
Shaghayegh Kashiyarandi
Ski10001@student.mdh.se

Examiner: Kristina Lundqvist
Mälardalen University, Västerås, Sweden

Supervisor: Barbara Gallina
Mälardalen University, Västerås, Sweden
Abstract

In context of safety oriented system development, safety standards define a set of requirements that constrain the development process targeting safety-critical systems. To certify a safety-oriented development process, such requirements must be fulfilled. For instance, process models specifying plans regarding the development should be provided. Todays, certification process is very time consuming and costly due to the proliferation of standards and the huge amount of evidence that is required. Therefore, considering approaches that are based on reuse is becoming of utmost importance especially in the context of re-certification or in the context of changes regarding the criticality.

“Safety-oriented process lines” allow process engineers to model a sets of safety oriented processes by systematizing their commonalities and variabilities. Processes may vary from each other in terms of stringency required but still in compliance with the requirements.

To model the observed approach and implement that model, several modelling languages and supportive tools have been studied that resulted in selecting “SPEM2.0” and “Eclipse Process Framework Composer” that provide a reasonable support to create reusable safety-oriented process-line models.

This thesis also extends the existing related works by investigating more process elements and variability types and how they can be applied to the real data considering the aspect of Re-use. As a case study to verify the usage of this approach, we applied it to specific portion of a set of cross-domain safety standards, which are ISO26262, EN 50 128 and DO-178C. A detailed guidance to the observed approach and implemented model are provided in this thesis.

Even though the implemented model is limited to restricted part of the mentioned standard processes, the proposed approach can be applied to any sets of cross-domain or inter-domain safety oriented process that have a reasonable number of commonalities even though they vary from each other. In conclusion, the summary of how to obtain a reusable safety oriented process model is presented and few ideas are suggested for the future works in this field.
Acknowledgment

Foremost, I would like to express my sincere gratitude to my supervisor Barbara Gallina for her continuous support, patience, enthusiasm and knowledge. Her guidance and dedication helped me through both my personal and academic development.

Special thanks go to my parents for their unconditional love, help and support. I am forever in their debt for giving me the opportunities and experiences that have made me who I am. Without their support, I could never start this journey and reach this far. Also, I would like to thank my uncle Dariush for his moral support throughout these years.

I would also like to express my gratitude and love to my dear Farshad. His daily support, love, and understanding have strengthened me through this challenging experience.

I am also thankful to my friend Sepideh for her encouraging support and pure friendship.

Finally, I would like to dedicate this thesis to my grandparents. I have been extremely fortunate in my life to have grandparents who have shown me unconditional love and support.
# Table of Contents

Abstract ................................................................................................................................. 1

Acknowledgment .................................................................................................................. 2

1 Introduction ......................................................................................................................... 9

1.1 Problem Formulation and Analysis ................................................................................. 10

1.2 Contribution ..................................................................................................................... 12

1.3 Document Organization ................................................................................................. 12

2 Research Methodology ....................................................................................................... 13

2.1 Literature Review .......................................................................................................... 14

2.2 Case Study Design ......................................................................................................... 15

3 Background ......................................................................................................................... 15

3.1 Safety Critical Systems ................................................................................................. 16

3.2 System Development Processes ..................................................................................... 16

3.3 System Development Process Modelling ....................................................................... 19

3.3.1 ProcPT approach ...................................................................................................... 21

3.3.2 Managing Standards Compliance: Emmerich’s approach ........................................ 23

3.3.3 Tailoring & verifying Software Process ................................................................... 25

3.3.4 Process Line Approach ............................................................................................ 26

3.3.5 Safety Oriented Process line approach .................................................................... 26

3.4 Development Processes within Safety Standards ............................................................ 27

3.4.1 ISO 26262 .................................................................................................................. 28

3.4.2 EN50128 ................................................................................................................... 28

3.4.3 DO-178C .................................................................................................................... 28

3.5 (Re) Certification ........................................................................................................... 29

3.6 Process Modelling Languages ....................................................................................... 29

3.6.1 Little-JIL ..................................................................................................................... 30

3.6.2 SPEM 1.1 ................................................................................................................... 32

3.6.3 UML4SPM; UML for Software Process Modelling .................................................. 33

3.6.4 SPEM 2.0 ................................................................................................................... 35

3.6.5 vSPEM ...................................................................................................................... 37

3.6.6 S-TunExSPEM .......................................................................................................... 38

3.6.7 Supportive Tool: EPF 1.0 .......................................................................................... 39

