better parameter classification according to group view for prototype 4. In this sense, we have defined a system configuration in order to handle specific parameters at the functions defined at the train Matlab/Simulink sample model that are dependent on whether the train runs with alternate current or discrete current. This system configuration was organized into integrated data set that holds partitioned datasets according to figure 26 at eASEE tool [28]. Then, we have defined another dataset to holds general parameters that are independent from the system change. In addition, we have set the user rights definitions in order to separate responsibilities among calibration engineer users according to a specific group of parameters into a distributed team environment. Besides, we have used a program set definition in order to clearly separate the software image, A2L files and parameter files. This program set is an object into the database that is referenced by previously defined datasets. The result that we have gotten represents a good strategy to organize parameters according to functions and responsible. Therefore, it enables code, projects, and configuration reusability because it is possible to export/import previous configuration from another projects that uses the same software. This reduces the redundancy and overhead into a distributed team via eASEE. This is shown in Figure 31.
Figure 27: Sample train model based on Simulink for SIL approach.

Figure 28: Sample .Net panel that holds test benches for generating step response from the train model.
Figure 29: DCU2 sample project that holds online calibration and measurement
Figure 30: DCU2 responses that fit with the online measurement and calibration.
Figure 31: Sample project that implements train system configuration and parameter organization.
In particular, scrum methodology has significantly helped us to fulfil every prototype requirement on time. This is because; we have organized project planning in an efficient way by distributing activities and updating project planning as long as it was needed. Therefore, we have organized tasks and stories according to 3 releases that were divided into 2 sprints each one. Every planned sprint has represented each prototype described above and were related to the planned milestones in section 3.4. Thus, requirements and software architecture was refined in every sprint in order to enable iterative process in conjunction with validation and verification activities. An example of stories organization in the last sprint is shown in Figure 32 by using Ice Scrum tool. Consequently, project risks stated in appendix A were mitigated or eliminated considerably, and it contributed to have a good communication between tool’s supplier (Vector), university’s supervisor, and Bombardier’s supervisor as well.

5.2 Recommendations

- We suggest implementing seed and key mechanism to enforce security at XCP prototype to avoid network intrusion and sniffing the protocol’s data during online calibration and measurement process.
- We recommend refining measurement process in order to improve system’s performance. This can be achieved by enabling event driven measurement every 10ms or 100ms.
- We advise to modify current software organization of the train application and variable definitions in order to make them as global to enable XCP access in the memory for online measurement and calibration. We consider that variables and signal definitions shall not contain identifiers like constant, type definition or static due to XCP protocol constrains. This mechanism will force the compiler and the operative system to access them into the write accessible memory segment. Another important constrain that we suggest to follow is that variables shall not be structure data type because this will prevent the A2L file reader to extract the required symbol information and memory addresses from object files. Additionally, the new software organization at should be based on a layered
architecture that includes XCP protocol philosophy, and component based architecture. This means that the train application will be decomposed into independent components that represent certain functions that will be handled by the application layer.

- We recommend having further discussions and workshops about refining system configuration mechanism at the DCU2 based on eASEE’s methodology and deployment process.
- We suggest implementing the presented solution in a small project, and having a transition phase or pilot program among application and software engineers to get further feedback for improving the solution.
- In order to guarantee a successful implementation in more complex project, we consider that it is important to handle training programs and workshops with the supplier in order to smooth the transition phase.
- We suggest building business case scenarios for the different Bombardier sites or divisions that are interested on using this tool during the transition phase or pilot program. This will reduce costs considerably since a fraction from the staff will test the tool instead of the whole organization. Then, further business case scenarios can be reformulated to get the required funds for completing the tool chain implementation.
- Once those previous steps have been implemented, we believe that it is important to get XCP protocol certification and the membership with ASAM association in order to provide the required legal and commercial approvals to be used into Bombardier’s products.

In summary, the evaluation of results has shown that we have optimized parameterization process at Bombardier by implementing commercial tools and standards. Therefore, we have decreased significantly the delays, overhead in configuring controller lack of documentation, lack of expertise and reliability at the data generated by introducing a process workflow for the parameterization of the DCU2 through ASAM standards, eASEE, CANape and ASAP2 tools. This process workflow has introduced an efficient lifecycle management for DCU2 parameters because it reduces redundancy, implements configuration management, and controlled versioning, manage roles and responsibilities; enhance team collaboration and improve the documentation management.
CONCLUSIONS AND FUTURE WORK

In this thesis, we address the DCU2 parameter configuration at the software development process, currently conducted at Bombardier. The current process at Bombardier uses an excel workbook to set up and exports the parameters when the train application is compiled and further downloaded into the target system. This approach consequently causes a number of drawbacks during the software development process, e.g., delays in the validation and verification steps, system configuration overwork, as well as suboptimal system reliability and performance. To address these issues, we first performed a review of the state of the art in the field, covering academic and as well industrial research, in order to be able design and implement a new philosophy for dynamically handling parameters during software project development. We show that the proposed solution has the ability to improve Bombardier’s software development processes by increasing the development efficiency and team collaboration into software life cycle management, as well as enabling a better view on the system, towards optimizing the overall performance.

In addition, the proposed approach for efficient parameter configuration at the train controller implements methods from model based development in transport industry, which optimizes the validation and verification steps during the parameter configuration process. This was possible by introducing online measurement and calibration procedures via XCP protocol, commercial tools from Vector Informatik like CANape, ASAP2 editor, and eASEE, and modeling languages like Matlab/Simulink, TDL, and UML. We developed a set of 4 prototypes, validated and verified successfully by following scrum methodology, in order to prove the proposed approach.

We suggest the implementation of the proposed solution in a small project, and having a transition phase or pilot program among application and software engineers that can provide useful feedback towards applying further improvements. Additionally, we believe that running a set of workshops and training programs is beneficial in order to smooth the transition phase. This also includes proposing different business case scenarios for pilot programs or full implementation in order to enforce cost analysis towards, e.g., successful certification from ASAM.

Finally, our recommendation is a reorganization of the software design during the pilot program to consider the previously described constrains and parameter classification methodology. In other words, design a software organization according to functions and models to maintain the traceability between the parameters and the software. Moreover, we recommend that the new organization should be based on a layered architecture that includes XCP protocol philosophy, and component based architecture in order to enhance the traceability between the parameter configuration and the software application.
Kapitel / Chapter 7

REFERENCES


### APPENDIX A-RISK MATRIX

<table>
<thead>
<tr>
<th>Risk ID</th>
<th>Risk Title/Category</th>
<th>Effects</th>
<th>Probability</th>
<th>Impact</th>
<th>Exposure</th>
<th>Trigger Events</th>
<th>Strategy</th>
<th>Preventive Task</th>
</tr>
</thead>
<tbody>
<tr>
<td>01</td>
<td>Requirements</td>
<td>Requirements have been specified, but they are continuously changing.</td>
<td>M</td>
<td>H</td>
<td>M</td>
<td>Receive courteously complains from team</td>
<td>Mitigate</td>
<td>There shall be a requirements agreement for each prototype that must be followed</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Lack of requirement specifications.</td>
<td>M</td>
<td>H</td>
<td>M</td>
<td>Designer/developer cannot understand specification details</td>
<td>Mitigate</td>
<td>Requirements must be granular enough and shall be described during designing tasks</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Requirements are so complex.</td>
<td>L</td>
<td>H</td>
<td>M</td>
<td>Neither Suppliers nor team understand how to fulfil the requirement</td>
<td>Mitigate</td>
<td>Complex requirements shall not be considered</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Requirements were defined, but they were not agreed between stakeholders or suppliers.</td>
<td>M</td>
<td>M</td>
<td>M</td>
<td>Stakeholders or customer didn´t approve requirements on time</td>
<td>Mitigate</td>
<td>If there is a missing approval, it is important to inform this to the Manager to scale an agreement.</td>
</tr>
<tr>
<td>02</td>
<td>Timetable tasks</td>
<td>Unrealistic Timetable or wrong estimation of customer requirements</td>
<td>M</td>
<td>H</td>
<td>M</td>
<td>There are so many delays on prototype deliveries</td>
<td>Mitigate</td>
<td>Estimations shall be discussed, revised and changed at every sprint. For instance, delete unnecessary or unrealistic tasks.</td>
</tr>
</tbody>
</table>

**Notes:**
- **Probability:** Low (L), Medium (M), High (H)
- **Impact:** Low (L), Medium (M), High (H)
- **Exposure:** Low (L), Medium (M), High (H)
- **Trigger Events:**
  - x indicates a specific trigger event that may lead to the risk.
<table>
<thead>
<tr>
<th>03 Development Environment</th>
<th>Effort is greater than specified</th>
<th>L</th>
<th>M</th>
<th>L</th>
<th>developer/designer may complain about activity complexity</th>
<th>x</th>
<th>Discuss and agree the amount of effort during sprint planning.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lack of expertise may delay some deliverable tasks</td>
<td>M</td>
<td>M</td>
<td>M</td>
<td>developer/designer may not solve bugs or errors on time</td>
<td>x</td>
<td>Provide enough documentation, references, help and other resources.</td>
<td></td>
</tr>
<tr>
<td>Testing environment resources are unavailable</td>
<td>M</td>
<td>H</td>
<td>M</td>
<td>There was a failure in HW services</td>
<td>x</td>
<td>Make a ticket via help desk, and track the ticket</td>
<td></td>
</tr>
<tr>
<td>Delay on procurement process to obtain HW and SW resources</td>
<td>M</td>
<td>H</td>
<td>M</td>
<td>Resources haven't arrived at the expected date</td>
<td>x</td>
<td>Inform to the manager delays to scale the problem/solution</td>
<td></td>
</tr>
<tr>
<td>Supplier’s tool failures during prototyping activities</td>
<td>L</td>
<td>M</td>
<td>L</td>
<td>There are so many unexpected Tool failures that needs supplier’s support</td>
<td>x</td>
<td>Request for urgent supplier’s support.</td>
<td></td>
</tr>
<tr>
<td>Administrative delays may cause issues to purchasing licenses</td>
<td>M</td>
<td>M</td>
<td>M</td>
<td>Licences were not provided at the expected date</td>
<td>x</td>
<td>Inform to the manager delays to scale the problem/solution</td>
<td></td>
</tr>
<tr>
<td>Delays on procurement approvals may create obstacles in prototyping activities</td>
<td>M</td>
<td>L</td>
<td>H</td>
<td>Prototyping resources have delays to be assigned to developer</td>
<td>x</td>
<td>Inform to the manager delays to scale the problem/solution</td>
<td></td>
</tr>
<tr>
<td>05 Product /Costs</td>
<td>Low quality tools may produce a rise of costs during validation and verification activities</td>
<td>L</td>
<td>H</td>
<td>M</td>
<td>There are so many unexpected Tool failures that needs supplier’s support for longer period of time</td>
<td>x</td>
<td>Study case analysis shall be considered before deciding on a solution.</td>
</tr>
<tr>
<td>06</td>
<td>Safety Critical/Security</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>----</td>
<td>--------------------------</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tools licences expires before a prototype or the project was finished</td>
<td>L</td>
<td>M</td>
<td>L</td>
<td>Tool cannot run and shows licence expired message</td>
<td>x</td>
<td>Licence expiration time shall be considered at the project plan, and it shall be renewed every time that it is needed in advance</td>
<td></td>
</tr>
<tr>
<td>Extra cost may be raised due to complex or unrealistic requirements</td>
<td>L</td>
<td>H</td>
<td>M</td>
<td>Suppliers may require to charge extra costs due to requirement complexity</td>
<td>x</td>
<td>Eliminate complex requirements</td>
<td></td>
</tr>
<tr>
<td>Tool hardware requirements may raise cost due to lack of hardware infrastructure resources</td>
<td>L</td>
<td>L</td>
<td>L</td>
<td>Hardware or technical pre-requisites were not considered.</td>
<td>x</td>
<td>Consider properly and carefully software and hardware pre-requisites before taking a decision. Analyze whether are they satisfied or not.</td>
<td></td>
</tr>
<tr>
<td>Safety critical requirements were not properly defined, and product might not fulfil minimum safety critical properties</td>
<td>H</td>
<td>H</td>
<td>H</td>
<td>Safety critical requirements are unrealistic, or they don’t have enough specifications</td>
<td>x</td>
<td>Review and refine safety critical requirements at every sprint</td>
<td></td>
</tr>
<tr>
<td>Improper safety critical function definition or design at tool may lead to Unit control critical failures.</td>
<td>H</td>
<td>H</td>
<td>H</td>
<td>Safety critical requirements were not considered properly during design/prototyping phases</td>
<td>x</td>
<td>Run safety critical validation and verification tasks with higher priority</td>
<td></td>
</tr>
<tr>
<td>Inaccurate software design and development may lead into Malfunction of Unit Control component</td>
<td>H</td>
<td>H</td>
<td>H</td>
<td>System design and analysis were performed without close supervision, validation and</td>
<td>x</td>
<td>Provide enough mentoring, supervision and documentation during prototyping</td>
<td></td>
</tr>
<tr>
<td>Stakeholders/ Customer</td>
<td>Stakeholders may require for product re-design since he or she has found it unsatisfactory</td>
<td>M M M</td>
<td>There were a lack of customer orientation design</td>
<td>x</td>
<td>Run customer interface validation during prototyping</td>
<td></td>
<td></td>
</tr>
<tr>
<td>------------------------</td>
<td>-------------------------------------------------------------------------------------------------</td>
<td>-------</td>
<td>-------------------------------------------------</td>
<td>---</td>
<td>-------------------------------------------------</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Stakeholders and the customers introduces new requirements continuously during finishing project phase</td>
<td>L M L</td>
<td>There is a lack of previous approvals, and requirement validation</td>
<td>x</td>
<td>Negotiate and lead these requirements for further development.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Stakeholders/Customers may not collaborate during validation and verification task or requirement refining</td>
<td>L H M</td>
<td>Stakeholders or customer reject to attend to one or more related tasks</td>
<td>x</td>
<td>Inform to the manager delays to scale the problem/solution</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Lack of security policies at transport and protocol layers may lead to sniffing and password intrusion

Lack of control profiles/responsibilities and parameter validations may lead to change safety critical parameters or dependant parameters without any authorization.

A safety critical parameter does not contain enough validations during prototype testing.

Consider this requirement into standards specifications.

Consider this requirement with high priority during prototype development.

Consider this requirement with high priority during prototype development.

Stakeholders and the customer may require for product re-design since he or she has found it unsatisfactory.
<p>| 08 | Personnel/Resources | Stakeholders/Customer don’t understand how to use tools or selected tool is not user friendly enough. | L  | L  | The tool seems to be so complicated and less intuitive during validation and verification tasks | x | Run customer interface validation during prototyping |
| 08 | Personnel/ Resources | Lack of personnel expertise on programming languages | L  | M  | Personnel does not understand the code or other references | x | Provide enough mentoring and documentation during prototyping |
| 08 | Personnel/ Resources | Lack of personnel expertise on modeling tools | L  | M  | Personnel does not understand the code or other references | x | Provide enough mentoring and documentation during prototyping |
| 08 | Personnel/ Resources | Personnel unavailability | L  | L  | Someone may be sick or absent and he or she is required to run some tasks | x | Inform or call in advance |
| 08 | Personnel/ Resources | Lack of supplier support or collaboration | L  | H  | They don’t give effective and opportune answer to different inquires or support | x | Inform to the manager delays to scale the problem/solution |
| 09 | Software Design/ Technical | Unclear design specification may generate confusion and misunderstandings | L  | H  | The design is so complex and there are developer/designer complains | x | Provide granular design, enough mentoring and documentation during prototyping |</p>
<table>
<thead>
<tr>
<th>Issue Description</th>
<th>Severity</th>
<th>Description</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Platform or software components incompatibility</td>
<td>L</td>
<td>Component integration was not successful due to incompatibility messages or software pre-requisites were not considered properly.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>H</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>M</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>X</td>
<td>Consider properly and carefully software and hardware pre-requisites into the design</td>
<td></td>
</tr>
<tr>
<td>Non-functional requirements or safety properties were not considered during software design</td>
<td>L</td>
<td>Non-functional requirements were not considered during validation and verification tasks or there is a lack of traceability between software design and non functional requirements.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>H</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>M</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>X</td>
<td>Run safety critical and non-functional requirement validation and verification tasks</td>
<td></td>
</tr>
<tr>
<td>Prototype’s scope may be so wide and difficult to complete</td>
<td>H</td>
<td>Prototype scope is unrealistic and contain complex task to develop in order to deliver the product</td>
<td></td>
</tr>
<tr>
<td></td>
<td>H</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>H</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>X</td>
<td>Narrow prototype scope into a specific function or scenario to deliver as a representation to prove a concept. Besides, build model and run a simulation based on a model to run a validation and verification to prove the concept previously</td>
<td></td>
</tr>
</tbody>
</table>
## Risk probability/Impact - Matrix Relationship