3.7 Related Works .................................................................................................................. 42
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>Method</td>
<td>43</td>
</tr>
<tr>
<td>4.1</td>
<td>Selecting a Reuse-based Modelling Method</td>
<td>43</td>
</tr>
<tr>
<td>4.2</td>
<td>Selecting a Proper Set of Processes</td>
<td>44</td>
</tr>
<tr>
<td>4.3</td>
<td>Identification of Process Elements</td>
<td>44</td>
</tr>
<tr>
<td>4.4</td>
<td>Selecting an Appropriate Process Modelling Language (PML)</td>
<td>44</td>
</tr>
<tr>
<td>5</td>
<td>Reuse-based solution: Safety Oriented Process Line Approach</td>
<td>45</td>
</tr>
<tr>
<td>5.1</td>
<td>Model the Process Line: SPEM 2.0</td>
<td>48</td>
</tr>
<tr>
<td>5.1.1</td>
<td>Selection of a Proper Set of Processes</td>
<td>49</td>
</tr>
<tr>
<td>5.1.2</td>
<td>Work Scope</td>
<td>50</td>
</tr>
<tr>
<td>5.1.3</td>
<td>Identifying Process Elements</td>
<td>50</td>
</tr>
<tr>
<td>5.1.4</td>
<td>Domain Engineering</td>
<td>52</td>
</tr>
<tr>
<td>5.1.5</td>
<td>Process Engineering</td>
<td>61</td>
</tr>
<tr>
<td>5.2</td>
<td>Export the Model: Machine Readability</td>
<td>62</td>
</tr>
<tr>
<td>5.2.1</td>
<td>Customized Configuration</td>
<td>63</td>
</tr>
<tr>
<td>5.2.2</td>
<td>Export the Model as a XML File</td>
<td>63</td>
</tr>
<tr>
<td>6</td>
<td>Implementation: A Case Study</td>
<td>64</td>
</tr>
<tr>
<td>6.1</td>
<td>Identifying the process elements</td>
<td>64</td>
</tr>
<tr>
<td>6.2</td>
<td>Process-related Domain Engineering</td>
<td>65</td>
</tr>
<tr>
<td>6.3</td>
<td>Process Engineering</td>
<td>69</td>
</tr>
<tr>
<td>6.4</td>
<td>Exchangeability</td>
<td>71</td>
</tr>
<tr>
<td>6.4.1</td>
<td>Library Configuration</td>
<td>72</td>
</tr>
<tr>
<td>6.4.2</td>
<td>XML</td>
<td>73</td>
</tr>
<tr>
<td>6.5</td>
<td>Case Study Evaluation</td>
<td>75</td>
</tr>
<tr>
<td>7</td>
<td>Conclusion</td>
<td>76</td>
</tr>
<tr>
<td>7.1</td>
<td>Summary</td>
<td>76</td>
</tr>
<tr>
<td>7.2</td>
<td>Future work</td>
<td>77</td>
</tr>
<tr>
<td>References</td>
<td></td>
<td>78</td>
</tr>
<tr>
<td>Appendix 1</td>
<td></td>
<td>81</td>
</tr>
<tr>
<td>Appendix 2</td>
<td></td>
<td>84</td>
</tr>
<tr>
<td>Appendix 3</td>
<td></td>
<td>84</td>
</tr>
</tbody>
</table>
# Index of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Research method’s flow of activities</td>
<td>14</td>
</tr>
<tr>
<td>2</td>
<td>Waterfall model.</td>
<td>17</td>
</tr>
<tr>
<td>3</td>
<td>V-Model, taken from [7].</td>
<td>18</td>
</tr>
<tr>
<td>4</td>
<td>Simple Reuse-based model work flow</td>
<td>19</td>
</tr>
<tr>
<td>5</td>
<td>Tailoring Procedures, taken from [19].</td>
<td>21</td>
</tr>
<tr>
<td>6</td>
<td>A section of a GV-Model, taken from [19].</td>
<td>22</td>
</tr>
<tr>
<td>7</td>
<td>Product Flow Schema taken from [19].</td>
<td>22</td>
</tr>
<tr>
<td>8</td>
<td>Product State diagram taken from [19].</td>
<td>22</td>
</tr>
<tr>
<td>9</td>
<td>Tailoring Procedure implemented in ProcPT, taken from [19].</td>
<td>23</td>
</tr>
<tr>
<td>10</td>
<td>Standards and compliance model, taken from [22].</td>
<td>24</td>
</tr>
<tr>
<td>11</td>
<td>Defining a standard compliant in Emmerich’s approach, taken from [22]</td>
<td>25</td>
</tr>
<tr>
<td>12</td>
<td>Little- JIL legend.</td>
<td>31</td>
</tr>
<tr>
<td>13</td>
<td>Portion of an example Little JIL process model</td>
<td>32</td>
</tr>
<tr>
<td>14</td>
<td>A portion of an example UML4SPM model</td>
<td>34</td>
</tr>
<tr>
<td>15</td>
<td>Simple SPEM 2.0 model</td>
<td>36</td>
</tr>
<tr>
<td>16</td>
<td>vSPEM variants and variation points icons based on SPEM icons [29]</td>