<table>
<thead>
<tr>
<th>Probability</th>
<th>Impact</th>
<th>Action</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>HIGH</td>
<td>HIGH</td>
<td>Monitor Probability Exposure: High</td>
<td>Reformulate Risk Plan Exposure: High</td>
</tr>
<tr>
<td>MEDIUM</td>
<td>HIGH</td>
<td>Monitor Probability Exposure: High</td>
<td>Reformulate Risk Plan Exposure: Medium</td>
</tr>
<tr>
<td>LOW</td>
<td>LOW</td>
<td>Ignore but log Probability Exposure: Low</td>
<td>Monitor Probability Exposure: low</td>
</tr>
<tr>
<td></td>
<td>MEDIUM</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>HIGH</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
APPENDIX B - SOFTWARE ARCHITECTURE DOCUMENT

Parameter Handler
Initial Software Architecture
2 General System Description

This project is composed some solutions that implement several system styles. One solution implements a client server application to manage parameter calibration. This also contains a calibration tool that will be accessed by any client via function call, and this is responsible of handling different parameter configuration levels and administrates ASAM standards. Other solution is based on real time systems event handling that is composed of a layered architecture that describes different features from XCP protocol and Vector’s specifications to be implemented. [6, 3]

In this sense, a deployment diagram was designed to represent different physical and software components to be deployed at this project (see figure 1). This diagram describes three subsystems that are Calibration Management System (client server component), CANape, and DCU2 component. This last component is composed of several layers such as application, protocol, transport and interface. Hence, a component diagram is shown in figure 2 that implement previous organization into a master /slave philosophy required by XCP standards. Therefore, there exist a set of ports stated in this diagrams that represent different access between one layer to other one and the usage of each layer. A master component is composed of eAEE clients (standard client, administrator and configurator client) and CANape tool. The slave component is represented by XCP/DCU2 subsystem and its layers.

In addition, a relationship between different requirements stated in requirement document (see Appendix B.2)and each component from component diagram is described at the table below:

<table>
<thead>
<tr>
<th>Requirement IDs</th>
<th>Software component name</th>
<th>Software Component ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>FR1, FR5, FR9, NF6, NF5, NFR7,</td>
<td>MasterComponent</td>
<td>SC1</td>
</tr>
</tbody>
</table>
3 Use Cases organization per prototype

In order to develop and deliver every prototype, it is needed select a finished project, it is important to select a set of use cases and scenarios to narrow prototype functionalities and development tasks. In this sense, an initial prototype shall be developed based on a model to validate concepts and design. Then, a low level initial prototype will prove the possibility to change parameter values in running application process. This is followed by the packages XCP over Ethernet functionality at the DCU to check XCP feasibility. Finally, client server prototype and calibration tool prototypes will be built to prove the overall concept between calibration tools and DCU. [3] A relationship between each prototype, use case short description, component ID and requirement ID from appendix B.2 is stated below:

<table>
<thead>
<tr>
<th>Requirement IDs</th>
<th>Software Component ID</th>
<th>Prototype ID</th>
<th>Use Case ID and diagram number</th>
<th>Short Description/remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>FR1, FR5, FR9, NF6, NF5, NFR7, NFR8, FR10, FR2</td>
<td>SC1, SC2, SC3, SC4,</td>
<td>3.4</td>
<td>See figure 3 and 4</td>
<td>UC7: Create Calibration Project: this use case represents creating and configuring projects in eASEE system and CANape. UC7.1: Configure product attributes: this allows...</td>
</tr>
</tbody>
</table>
configuring products attributes for a software key related to the calibration project.

UC8: Modify Calibration Project: this allows to modify the calibration project properties

UC9: Create Dataset: this represents a container for holding parameter files that corresponds to a set of calibration data. This may be organized by function view or group view.

UC10: Set up calibration ownerships: this permits configuring settings for assigning tasks to calibration engineers and other stakeholders.

UC11: Handle Historical logs: this allows seeing an historical log about project changes, calibration data, parameter sets, and other related information.

UC12: Handle Reports: this is able to generate reports according to the calibration tasks.

UC13: Handle baselines: this creates a baseline for calibration projects.

UC14: Deploy data sets and parameter: this generates the deployable version for a dataset in case of eASEE. For CANape, this case generates a deployable parameter file to be integrated at the deployable software product.

UC15: Handle profiles: this allows configuring roles and responsibilities at the eASEE administrator tool.

UC16: Deliver parameter sets: this allows delivering parameter sets into the parameter file at the eASEE.

UC17: Login: this allows to eASEE users to logging into the system and this includes Authentication Methods.

UC17.1: Authentication Methods: this allows defining authentication method kind to different users at the eASEE administrator tool.

UC18: Handle Calibration data: this use case allows performing basic calibration configuration, online and offline calibration and measurement.

UC19: ASAP2 Database configuration: this permits set up transport layer and devices configuration into the A2L file descriptions. This will enables the physical connection between master and slave under XCP over Ethernet.

UC20: Generate A2L: this enables the generation
of the A2L file by extracting and handling object files from the software after compilation process in order to extract parameters information.

<table>
<thead>
<tr>
<th>SC9</th>
<th>1,2</th>
<th>Figure 5, UC1-</th>
</tr>
</thead>
<tbody>
<tr>
<td>SC10</td>
<td>1,2</td>
<td>Figure 5, UC2, UC3</td>
</tr>
<tr>
<td>SC11</td>
<td>1,2</td>
<td>Figure 5, UC3, UC2, UC1</td>
</tr>
<tr>
<td>SC12</td>
<td>1,2</td>
<td>Figure 5, UC3, UC2, UC1</td>
</tr>
<tr>
<td>SC13</td>
<td>3,4</td>
<td>See figure 5</td>
</tr>
</tbody>
</table>

**UC1: XCP Configuration**, a set of initial definitions and base configuration has to be set in the DCU2 component in conjunction to commands configuration in order to get the necessary base XCP configuration to execute different XCP layers.

**UC2: Handle Master/Slave connection**, different task has to be defined at the DCU2 to be able to manage master-slave connection over Ethernet. Some of main tasks that shall be executed are: setting up a session, synchronize tasks, and initialization. They represent associated used cases respectfully.

**UC3: Memory Page**, this implements different memory page management functions to control application layer every time that an XCP command is executed to request different associated use cases such as get page, set page, copy page and handle pointers.

**UC3.1: Get Page**, this use case is responsible of getting the memory page required by a XCP command and it includes check memory status use case (UC3.4).

**UC3.2: Set Page**, this use case is responsible of setting the memory page required by a XCP command and it includes check memory status use case (UC3.4)

**UC3.3: Handle pointer**, this use case is responsible of managing memory pointers when a memory page is get or set and it includes check memory status use case (UC3.4)

**UC3.4: Check memory status**, this use case is responsible of checking memory availability and access wherever any command or resource request a memory page. Therefore, it implements mutex property described in section 6.

**UC3.5: Handle files**, this use case is responsible of managing files in order to create, read and write parameter files. This includes check memory status use case (UC3.4)

This represents use cases UC20 and UC19
NF1, NF2
FR4, FR11, FR11.1, FR11.4, FR11.5, NF6, NFR8, NF3, NF4, NF1, NF2
SC14 2,3 Figure 5, UC6

Furthermore, this document represents detailed information about prototype 1. Therefore, different models were designed based on state machines, Simulink and TDL to run model based validation and verification. In this sense, this section is complemented in sections 6, 7, 8, 9 and 10. They present a short model explanation, result discussion and further work to be done from prototype 2 to 4 in order to refine these models.

4 Initial Use Case description
See Appendix B.1. This will be refined in that section during project development.

5 Initial Communication Sequence Diagram according to use cases
According to previous sections and ASAM specifications [7], a set of communication diagram between MasterComponent and Slave component was defined. This also complements use case descriptions. Thus, the following table shows a relationship between each use case (according to the ones defined in section 2 and sequence diagram.