<td>37</td>
</tr>
<tr>
<td>17</td>
<td>Simple vSPEM variability example: (a) configurable model, and (b) a valid configuration of this model [30]</td>
<td>38</td>
</tr>
<tr>
<td>18</td>
<td>Set of S-TunExSPEM icons [28]</td>
<td>38</td>
</tr>
<tr>
<td>19</td>
<td>A portion of S-TunExSPEM safety oriented process model [28]</td>
<td>39</td>
</tr>
<tr>
<td>20</td>
<td>Method Content, class diagram [32]</td>
<td>40</td>
</tr>
<tr>
<td>21</td>
<td>A process and related breakdown structure</td>
<td>41</td>
</tr>
<tr>
<td>22</td>
<td>Work flow: how to create the process line</td>
<td>48</td>
</tr>
<tr>
<td>23</td>
<td>A portion of EN 50128:2001-Section 9.4</td>
<td>51</td>
</tr>
<tr>
<td>24</td>
<td>A portion of ISO26262</td>
<td>51</td>
</tr>
<tr>
<td>25</td>
<td>Work products defined at ISO/FDIS 26262-6:2011</td>
<td>52</td>
</tr>
<tr>
<td>26</td>
<td>Base Line created in EPF</td>
<td>53</td>
</tr>
<tr>
<td>27</td>
<td>SPEM 2.0 diagram shows a contribution between Base and Variant roles.</td>
<td>56</td>
</tr>
<tr>
<td>28</td>
<td>SPEM 2.0 diagram illustrate an example of Extend variability type</td>
<td>57</td>
</tr>
<tr>
<td>29</td>
<td>SPEM 2.0 diagram illustrate an example of Replace variability type taken from [5]</td>
<td>58</td>
</tr>
<tr>
<td>30</td>
<td>SPEM 2.0 diagram illustrate an example of Extend- Replace variability type taken from [5]</td>
<td>59</td>
</tr>
<tr>
<td>31</td>
<td>Base Line implemented in EPF</td>
<td>61</td>
</tr>
<tr>
<td>32</td>
<td>An example delivery process implemented in EPF</td>
<td>62</td>
</tr>
<tr>
<td>33</td>
<td>A simple Activity diagram implemented by EPF from Delivery process 1</td>
<td>62</td>
</tr>
<tr>
<td>34</td>
<td>A customized configuration in EPF</td>
<td>63</td>
</tr>
<tr>
<td>35</td>
<td>EPF provides possibility to export the library in different formats</td>
<td>64</td>
</tr>
<tr>
<td>36</td>
<td>Design _Baseline, a portion of the EPF model</td>
<td>65</td>
</tr>
<tr>
<td>37</td>
<td>Base_Commonality_Full, a portion of EPF model</td>
<td>66</td>
</tr>
<tr>
<td>38</td>
<td>Design_Base_Commonality_Partial, a portion of the EPF model</td>
<td>66</td>
</tr>
<tr>
<td>39</td>
<td>Design_Base_Commonality_Partial_Safety, a portion of the EPF model</td>
<td>67</td>
</tr>
</tbody>
</table>
Figure 40. Design_Base_Optional, a portion of the EPF model ........................................ 67
Figure 41. Design_Variants, a portion of the EPF model .................................................. 68
Figure 42. How to assign a proper variability type to a task in EPF .................................. 69
Figure 43. Work breakdown structure view of a delivery process, a portion of a EPF model .. 70
Figure 44. It is possible to set predecessors in EPF ............................................................ 70
Figure 45. Sequence diagram, implemented in EPF ............................................................ 71
Figure 46. Different formats to export a EPF model ............................................................. 72
Figure 47. DO178B Library Configuration implemented in EPF .......................................... 72
Figure 48. Export a portion or entire model as a XML file from EPF .................................. 73
Figure 49. Selecting one or more plug ins to be exported as XML .................................... 73
Figure 50. Briefed XML file exported from a EPF model ................................................... 74
Figure 51. A portion of EPF model .................................................................................... 75
Index of Tables