<table>
<thead>
<tr>
<th>Use Case IDs</th>
<th>Communication Diagram and figure No.</th>
<th>Communication Sequence ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>UC6</td>
<td>XCP Standard Communication Sequence, figure 6</td>
<td>COMS1</td>
</tr>
<tr>
<td>UC2, UC1</td>
<td>Setting up a Session communication sequence, figure 7</td>
<td>COMS2</td>
</tr>
<tr>
<td>UC3</td>
<td>Get Memory Pages communication sequence, figure 8</td>
<td>COMS3</td>
</tr>
<tr>
<td>UC3.1, UC3.2, UC3.3, UC3.4</td>
<td>Read/Write DCU2 Parameters Communication Sequence, figure 9</td>
<td>COMS4</td>
</tr>
</tbody>
</table>

6 Memory Page Constrains
In order to follow ASAM specifications with respect to handling slave memory at the DCU2 it is important to define a set of constrains and a concepts in order to implement memory page handling philosophy properly. [7] Thus, a conceptual diagram is stated in figure 10 where it describes following features:

- A slave memory is composed of a Logical Layout and a Physical Layout that requires an initialization.
- A logical layout describes a memory segment that may contain one or more memory pages.
The physical layout describes sectors that contain a limit and a size.
One or more segments may have one or more sectors. However, sectors usages apply for flash programming and other features.
A memory page is composed of page number, address, name, size, access flag ID, and mirrored segments offsets (if it is applicable).
A memory page requires a XCP access and DCU2 access validation in order to check whether this page is available or not. This assures that two or more resources from either XCP protocol or DCU2 cannot access to the same memory page at the same time (mutex property).

7 State Machines Description

As part of prototype 1, some state machines were designed in order to specify different constrains and states that XCP protocol shall have. This specifies the precedence relations between one state to another state as well. In particular, each state represents a task to be executed by the operative system in a given period of time. In this sense, some state machines were designed. One is a general state machine for representing XCP protocol behaviour and another one is a specific state machine that represents memory page access based on mutex protocol. Thus, they represent a reactive behaviour in a given environment handled by commands that came from calibration tasks and XCP configurations.[7] A description from each state and its relationship between use cases and requirements is described below:

7.1 XCP Protocol State Machine

As it shown in figure 12, this state machine describes a set of modes, states and guards to be fulfilled to go from one task to other during protocol execution.
From start, state XCP protocol may go from resume mode to other mode. When resume mode is true, it goes to transferDTO otherwise it goes to disconnected.
If it is connected, it can execute any command and remain into connected state if only if a command send a response that may have a error level less greater or equal to 2. Else, it goes into a disconnected state.
If other mode that is different from resume mode is true, then it will remain connected.
If other mode that is different from resume mode is false, then it will remain disconnected.

7.2 Memory Page Access State Machine

As it shown in figure 13, this state machine describes a set of modes, states and guards to be fulfilled to go from one task to another during memory page checking by applying mutex protocol.
From start state it can state in any mode until a memory request has been sent from XCP protocol or another common software application at the DCU2.
Once a memory request has been sent, it goes to waiting stated called commandWait for a command to be executed.
If the memory is accessible and available a task can get the requested memory page resource. Otherwise, it will remain in waiting state until another memory page or that memory page is available and accessible.
Once the command has finished using the memory page resource, it can go to other mode and the start state to process a new request.
8 TDL/Simulink model description

In order to model the timing behaviour in embedded systems, a TDL/Simulink (Time definition language based on Matlab/Simulink) model has been designed and simulated in order to represent XCP protocol and memory page handling behaviour. In particular, this model has been built in TDLVisualCreator and Simulink via library importing (see sample working environment at figure 26). However, there are many task that are needed to be modelled and they are defined in a separated c code (for instance, XCP protocol layer API) which are not possible to describe them at the Simulink environment, so they have been created as called “dummy functions” as it is shown in figure 20. Therefore, a base module that is composed of several dummy functions has been included in the model in order to be able to simulate an initial behaviour. In addition, different tasks, modes, guards and modules were defined according to state machines described previously. A pre-condition to this a polling method has been implemented in order to simulate a time-triggered system based on the given specifications at the TDL model stated in section 16.9, and this is a textual model that was generated automatically by TDLVisualCreator tool. Thus, the following list states a description per each model, their components and the relationship between each component, use case and requirement:

8.1 XCP Protocol base module

- As it stated in figures 14, this module is connected to a TDL model that describes different tasks to be executed by following XCP protocol state machine conditions. Consequently a polling method configuration was set in the model in order to make a sample simulation based on a period of 10 ms.
- A sensor from the XCP protocol represents when a command is executed that may included connected and disconnected states.
- According to the XCP protocol state machine, task to be executed are: Connect, disconnect, transferDTO, response and executeCMD.
- An actuator for each task has been defined in order to enable a triggering event when the task is requested by the actuator. This configuration will be defined at Mitrac tools by the moment that Logical Execution time is defined.
- Modes to be handled by the XCP protocol are resume and other mode. Thus, corresponding guards will control mode switching in a given period of time.

8.2 Memory Page Base Module

- As it described in figures 15, this module is connected to a TDL model that states different tasks to be executed by following memory page state machine conditions. Consequently a polling method configuration was set in the model in order to make a sample simulation based on a period of 10 ms.
- A sensor from the memory page management represents when a memory requirement is made by a DCU2 resource or XCP protocol commands.
- With respect to memory page access state machine and set of use cases UC3, task to be executed are: reqMem, locateMem, calcPointer, setPage, mutex function and checksum. They corresponds to the states request memory, locate memory, calculate pointers, set pages and verify mutex property to check memory page availability and checksum function to check memory´s accessibility. [7] Therefore, a set of proposed algorithms has been modelled in Simulink for understanding memory location and pointers management. See figures 16 and 17.
- An actuator for each task has been defined in order to enable a triggering event
when the task is requested by the actuator. This configuration will be defined at Mitrac tools by the moment that Logical Execution time is defined.

- Modes to be handled by the memory module are idle and calibrate. Thus, corresponding guards will control mode switching in a given period of time. Some sample function that can be executed to switch one mode to other is given by figures 18 and 19 that represent Simulink models as well.

9 Results discussion of State Machines simulations

According to the executed simulations and model checking tasks that were made at UPPAAL tool, a set of results are presented and discussed in this section. In this sense, the XCP protocol state machine is able to go to all requested states without having a deadlock state. This is shown in figures 21 and 22. Besides, due to the nature of the XCP protocol, it will process several commands at the same time so this state machine has more than one instance at the simulation to check when all different command types might be processed. However, this state machine is not able to control different resources and managing different command request and responses in a synchronous mode. Thus, a polling method shall be implemented initially to run periodical tasks in a given period of time. Hence, it is also needed to implement queue methods and other algorithms to improve performance during measurement tasks. In particular, it is required to handle a list with the different elements and data to be calibrated and measured through XCP protocol according to ASAM specifications [7].

With respect to memory page access state machine, it implements properly the mutex property in order to satisfy safety requirements (NFR4, NFR4.1, NFR4.2, and NFR4.3) that imply that a DCU2 or XCP resource cannot access to the same memory page at the same time. It also is deadlock free and the memory page can be accessed if only if it is available. [5] This can be detailed at figures 23 and 24.

In conclusion, both state machines have served as input for designing TDL/Simulink models required to design logical execution time management by the application layer at the DCU2 component. They also satisfy different safety critical requirements.

10 Results discussion of TDL/Simulink simulations

As it has been explained previously, simulations were executed via Matlab/Simulink environment to check the logical time execution behaviour in a given period of sample time. A time trigger event was configured based on 10 ms as a sample time for simulating polling methods in both XCP protocol and memory page module. An important result from this simulation is that given task may be managed in that period of time and they can be synchronously executed according to the specifications that may be configured at Mitrac Tools. This is described in figures 25 and 26. However, it is important to state a scheduler to handle different time periods according to what function needs. Thus, it is also needed to improve performance during measurement task by using event triggering policies to enable required interrupts when different events are requested. This is also suggested by ASAM specifications at the corresponding sections. [7]

Since dummies functions where defined to simulate behaviour in different tasks that are C code based, it is important to validate Mitrac configuration in real time because it will deliver some results that are closer to the reality according to C functions that will be defined for each task purpose. Thus, it is also useful to run an upcoming configuration that includes XCP behaviour based on Vector’s plugin that was installed in Matlab/Simulink.

In summary, different modules can execute the different set of tasks in a given period...
of time successfully by applying polling methods. Nevertheless, it is important to improve the performance for measurement task by implementing event triggering configurations.