Table 1. Set of Safety Standards [10]............................................................................................................. 27
Table 2. A subset of SPEM 2.0 process elements .......................................................................................... 35
Table 3. SPEM 2.0 Variability types [5].......................................................................................................... 36
Table 4. Concept mapping [28]......................................................................................................................... 39
Table 5. Reuse-based methods comparison .................................................................................................... 46
Table 6. Process modelling languages comparison result .................................................................................. 49
Table 7. An example of how to create a new Full Commonality element ......................................................... 54
Table 8. An example of how to create a Partial Commonality element ............................................................ 54
Table 9. An example of how to create an Optional element ........................................................................... 55
Table 10. Creating a Base Line as a result of Domain Engineering ................................................................. 60
## Acronyms and Abbreviations

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>COTS</td>
<td>Commercial Off-The-Shelf</td>
</tr>
<tr>
<td>E/E</td>
<td>Electrical and/or Electronic</td>
</tr>
<tr>
<td>EPF</td>
<td>Eclipse Process Framework</td>
</tr>
<tr>
<td>SIL</td>
<td>Safety Integrity Level</td>
</tr>
<tr>
<td>SPEM</td>
<td>System and Software Process Engineering Metamodel</td>
</tr>
<tr>
<td>SSDP</td>
<td>System and Software Development Process</td>
</tr>
<tr>
<td>UML</td>
<td>Unified Modelling Language</td>
</tr>
<tr>
<td>PML</td>
<td>Process Modelling Language</td>
</tr>
</tbody>
</table>
1 Introduction

System development processes are specific set of activities which are required to be accomplished in a specified order, by defined roles and consuming different type of resources, according to the system requirement specification. As systems became bigger and more complex, the development processes turned out to be more difficult to manage, and need of a systematic process to represent these complex processes raised up [7].

Process development modelling is a schematic way of representing a development process. A process model helps to organise all elements included in a development process, from activities to human resources. It helps to sketch the requirements, guidelines and delivery outcomes. It is also a proper way to identify, document and organize tools, technologies and resources which are needed to develop the process in real world.

In the context of safety critical systems, process- based artefacts represent evidences that are used to show compliance with regulations during the certification process [15]. Still no specific model has been defined to represent a set of processes, neither a supportive tool exists to implement such model.

Moreover, in the domain of safety oriented system development, to be- certified –processes, should comply with specific safety standards. A safety standard contains requirements which prescribe/describe an abstract process specification that should be refined and applied to develop the systems. In the real world, the adopted standard process, needs to be customized from one company’s criteria to another one or even from a project to another project. Some elements need to be removed and some need to be added to the process. All these customizing activities, should be done in compliance with the standards’ tailoring rules and safety requirements specification. Producing such compliant process model after each customization is time consuming and costly. Yet, no tool supported approach exists to enable reuse of existed model of processes or process elements.

Above sited limitations, motivate to proposing a systematic modelling approach to enable reusing development processes and process elements in context of safety oriented system development and (re) certification.

To propose such a solution, several safety oriented modelling approaches have been studied by considering following criteria:

- The approach should support reusing the modelled processes and process elements.
- A tool supported modelling language should be existed to represent such modelling approach and its capabilities.
- The model should be expressive, and allows applying ongoing changes to the modelled processes.
Researches and assessments have been done to propose such solution and fulfil considered requirements. This thesis focuses on Safety Oriented Process Line notation, presented in [15]. However, the authors didn’t propose a tool supported modelling language to implement this notation. Also, it is limited to investigating one variability type and process element.

This thesis investigated several modelling languages to implement such notion and thus enabling reuse of process elements that directs to select SPEM 2.0 [5] modelling language. The proposed approach permits process engineers to configure desired processes when moving from 1) one version of the standard to a different one, 2) one domain to a different one, 3) one project to a different one.

This thesis extended the existed works by;

- Selecting a modelling language to model the safety oriented process-line, and providing a systematic guideline to enable modelling reusable process elements in context of safety oriented system development and certification.
- Modelling:
  - Variability types: *Extend* and *Replace* in addition to *Contribute*.
  - Process Elements: *Task*, *Step*, *Role*, *Work product* in addition to *Activity*.
- Applying the approach to a real set of data as an illustrative example to approve the usage of the proposed solution to enable the reuse of modelled processes and process element. Considered model is implemented in Eclipse Process Framework (EPF).

### 1.1 Problem Formulation and Analysis

Safety standards such as ISO26262, EN50128 and DO-178C define safety system development processes to ensure the (software) systems are developed under safety consideration. In addition, these processes should be certified/assessed. This certificate/assessment confirms that the process has been performed in compliance with the related safety standards.

Process-based certification/assessment is costly and time consuming. These resources can be spent for developing and improving products or other relevant activities.

When new versions of standards are issued or when systems (and their associated documentation) are expected to cross domains, the re-certification cost is not negligible.

Nevertheless, reusing the modelled process and elements seems to solve this problem, but still presenting a systematic approach to enabled the reuse of the process elements or even processes within the context of safety critical systems development is a problem itself.
To brighten up the objectives of this thesis, the main problem, concerning time and cost consuming process-based certification has been decomposed into several sub problems which also includes the problems related to the specific case study implemented in this thesis.

1. **Which safety oriented modelling approach enable the reuse of modelled processes and process elements?**

The question is, “which process modelling approach will lead to enable reuse of process elements as much as possible?” and “which final model is easy to understand and maintain regarding different sets of safety standard process or under specific conditions?” As it was mentioned before, there is no certain systematic approach to define a structured reusable safety-oriented model. An approach which can lead to create a model that can support variety of safety oriented process elements, also can provide a mechanism to reuse predefined elements or change them according new requirements and have a strong tool to support it. The method has been used to answer this question is presented in Section 4.1 and the achieved result is indicated in Chapter 5.

2. **Which modelling language can support the selected modelling approach?**

It is important to choose a proper modelling language to present the selected approach. Expressiveness, modularity, tool support, exchangeability and providing good possibilities to reuse the process elements are the most important parameter to choose the modelling language in this thesis. The method has been used to response to this question and related solution are presented in Section 4.4 and 5.1.

3. **Which safety critical set of processes are appropriate to model?**

This selection must be performed among the plenty of existing safety standards, considering the main idea of Reusing Process Elements in Context of Safety Critical System Development and (re)Certification. The method has been followed to answer this question is presented in Section 2.2 and 4.2. The achieved solution is specified in Section 5.1.1.

4. **Which process/process elements would be modelled?**

Safety standards are abstract specification of the development processes. This is essential to identify clear process elements such as role, task or activity from each standard specification. While identifying the process elements, keeping the exact content of the standard is vital. The method has been used to answer this question is presented in Section 4.3. The achieved solution is stated in Section 5.1.3.

5. **Which tool provides the reasonable support to implement the model?**

It is important to select a supportive tool which makes it possible to implement the maximum features of the selected modelling approach and modelling language. Most importantly to provide a mechanism to enable the reuse ability. This question is answered in Section 2.2
1.2 Contribution

This thesis builds on top of the safety-oriented process line notion proposed by Gallina et al. in [15] and provides a feasible methodological support for modelling safety-oriented process lines based on SPEM2.0 [5] /EPF Composer [6].

By adopting safety-oriented process line approach, process engineers have a mean at disposal to clearly identify the safety standard processes and their commonalities and variabilities. As a beneficial component, they can also define and organize the reusable processes elements or either an entire process.

According to the concept of reuse and process line, the candidate processes to be modelled should have a reasonable amount of similarities while they are varying from each other. This thesis considered three safety oriented process development specifications which are selected from different domains of avionic, car and train industries, but within the main field of transportation to present a structured way to model reusable set of processes and process elements. To do this, it was necessary to identify the process elements, their relations and dependencies and organize them in a proper structure. These made it possible to model the safety-oriented process line to enable the reuse of the defined elements as well as representing their relations and how they vary from the original process to the line process.

This thesis implements a safety oriented process line model, including reusable sets of process in SPEM2.0/EPF composer. SPEM2.0 is a standard process modelling language that offers well enough modelling capabilities for the purposes of this thesis. EPF composer is a tool that permits users to model processes in compliance with SPEM2.0.

Briefly, the main contributions of this thesis are:

- A tool-supported methodological guideline to model safety-oriented process lines that enables reuse of processes and process elements and reduce the time and cost of development and (re) certification of safety critical systems.
- An implemented model to illustrate the usage and usefulness of the approach
- This approach has also been used, further illustrated and validated in the context of the EU ARTEMIS SafeCer project [1] and 2 publications in cooperation with TTTECH (Austria) and ViF (Austria) stemmed [17] [31].

1.3 Document Organization

An inventory of the chapters of this report is as follows; Chapter 2, presents the research methodology that has been followed to accomplish this thesis. Chapter 3, recalls the background and existing information that this thesis is based on. Chapter 4, presents the activities which have been followed to reach the solution. Chapter 5, describes the identified solution and defines how to apply the solution to a set of data. In Chapter 6 a case study implemented according to the established solution. Chapter 7,
briefly reviews the achievement of this work and gives a vision about possible future work to optimize this research.