11 Class Diagram and Libraries organization

According to previous schemas and model descriptions, a general class diagram was defined in order to represent different c files and h files organization with respect to the software development. In this sense, a package may represent either a library or a layer. In particular, there is a general class diagram organization that includes all layers and their main c files as it is shown in figure 11. Then, a specific class diagram per layer is also described in this report, so application layer was explained in figure 11.1 which belongs to prototype 2 (Memory Page handler prototype) and represents code skeleton for this layer. In addition, the software structure and organization for prototype 3 is described in figure 11.2. In addition, a full XCP protocol implementation and integration with transport, application and interface layer is described in figure 11.2.

11.1 XCP Protocol Integration by function call for Online Calibration

Function call process for online calibration is state din figure 6.1 that represents a sequence diagram. This states the order of function calls, source and destination layer. In this sense, in order to enable online calibration it is needed to get the described layers interaction by the following functions:

- XcpInit: this will initialize the XCP protocol layer from the application layer into the init task.
- AppXcpGetPointer: this function will transform XCP protocol pointer type into standard C code pointers at the application layer.
- XcpCommand: this will call the command processor from the transport layer to the protocol layer.
- XcpSendCallBack: this is used for handling the sending ack messages between protocol and transport layer.

11.2 XCP Protocol Integration by function call for Online Measurement

Likewise to online calibration, it is required to enable the same functions in conjunction to functions for enabling handling events and checksum functions. In this sense, figure 6.2 states the required additional functions for these purposes and are listed below:

- XcpBackground: enables checksum functions at init task at the application layer.
- XcpEvent: this function handles the events for measurement and DTQ lists.
- AppXcpSend: this is responsible of sending measurement information according to the events transmitted at the DTQ lists between protocol layer and the transport layer.

12 XCP-Train Device Integration

In order to integrate XCP application layer with train devices, a preliminary solution has been designed based on previous point's discussion. Thus, an initial schema about this integration has been described in figure 2.1. This states that it is possible to get parameter values via A2L files. This set of files is handled by the interface layer that can process object/Motorola files from A2L in both modes: online and offline. Then, the transport layer will manage object/Motorola files in order to send and request some needed information to protocol layer. Therefore, the application layer will handle this information by function call to
protocol layer and POU. Hence, any device can get parameter values from A2L information that is at CANape. Besides, there exists offline mode and online mode. In online mode, A2L file information is managed by the interface layer as it was previously described. In case of offline mode, it is possible to generate C code from some templates to handle parameter by function call. Then, this code is added as a resource and downloaded by MTVD tool. It is important to clarify that this step shall be executed once that calibration tasks have been completed.

However, this solution represents a preliminary proposal since there are some inquires that were clarified about Interface layer and a formal description will be presented during release 4:

• There is no need to generate a parser to handle map and object files since CANape database editor provides the required mechanism to extract parameter information. Therefore, Vector has generated a patch for enabling database editor to read VxWorks object files according to GNU compiler specifications.

• It was defined that offline mode is required for the first time to gather parameter information from object files in order to generate A2L file. Then, this process is also required to be performed to deploy parameters into product C code once calibration and measurement task were completed and certified.

• For deploying parameters into C code, there exist a set of templates provided by Vector that the Tool engine uses for generating deployable files.

• Once that the file was deployed, it is needed to be compiled and merged into the product code and downloaded into the DCU2. Therefore, parameters will be assigned by function call by other devices or application functions.

• There is one limitation with respect to this approach, it is required to make changes into software organization, and parameter identifiers since the database editor do not support type definition structures for a set of parameters. In addition, the database editor pre-requisite states that parameters shall be defined as a global variable at the first time in order to get the complete symbol table description according to GNU compiler. When parameters will be deployed, they will get static identifier into a header file in order to allow the inclusion of this file at the main c code and the function call process.

13 Parameter classification constrains according to eASEE Methodology.

• As it is stated in figure 10.1 there are a set of constrains and rules to be followed in order to fulfil eASEE methodology pre-requisites into a calibration project development.

• A calibration project requires one software key. This software key is composed of a product key, product attributes and a variant.

• A variant requires a parameter dataset that is composed of a calibration set file.

• There different kind of parameter datasets. One is the standard dataset that contains the whole controller description. Other one is integrated dataset that is composed of partitioned datasets that will contain a fraction of software and parameter from the complete calibration project.

• There are different kinds of views to organize calibration sets into datasets. One the standard view that is generated by default if other options are not defined. Another is function based view or parameter group view.

• Function based view is defined by ASAP2 definitions at the A2L file.
Group based view is user defined view by dragging and dropping parameters from ASAP2 specifications.

14 Design Decisions

A set of decision has been made about selecting the most convenient option to solve different design issues or questions that represent a significant impact into software development. Thus, solution selection was based on criteria-concern analysis where the concern corresponds to some questions and concerns. The following sub-section describes the rationale from each option and the corresponding concern:

Concern: Which method shall be used for checking memory and resource availability?

Criteria: Functions developed in operative system, complexity, time effort.

Mutex function for RAM and Checksum services for persistent memory

Rationale:

- Functions developed in operative system: there exist all the needed functions for handling mutex semaphore in RAM and checksum services for handling file security.
- Complexity: code development is not so complex since functions are already defined at the API, so it is only needed to design a flow for calling them into the code.
- Time effort: this provides code reusability since it implements already defined functions and reduces time effort from developer.

Decision: this option has been selected because functions are already defined and tested in the API. Thus, this decreases time effort from developer.

Checksum services for persistent memory and RAM

Rationale:

- Functions developed in operative system: there exist all the needed functions for handling checksum services for handling file security. However, there are not functions defined at the operative system for handling RAM.
- Complexity: code development may be complex because there are not functions for handling RAM memory.
- Time effort: the developer have to spend significant time in defining and developing functions for handling checksum services in RAM.

Decision: this option has been rejected because the developer may need to make significant effort on implementing functions for handling memory in RAM.

Concern: Which persistent memory must be used for storing parameters when DCU2 is off or disconnected?

Criteria: security, maximum memory storage capacity, back up resources, memory space availability, and memory overload risk

File System

Rationale:

- Security: unexpected power loss may corrupt file information. Then, checksum
control must be implemented.
• Maximum memory storage capacity: it has limit of 13MB.
• Back up resources: it has an extension for back up of 32MB.
• Memory space availability: there is availability for parameter storage up to 3MB, and it contains sufficient space for storing big amount of parameters because it is possible to handle around 5000 parameters that may have a size of 2MB in total. (based on an assumptions and address size specifications from ASAM standards [7])
• Memory overload risk: there is a riskless memory overload since there is sufficient space for storing parameters into an application.

**Decision**: this option has been selected because it offers a riskless memory overload, back up resources and enough memory space capacity in a worst case scenario. However, it is necessary to consider parameter amount and size as requirement for long scale projects when DCU2 software from common application is designed.

**Non-Volatile RAM (NVRAM)**

**Rationale:**
• Security: if there were an unexpected power loss, stored data at the memory would be corrupted. Then, checksum control must be implemented.
• Maximum memory storage capacity: it has limit of 1MB.
• Back up resources: there are not resources for back up.
• Memory space availability: there is a limited space of 500KB
• Memory overload risk: there is a high risk of memory overload in projects where huge amount of parameters may need to be configured.
• Decision: this option has been rejected because it does not offers a riskless memory overload, back up resources and enough memory space capacity in a worst case scenario.
• Concern: Which file format shall be used for handling parameter file stated in section 13.2 this is for storing memory page information into a persistent memory?

**Criteria: file size, implementation complexity**

**Binary File**

**Rationale:**
• File size: this file format has a small size.
• Implementation complexity: it has a low complexity since it is possible to handle any data type by using file pointers to memory segment structures. Then, is possible to handle this at POU’s.

**Decision**: this option has been selected because it offers small size for saving memory in flash and its implementation is not so difficult.

**Hex File**

**Rationale:**
• File size: this file format has a bigger size than a binary file.
• Implementation complexity: it has a high complexity since it is needed to cast file pointers and requires an extra effort to handle this format at POU.
Decision: this option has been rejected because it may be difficult to be implemented and requires a bigger space for storing the file than binary format.