2 Research Methodology

The research process that has been followed to conduct this thesis is designed based on the software engineering process and experimental/case study research method [34] that is customized according to this thesis requirements. The process has been started with problem statement to define the thesis objectives in context of safety critical system development and certification. The research method used to conduct this report consists of two main approaches; literature review and a case study. shows the flow of activities that have been done to make this thesis.
2.1 Literature Review

According to the stated problems and thesis objectives declared in Chapter 1, a research has been done to gather the existing data from earlier studies, techniques and related works in context of safety critical system development and specifically in the field of safety-oriented system development modelling. A set of existed approaches, supportive tools and guidelines have been studied to understand the possibilities and limitation to find a proper solution to resolve this thesis stated problem. The research process continued by data collection procedure; as mentioned in Section 1.1, a set of criteria has been defined to assist collecting the proper data in line with the thesis objectives. Next, all collected data have been analysed and processed, in which leaded to propose a solution to resolve the problem stated in 1.1;

- Proposing a safety oriented modelling approach and a supportive modelling language to enable reuse of process and process elements.

Considering the main goal of this thesis, a research has been done to gather information about existing safety oriented modelling approach which enable re-use of modelled processes and process elements. It was also important that an expressive modelling language exists to support such notation. The other important leman for this selection was existence of an appropriate tool
to support the modelling language. All these lead to select the reuse-based modelling approach, Safety Oriented Process Line, and SPEM 2.0 modelling language to represent the model.

More details about the flow of activities to select the proper safety oriented modelling approach are described in Section 4.1.

2.2 Case Study Design
As Chapter 6 presents, a case study has been designed to validate and verify the usage of proposed solution within this thesis. The suggested approach has been applied to a selected set of real data and a safety oriented process line model implemented that enables reuse of processes and process elements. The procedure to design such a case study starts by going through each problem question stated in Section 1.1.

- **Selection of an appropriate set of safety oriented processes.**
  According to the concept of re-use, the selected set of processes should have a reasonable number of commonalities while they have their own specific attributed. Having this in mind, since this thesis is partially supported by SafeCer project [1], a set of safety oriented development specification from the main field of transportation but from different domains have been selected; which are ISO26262 (Vehicle industry), EN50128 (Train industry) and DO-178C (Avionic industry). This selection also proves the proposed approach can be applied to the cross-domain as well as intra-domain processes. Activity flow of this selection is presented in Section 4.2.

- **Scope of the case study.**
  The case study is limited to partially model the Design Activity from the Software development process of the selected safety oriented specifications. The considered process elements are; Activity, Task, Step, Role, Work product and Guildline.

- **Selection of the proper tool to implement the model.**
  According to the selected modelling approach and the representing modelling language, SPEM 2.0; Eclipse Process Framework (EPF) used to implement the model. EPF provides a reasonable support to model safety oriented processes and process elements. Also, makes it possible to model different Variability types to support the relations within the process elements in a process line to enable the reuse of process and process elements.

3 Background
This chapter presents the background information on which this thesis is based on. Section 3.1 briefly introduces the safety critical systems. Section 3.2 presents a short elucidation of safety oriented software development processes. Section 3.3 discusses the importance of process modelling and recalls some
safety critical process modelling methods. Section 3.4 gives brief information about safety critical system development process in complying with safety standards. Also, presents a set of three safety standards which have been considered within this thesis work. Section 3.5 shortly reviews the concept of (Re) Certification. Section 3.6 recalls some modelling languages and corresponding tool supports. Related works are presented in Section 3.7.

3.1 Safety Critical Systems
A safety critical system is a system which its malfunction may result in a catastrophic event, severe damage to the equipment and the environment or even results in death or serious injury to people [8]. It means the human safety is relied on the correct operation of the system.

Nowadays there are many systems developed that can influence the safety of human and the environment. The role of these systems is growing immensely in everyday life. Embedded Control Systems are traditional safety related examples. From railway signalling systems, which directing trains and preventing them from colliding, to avionics systems, which control air crafts, to medical systems. They are obvious examples of safety critical systems. Even a simple traffic light can be counted as a safety system, since its failure can lead to a harmful event [8].

3.2 System Development Processes
This section recalls essential information on system development processes which is built on top of the research presented by Marciniak [7].