15 Future Work

- It is important to implement seed and key mechanism to enforce security at XCP prototype.
- It is necessary to refine measurement process in order to improve performance.
- It will be necessary to modify and change current train application software organization and variable definitions in order to make them as global to enable XCP access in the memory.
- It is recommended to certify and implement this solution with a small project, then having a transition phase or pilot program among application engineers where they can test and play with this philosophy. After this, it will be necessary to apply further improvements according to this feedback. Finally, it will be possible to run a full implementation and certification for a complete train application.
- In order to guarantee a successful implementation in more complex project, it is required to handle training programs and workshops with the supplier in order to adapt and evolve this philosophy in early stages for small projects and large projects.
16 Diagram and Model list

16.1 Deployment Diagram

Figure 1: deployment Diagram
16.2 Component Diagram

Figure 2: Component Diagram
Figure 2.1: XCP-Train Device Integration workflow process
16.3 General Use Cases

Figure 3: General Use Case Diagram
16.4 Initial Use cases

16.4.1 Calibration Tool Use Case Diagram

![Calibration Tool Use Case Diagram](image)

Figure 4: Calibration Tool Use Case Diagram

16.4.2 XCP/DCU2 Use Case Diagram

![XCP/DCU2 Use Case Diagram](image)

Figure 5: XCP/DCU2 Use Case Diagram
16.5 Initial Communication Sequences

16.5.1 XCP Standard Communication

Figure 6: XCP Standard Communication Sequence
Figure 6.1: XCP Layer integration by function call for Online Calibration

Figure 6.2: XCP Layer integration by function call for Online Measurement
16.5.2 Setting up a Session

Figure 7: Setting up a Session communication sequence
16.5.3 Get Memory Pages

Figure 8: Get Memory Pages communication sequence
16.5.4 Write/Read DCU2 Parameters

Figure 9: Read/Write DCU2 Parameters Communication Sequence
16.6 Conceptual diagrams and constraints

Figure 10: Memory Page Conceptual Diagram and Constrains
16.7 XCP/DCU2 Class/Package Diagrams
Figure 11.1: Application Layer Class Diagram for prototype 2

Figure 11.2: XCP/DCU2 General Class/Package Diagram for prototype 3
16.8 State Machines

16.8.1 XCP Protocol State Machine
16.9  TDL Textual Models

module MemoryModule {

    public const characteristic = 10;
    public const addr = 40;

    public sensor int req uses getReq;

    actuator int aReq uses setAReq;
    actuator byte aLoc := addr uses setALoc;
    actuator byte aPointer := characteristic uses setAPointer;
    actuator byte aSet := addr uses setASet;
    actuator int aMutex uses setAMutex;
    actuator byte aSum := addr uses setASum;

    public task reqMem {
        output
        int o;
        uses treqMem(o);
    }

    public task locateMem {
        output
        byte o := addr;
        uses tlocateMem(o);
    }

    public task calcPointer {
        output
        byte o := characteristic;
        uses tcalcPoint(o);
    }

    public task setPage {

output
  byte o := addrs;
  uses tsetPage(o);
}

public task mutex {
  output
  int o;
  uses tmutex(o);
}

public task checkSum {
  output
  byte o := addrs;
  uses tcheckSum(o);
}

start mode ide [period=10 ms] {
  task
    [freq=1, slots=1*] reqMem {}
    [freq=1, slots=1*] locateMem {}
    [freq=1, slots=1*] calcPointer {}
    [freq=1, slots=1*] setPage {}
    [freq=1, slots=1*] mutex {}
    [freq=1, slots=1*] checkSum {}
  actuator
    [freq=1, slots=1*] aReq := reqMem.o;
    [freq=1, slots=1*] aLoc := locateMem.o;
    [freq=1, slots=1*] aPointer := calcPointer.o;
    [freq=1, slots=1*] aSet := setPage.o;
    [freq=1, slots=1*] aMutex := mutex.o;
    [freq=1, slots=1*] aSum := checkSum.o;
  mode
    [freq=1, slots=1*] if ide2calibrateGuard(req) then calibrate {reqMem.o := reqMem.o;locateMem.o := locateMem.o;calcPointer.o := calcPointer.o;setPage.o := setPage.o;mutex.o := mutex.o;checkSum.o := checkSum.o;}
}

mode calibrate [period=10 ms] {
  task
    [freq=1, slots=1*] reqMem {}
    [freq=1, slots=1*] locateMem {}
    [freq=1, slots=1*] calcPointer {}
    [freq=1, slots=1*] setPage {}
    [freq=1, slots=1*] mutex {}
    [freq=1, slots=1*] checkSum {}
  actuator
    [freq=1, slots=1*] aReq := reqMem.o;
    [freq=1, slots=1*] aLoc := locateMem.o;
    [freq=1, slots=1*] aPointer := calcPointer.o;
    [freq=1, slots=1*] aSet := setPage.o;
16.10 Simulink Models

16.10.1 XCP protocol Base Module

Figure 14: XCP Protocol Base Module

16.10.2 Memory Page Base Module

Figure 15: Memory Page Base Module
16.10.3 Calculate Memory Pointer Algorithm Based on Simulink model

Figure 16: Calculate Memory Pointer Algorithm
16.10.4 Locate Memory Segment Algorithm Based on Simulink model

![Diagram of Locate Memory Segment Algorithm]

Figure 17: Locate Memory Segment Algorithm

16.10.5 Sample Guard functions for switching ide to calibrate modes at memory page module

![Diagram of Ide to Calibrate Guard Function]

Figure 18: Ide to Calibrate Guard Function

![Diagram of Calibrate to ide Guard Function]

Figure 19: Calibrate to ide Guard Function
16.10.6 Sample dummy function used at the base modules

Figure 20: Dummy Function based on checksum function

17 Simulation Results Graphs

17.1 State Machine Results

17.1.1 XCP/DCU2 State Machine Simulation

Figure 21: XCP/DCU2 state machine simulation
17.1.2 XCP/DCU2 State Machine Model Checking Results

Figure 22: XCP/DCU2 Model Checking

17.1.3 Memory Page Access State Machine Simulation

Figure 23: memory page access state machine simulation
17.1.4 Memory Page Access State Model Checking Results

Figure 24: Memory page access model checking
17.2  
**TDL/Simulink Simulation Results**

17.2.1  
**XCP protocol Base Module Simulation**

![XCP protocol base simulation](image1)

Figure 25: XCP protocol base simulation

17.2.2  
**Memory Page Base Module Simulation**

![Memory Page Base Simulation](image2)

Figure 26: Memory Page Base Simulation
Appendix B.1- Use Case Descriptions

Use-Case: <Memory Page Management>

Use Case ID: UC3

Brief Description:

This implements different memory page management functions to control application layer every time that an XCP command is executed to request different associated use cases such as get page, set page, copy page and handle pointers.

Actors:

DCU2

Preconditions

NA

Basic Flow of Events

1. The use case begins when DCU2 is in resume mode (idle) and connect command is re-quested from CANape tool.
2. DCU2 goes to other mode and the application layer must check memory availability and access. This includes check memory status use case
3. The application layer request for memory allocation in RAM and generates memory segments structure.
4. Memory segments are initialized and memory address is assigned.
5. When DCU2 goes back to resume mode, disconnect command is processed and there is a waiting time of 5 seconds (this time should be refined later).
6. When this time is expired, the allocated memory information is copied into a file in the file system and the memory is free in order to provide a new space for next event.
7. The use case ends.

Alternative Flows

1.1 If in step 1 DCU2 is different from resume mode, then it must remain in disconnected state.

1.2 If in step 2 the memory is not accessible and available, then an error message with high sever-it must be sent and DCU2 goes to disconnected state.

1.3 If in step 3 memory request for allocation and there is no enough space in DCU2 ram, then an error message with high severity must be sent DCU2 goes to disconnected state.

1.4 If there is an unexpected power loss in the DCU2 during file storage, then
checksum services must be enabled in order to check whether the file is corrupted or not.

Post-conditions
1.1 Parameter files shall be able to be modified or deleted if it is requested.

Use-Case: <Check Memory Status Description>
Use Case ID: UC3.4
Brief Description:
This use case is responsible of checking memory availability and access wherever any command or resource request a memory page.
Actors:
DCU2

Preconditions
1. Checksum file shall be created at the same directory of parameter file, and it must have the extension *.crc with the respective checksum.
2. Checksum´s filename shall be the same as parameter file name.
3. Checksum buffer type must be of type unsigned long.

Basic Flow of Events
1. The application layer request checking RAM memory availability.
2. The application layer calls mutual exclusion semaphore function from the operative system.
3. The function allocates and initializes the mutual exclusion algorithm.
4. The application layer checks file system accessibility via checksum service call from operative system.
5. When the function is called, it verifies the file name against checksum file
6. When the checking procedure has finished, the file should be available to be accessed as it is requested by the DCUTerm application.
7. The use case ends.

Alternative Flows
1.1 If in step 3 DCU2 is out of memory, or an invalid mutex option has been specified in the function, then it returns an error message or NULL pointer.
1.2 If in step 5 the parameter file does not match with the checksum file, then it returns 0xFFFF as address.

Post-conditions
NA.
## Appendix B.2-Requirements

<table>
<thead>
<tr>
<th>REQ ID.</th>
<th>Type</th>
<th>Subsystem</th>
<th>Description</th>
<th>Priority</th>
</tr>
</thead>
<tbody>
<tr>
<td>HR1</td>
<td>Hardware Requirement</td>
<td>XCP/DCU2</td>
<td>XCP-DCU2 system must be composed of a DCU2 hardware unit and Ethernet physical interface that allow communication between PC/DCU2. This DCU2 unit serves a sample to build low level prototypes.</td>
<td>High</td>
</tr>
<tr>
<td>HR2</td>
<td>Hardware Requirement</td>
<td>Client Server</td>
<td>In order to build a client server prototype it is required to have a dedicated computer that will contain virtual servers that may serves as database, application server and Repository server.</td>
<td>High</td>
</tr>
<tr>
<td>HR3</td>
<td>Hardware Requirement</td>
<td>Client Server</td>
<td>Client hardware should fulfil minimum requirement with respect to processor frequency, screen resolution, memory and hard disks. Some requirements are described according to Vector specifications. (see More details at requirement source reference)</td>
<td>low</td>
</tr>
<tr>
<td>HR4</td>
<td>Hardware Requirement</td>
<td>Client Server</td>
<td>In case of physical server implementation, a client workload analysis should be described in order to fulfil minimum performance requirements. A relationship between client workload and hardware parts is stated below according to Vector specifications. (see More details at requirement source reference)</td>
<td>low</td>
</tr>
<tr>
<td>HR5</td>
<td>Hardware Requirement</td>
<td>Client Server</td>
<td>Similar to HR4, database and repository server hardware requirements should to be defined according to Vector’s specification and number of clients analysis.</td>
<td>low</td>
</tr>
<tr>
<td>HR6</td>
<td>Hardware Requirement</td>
<td>Client Server</td>
<td>According to Vector specifications, network requirements should fulfil some expected band width and latency time</td>
<td>low</td>
</tr>
<tr>
<td>SPR1</td>
<td>Software Pre-Requisite</td>
<td>Client Server</td>
<td>Operative System in client computer shall need to get installed any version of Microsoft window from version 2000 up to now, .net framework, Adobe SVG viewer, crystal report and other additional features described at vector specifications.(see More details at requirement source reference)</td>
<td>Medium</td>
</tr>
<tr>
<td>SPR2</td>
<td>Software Pre-Requisite</td>
<td>Client Server</td>
<td>Administrator and configuration management clients will need similar software pre-requisites as SPR1, Oracle client and Oracle JDBC compliant with Oracle client according to vector specifications. (see more details at requirement source reference)</td>
<td>Medium</td>
</tr>
<tr>
<td>SPR3</td>
<td>Software Pre-Requisite</td>
<td>Client Server</td>
<td>With respect to application server software pre-requisites, it will depend on operative system and application server product. Possible combinations are stated at Vector specifications</td>
<td>Medium</td>
</tr>
<tr>
<td>SPR4</td>
<td>Software Pre-Requisite</td>
<td>Client Server</td>
<td>Database pre requisite will contain different combination between server operative system and database manager version according to vector specifications</td>
<td>Medium</td>
</tr>
<tr>
<td>SPR5</td>
<td>Software Pre-Requisite</td>
<td>Client Server</td>
<td>Repository server will also have similar possible software pre requisite combinations as stated in SPR3, SPR4. See vector specifications.</td>
<td>Medium</td>
</tr>
<tr>
<td>FR1</td>
<td>Functional Requirement</td>
<td>All</td>
<td>It is needed to develop a client server system that provides a required framework to develop calibration tasks in a distributed team. This system is composed of three subsystems Desktop application, Application Server and Database server</td>
<td>High</td>
</tr>
<tr>
<td>FR2</td>
<td>Functional Requirement</td>
<td>All</td>
<td>It is needed to develop an interface that permits the communication between Calibration tool and DCU2 controller. This also represents developing an internal DCU2 application interface that will handle different ASAM standards. This is described as XCP –DCU2 Implementation System.</td>
<td>High</td>
</tr>
<tr>
<td>FR3</td>
<td>Functional Requirement</td>
<td>All</td>
<td>It is required to implement ASAM standards such as ASAP2 in order to exchange A2L files which contains parameter information in client server system or application interface at DCU2 controller.</td>
<td>High</td>
</tr>
<tr>
<td>FR4</td>
<td>Functional Requirement</td>
<td>All</td>
<td>MCD-1 XCP (XCP on Ethernet V1.1.0) is a communication-transport protocol standard that shall be implemented in order to enhance communication according to</td>
<td>High</td>
</tr>
</tbody>
</table>
requirement FR2.