System Development Process is a specific set of activities which are ordered to be performed to achieve a required system. Preforming these activities in a defined order by a role/agent with usage of the resources is expected to lead to develop the requested system. Typically, this chain of activities can be organized in several phases, which classically named: requirements specification, analysis and design, implementation, verification and maintenance. Since industries started to develop complex systems, they faced the need of a systematic progress to manage the development process. To overcome this need, different process methodologies have been introduced.

A basic system development process includes a set of the following activities:

**Planning**: Where do systems come from? Is it a new system? or an existd feasible system will be replaced or supplement with a new one?

**Requirement Analysis and Specification**: What is the problem the new software/system needs to solve? What is the desired performance of the software/system? Which resources are needed to support and maintenance the new software/system operation?
**Architectural Design and Specification:** Defines the subsystems, their components and modules and their internal relations and interfaces to be used in a detailed Design activity.

**Implementation:** Implementing an operational source code based at predefine specification.

**Validation:** Justifying that each component implemented accurately to meet its specified requirements.

**System Integration and Verification:** Combining all components and modules to subsystems and subsystems to product and verifying the performance of all connections and interfaces regarding the required specification.

**Deployment and Installation:** deploying and installing the produced in the host environment.

**Training and Use:** train system users how to efficiently work with the new system and provide them with effective system guidance to make them aware about the abilities and limitations of the system.

**Maintenance:** Supporting the system in the host site by providing requested repairs, performance improvements or further changes.

In addition to these classic sets of activities, a safety-oriented development process includes more specific assessments and activities like *Safety Analysis and Specification* regarding the system safety assurance.

Following paragraphs recall some existing general-purpose system development methods:

**Waterfall** is presented by Royce [9]. Each phase in the Waterfall model comes sequentially off after each other with a predefined start and finishing point. Figure 2, shows the sequence of phases in Waterfall model.

![Waterfall model](image)

**Figure 2.** Waterfall model.

In waterfall model, the output of one phase is considered as the input of the next phase. It means that the next phase cannot be started before the previous one is finished. As Figure 2 illustrated, Waterfall model is simple and easy to follow, but as there is no iteration to be considered, ongoing changes are suitable
for modelling smaller projects with well-defined requirements. The waterfall model is not flexible to ongoing changes. Any changes during the process will affect all previous works.

**V-Model** [7] introduced as an advanced waterfall model. It changes the linear water fall model to a V shaped model. In this model, a validating activity is added to each phase to assure that the output of each phase is adapted to the identified requirements. Figure 3, illustrates the V-Model work flow.

![V-Model Diagram](image)

**Figure 3. V-Model, taken from [7].**

V- Model is used widely as it is simple and easy to use, but same as the Waterfall model it is not flexible to on-going changed during the process.

**Reuse-based** method is another system development process approach which has recently been studied [13] [15] [26-28]. The focus of this method is to reuse the existing process elements to develop new systems. Figure 4 illustrates the Reuse- based method abstract work flow. Few examples of Reuse- based methods will be briefly reviewed in Section 3.3.
3.3 **System Development Process Modelling**

Process Model is representing a development process using a Process Modelling Language (PML) [7]. A process model presents an explicit view of process elements, their relationships and sequences. This work scaled to focus on the software development process portion of the main system development process. The software development process provides a precise specification of what are the process elements and how they take roles within a software development project.

The primary reason for the rise of development process modelling concept is back to the 1950’s and 1960’s which points to the date of the earliest large software developing project. That project was to provide a conceptual schema to serve as a source for preparation, staffing, budgeting and leading the software development activities [7].

There are verities of purposes to motivate organizing a system development process by modelling the process. A process model potentially serves as:

- schema to manage all involved systems/software development project’s activities from planning and human resourcing to budgeting and every activity which takes place within the project
- sketch of required document that should be delivered to the client.
- source for to identify proper tools, technologies and methodologies for performing process activities.
- approximate outline of resource allocation.
• guidance to conduct practical studies regarding what affects the performance, cost and quality of the product.

A process model is effective when it is easy to understand, flexible in applying the on-going changes and improvements and has the possibility to model detailed process elements as well as structured or atomic high-level elements. A process model can perform the role of a guideline to enforce the software engineering criteria to the process. Also, it may provide the process with the possibility of being automated. A process model tries to clarify:

• What activity should be done?
• Who should do this activity?
• Which tool support the activity to be performed?
• What should be the result of this activity?
• And when should this activity perform?

To explain above sited questions, each process model proposes some process elements and defines different types of relations between these elements. All these together give a prescriptive view of the process. From the static point of view, the skeleton of a process model consists of four process elements, called activity, product, role and tool. Each of these elements can be defined as a structured element which built from other process elements. For example, each activity can be consisted of number of steps, or based on the software specification, one product can be structured as a set of products. On the other hand, the answer to the above sited question “When activity should be done? “defines the sequence and priority of the activities which presents the dynamic view of the process model.