| FR5 | Functional Requirement | All | The system shall provide a framework to control version and configuration management. Control version must be provided for projects, parameters, and calibration tasks. Configuration management has to provide required boundaries to manage files, models, modules and documentation repositories. This can be handled via repository server. | Medium |
| FR6 | Functional Requirement | All | A set of rules shall be defined and specified in order to adapt current parameter classification into a calibration system. They will be related to responsibilities and divided according to project, functions, devices and modules. | Medium |
| FR7 | Functional Requirement | Client Server | It is needed to have a desktop application that permits configuring DCU2 parameters through an access to a database that records some preliminary and fixed configuration. Besides this application might be accessed from any computer to a server which will hold this application. Desktop application represents a calibration tool and application server will handle a data management system as it is displayed in figure 1. In addition, client server system shall have following features:  
• This system may be accessed in an office or in a remote place where internet or network access is limited.  
• This application can access to other projects configurations through a connection with the toolbox application that is a tool that assists engineers to set up train devices and motors. This represents a task distribution and responsibilities during development task in a distributed team.  
• This system must allow having offline mode and online mode in order to process information via TCP/IP, batch processes or other related.  
• System design shall be composed of 3 components or subsystem. One is the desktop application placed on a client. Other subsystem represents an application server who will be responsible of managing the access to different users and application functionalities. The last subsystem is a database that will hold all the information regarding to variables, parameters, fixed configuration for a project and user authentication. | High |
<p>| FR8  | Functional Requirement          | Client Server/Desktop application | A user friendly GUI shall be developed in order to be implemented in a desktop. There is also needed to consider being scalable to further extensions like new generation of train controllers. This application will be connected to the application server in order to process a set of functionalities explained in requirement FR5. See more details at preliminary project plan document | Medium |
| FR9  | Functional Requirement          | Client Server                      | This component will be responsible of calculating and processing parameter’s handling philosophy, handle profiles, reports, roles and responsibilities. This will also control version and manage configuration management tasks. See further details at preliminary project plan document | Medium |
| FR10 | Functional Requirement          | Client Server                      | A database will record data from parameters, variables and other useful information to assists parameter handler application to run queries. This will also need to provide the necessary framework to fetch data in a distributed development environment. See further details at preliminary project plan document | Medium |
| FR11 | Functional Requirement          | XCP/DCU2                           | A communication interface based on XCP protocol shall be implemented to allow package exchange, command and events based on a master-slave philosophy. In this case, the master will be a calibration tool and the slave a DCU2 component or subcomponent. | Medium |
| FR11.1| Functional Requirement          | XCP/DCU2                           | The communication interface needs to have a slave device identification to recognize DCU2 software version through a string that the XCP station shall provide. | High |
| FR11.2| Functional Requirement          | XCP/DCU2                           | Seed &amp; key XCP features shall be enabled in order to allow individual access protection for calibration, flash programming, synchronous data acquisition and data stimulation. | Medium |
| FR11.3| Functional Requirement          | XCP/DCU2                           | The application interface at the DCU2 needs to initialize the XCP protocol layer and its internal variables before any XCP function is called. | High |
| FR11.4 | Functional Requirement | XCP/DCU2 | The XCP protocol shall handle a data acquisition event channel in order to send packages. This may be either synchronous or asynchronous. This implies handling triggers to sample and transmit DAQ lists that are assigned at the event channel. Event channels shall be configured at CANape previously for synchronous transmission. In case of asynchronous transmission of event codes, this shall be done via Event Packages (EV). | Medium |
| FR11.5 | Functional Requirement | XCP/DCU2 | The communication interface at the DCU2 shall be able to connect or disconnect from XCP master. This also shall be responsible of making transmissions of response or error message in case that it is required. | Medium |
| FR12  | Functional Requirement | XCP/DCU2 | An inner DCU2 software component prototype shall be developed in order to allow signal communication, parameter file (A2L) parsing and memory pages handling in different DCU2 memory sectors. This component will be responsible of handling A2L files between application interface and XCP protocol. For this purpose, it is important to develop a parser that will allow parameter’s memory allocation according to A2L descriptions. This also suggests some DCU2 OS adaptations in order to read and handle A2L descriptions. | Medium |
| FR12.1 | Functional Requirement | XCP/DCU2 | The master calibration system (MCS) may perform an automatic session configuration by transferring MAP filenames from the XCP slave to master. This refers getting any MAP filename as it is needed. | Low |
| FR13  | Functional Requirement | XCP/DCU2 | A C script provides memory page handling philosophy shall be developed in order to differentiate flash memory and DCU2 RAM memory allocation. In particular, flash memory will store parameter information that will represent a working page. Reference page will determine RAM memory allocation from DCU2 Software. This will allow parameter information exchange from DCU software and A2L file. | High |
| FR13.1 | Functional Requirement | XCP/DCU2 | Calibration data organisation for exchanging dynamically parameters at the DCU2 have to be represented in two kinds of layouts: logical and physical. The physical layout is described as sectors that may be used for flash reprogramming or for stating limits and size. The logical layout refers to memory segments where calibration data or parameters are stored. Size and segment size and limits are independent from sector. In particular, the slave’s memory layout shall be described as a continuous physical space where the elements are referenced with 40 bit address (32 bit XCP address + 8 bit XCP address extension). | High |
| FR13.2 | Functional Requirement | XCP/DCU2 | Memory segments at the DCU2 must be composed of one or more memory pages. They describe address data, properties and access. This means, each memory segment have to contain information like name, address, size and other features as long as they are required. | High |
| FR13.3 | Functional Requirement | XCP/DCU2 | A memory page description has to indicate a default page number. This is because a page must be initialized for all segments. Besides, it contains flags that describe the access to XCP or DCU2. | low |
| FR13.4 | Functional Requirement | XCP/DCU2 | The application layer needs to call for a service at the XCP protocol layer to check valid address before writing in a memory segment. This will prevent handling non valid writing access at the DCU2 memory. | High |
| FR13.5 | Functional Requirement | XCP/DCU2 | It is needed to get, and set calibration pages for the specified access mode and data segment. Access mode determines whether is accessed by XCP or by DCU2. Besides, it requires to handle data segment number and logical data page number | High |
| FR13.6 | Functional Requirement | XCP/DCU2 | The C script may copy calibration data pages from source segment and memory page to destination segment and memory page. | low |
| FR13.7 | Functional Requirement | XCP/DCU2 | Memory segments may be saved into a persistent memory when DCU2 is offline in order to maintain a back up during calibration tasks. | medium |
| FR14   | Functional Requirement | XCP/DCU2 | It is needed to propose a design that adapts current DCU2 sub application components into memory page philosophy. This will provide a solution to complement FR13 requirement since it will allow exchange parameters dynamically | High |
| FR14.1 | Functional Requirement | XCP/DCU2 | The application layer shall handle a service called by the protocol layer that allows converting memory address from XCP format into a C Style pointer. This enables CANape to read ASAP2 files from a database or a linker map file. This schema may be also used for downloading and uploading files. | High |
| FR14.2 | Functional Requirement | XCP/DCU2 | The application layer, protocol layer and transport layer shall handle the XCP package format that is composed of a header, a XCP packet and a tail. The xcp packet contains identification field, timestamp field and data field. Further details are described at ASAM specifications. | High |
| FR14.3 | Functional Requirement | XCP/DCU2 | The application layer, protocol layer and transport layer shall handle general data elements that are located at the slave’s memory to be transmitted to the master component. In case of synchronous data transfer model, the Object Description Table (ODT) and DAQs descriptions shall be implemented. | Medium |
| FR15   | Functional Requirement | XCP/DCU2 | It is needed to configure an XCP protocol layer which will responsible to handle DCU2 OS functions required to activate services that are required by the transport layer. This may be done by enabling set of functions that will handle different services that are called by the XCP protocol. Some of these functions are described in XCP protocol layer specification pages 58-65 | High |
| FR16   | Functional Requirement | XCP/DCU2 | A solution shall be designed for integrating application layer and trafo control via function call. | Medium |
| FR17   | Functional Requirement | XCP/DCU2 | A solution should be designed for switching systems into DCU according to requirement FR16 | Low |
| NFR1   | Non Functional Requirements | ALL | Reliability: The system shall be reliable since it needs to generate and handle consistent parameters based on dependencies and mathematical calculations. Besides, it requires controlling parameter information at the DCU2 memory allocation during XCP | Medium |</p>
<table>
<thead>
<tr>
<th>Code</th>
<th>Non Functional Requirements</th>
<th>Category</th>
<th>Description</th>
<th>Priority</th>
</tr>
</thead>
</table>
| NFR2 | Non Functional Requirements | ALL      | Scalability  
All different system and subsystem components must be scalable to allow DCU2 evolution into a new generation of train controllers. This also enhances software reusability to extend parameter handler’s new philosophy in further projects | High     |
| NF3  | Non Functional Requirements | ALL      | Security  
Security policies must be implemented in high level and low level components. This is in order to prevent system intrusion, password sniffing, hacking attacks and lack of historical logs changes and control version. Thus, authentication methods, profile management, password and message encryption, authorization levels, and other mechanisms to enhance security in distributed development during different calibration tasks. | High     |
| NRF3.1| Non Functional Requirements | XCP/DCU2 | Some security policies shall be implemented in order to prevent intrusion during calibration tasks according to requirement FR11.2. Therefore, protection handling features shall be considered from ASAM specifications as well. This also implies implementing a separated interface to handle different functions like Seed & Key | low      |
| NRF3.2| Non Functional Requirements | Client Server | Some authentication methods according to and password encryption policies shall be handled at the client server scenario | low      |
| NF4  | Non Functional Requirements | XCP/DCU2 | Safety  
Because DCU2 is an important controller in a train system, safety critical properties shall be considered in order to avoid controller malfunction and lack of product quality. Therefore, safety critical functions inside XCP –DCU2 System shall be described and implemented carefully. | High     |
<p>| NRF4.1| Non Functional Requirements | XCP/DCU2 | It is needed to check memory availability before running any switching page functions. | High     |
| NRF4.2| Non Functional Requirements | XCP/DCU2 | DCU2 must have loaded some default parameters that shall never be changed and they are basic to start the initialization. | High     |</p>
<table>
<thead>
<tr>
<th>NFR4.3</th>
<th>Non Functional Requirements</th>
<th>XCP/DCU2</th>
<th>XCP/DCU2 must have some mechanism that prevents accidental changes or unexpected changes into the parameter file during switching pages, loading parameter or other calibration tasks.</th>
<th>High</th>
</tr>
</thead>
</table>
| NF5 | Non Functional Requirements | ALL | User Friendly  
Client server System shall be user friendly since it will enhance distributed team productivity and efficiency. Then, a proper help design and document management development will enhance this property. | Medium |
| NF6 | Non Functional Requirements | ALL | Performance  
It is needed to consider a high performance in a client server system in order to provide proper services on time in a distributed team. Besides, XCP over Ethernet protocol shall be configured such that performance in calibration tools can be adapted according to ASAM standards. | High |
| NFR6.1 | Non Functional Requirements | ALL | XCP-DCU2 system shall fulfil performance limits for any kind of package that specifies maximum length, maximum event channel, number of entries and other important features according to type list to be implemented (for instance, DAQ, STIM, etc). See further details at XCP overview ASAM specification | High |
| NF7 | Non Functional Requirements | ALL | Efficiency  
It may be necessary to enhance team efficiency and productivity by managing parameter information in a structured way to allow team collaboration and cooperation. Besides, in order to accomplish calibration tasks on time, it is needed to assure that there exists a process workflow composed of roles, team collaboration and responsibility distribution. | low |
| NF8 | Non Functional Requirements | ALL | Fault Tolerance and availability.  
Client server system must implement different mechanism to assure data integrity, backup and recovery, system availability. This is because the system shall be available to provide services in a distributed team 24 hours per 365 days per year. Therefore, backup policies shall be considered. | High |
| NFR8.1 | Non Functional Requirements | ALL | XCP-DCU2 system must handle a set of initial error handling policies for memory page prototype according to OS or Protocol specifications | High |