Following section briefly reviews the definition of a set of basic process elements which is used within this report.

The followings are a set of the basic elements which are essential parts of each system/software development process:

• **Activity** is usually a structured element consist of several *Tasks* that can be a set of *Steps* itself. These tasks are assigned to the role and their performance results to a product.
• **Role** defines a human or system agent which is responsible to perform an activity. It is possible to assign more than one role to an agent.
• **Work Product** it is a result of performed activity(s). A product can be defined structured as a set of products or being atomic. There are three types of work products: *Artefacts* which are definitions of other tangible work products. *Deliverables* are a set of products, mostly those Artefacts which presents an output of a process to a customer and *Outcomes* are the intangible results of a process.
like an optimized network. Source code, scripts, architectural files, analysis, documents, executable or any other deliverables count as Product.

- **Tool** includes any means used during the process development, including editors, compilers, debuggers, process modelling tools, etc.

- **Guideline** – provides recommendations and ‘how to’ information regarding the related element. It can include alternatives or rules about how to use the related element. For example, a guideline can recommend different techniques to operate a task or give a detailed step by step guide to perform an activity to develop the requested work products.

In the following section, few safety critical process modelling approaches have been reviewed.

### 3.3.1 ProcPT approach

Authors of tailoring and conformance testing of software processes: The ProcPT approach [19] presented a procedure to tailor processes regarding the new project specific requirements and testing the result due to the compliance with the under-consideration standard and quality management requirements. This section briefly overviews the ProcPT approach. Figure 5 illustrates two possible work flows of the tailoring procedures to reach the project specific development process from the given process model.

![Tailoring Procedures, taken from [19]](image)

As the initial step, it is needed to formalize the given process model; this approach decomposes the process to the activities and document basic elements which are identified by specific rules. Specified activities and their related produced or modified documents will be modelled via German process model VORGEHENSMODELL known as GV model [21]. GV- Model counts documents as Products and express the flow between sub models by arrows. Figure 6 illustrates a sub model of the GV- Model.
Also, a Product flow schema and a product state diagram would be assigned to each product, which presents the attribution of each product and its modification/deletion flow from the beginning to the end. Figure 7 and Figure 8, present an example of a Product flow schema and a Product state diagram.

<table>
<thead>
<tr>
<th>Table 1: Product Flow Scheme</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>INPUT</strong></td>
</tr>
<tr>
<td>product1 from activity</td>
</tr>
<tr>
<td>product2.xxx</td>
</tr>
</tbody>
</table>

Figure 7. Product Flow Schema taken from [19]

Figure 8. Product State diagram taken from [19]

The second step is to test and improve the process model, which is basically to test and analyse the process and find those elements which should be removed from the process as they are not fulfil the standard conformance. These could be deleted by the act of Deletion.
The third and last step is the tailoring to the project-specific procedure. In this step due to the optimizing process, those activities and related products which can be removed from the process model are identified and will be deleted under the specific rules. This will guarantee the compatibility of the tailored process to the standard process.

The procedures of tailoring and conformance testing in this approach are implemented by the tool ProcPT which itself implemented in PROLOG [20]. The ProcPT database consists of products and activity facts, deletion conditions and rules for conformance testing and tailoring. Figure 9 presents a tailoring procedure implemented in ProcPT.

![Tailoring Procedure implemented in ProcPT](image)

**Figure 9. Tailoring Procedure implemented in ProcPT, taken from [19]**

### 3.3.2 Managing Standards Compliance: Emmerich’s approach

Managing Standards Compliance; Emmerich’s approach [22] as its title indicates, focuses on the compliance of the project specific process with the standard process. Emmerich’s approach is a research purpose approach which presents a *Model of standards and support for compliance management*. Figure 10 illustrates an entity relationship diagram which presents the general principle elements of the standard and related compliance model.

The objective of this approach is to support users to estimate the current compliance state of any part of the process respecting the standard specification. A subset of the Unified Modelling Language, UML, is used as the modelling language to model the different aspects of the approach, such as documents,
their components and their assigned values modelled by using the UML classes and their attributes. Or the aggregation relationship used to model the tree of how documents decomposed.

![Figure 10. Standards and compliance model, taken from [22]](image)

In this approach, a standard is intended as a set of practices. Each practice has some properties which introduce some of its static attributes.

The Emmerich’s approach is not focused on the dynamic flow of the process, yet it concerns the flow of actions which cause a product to reach the compliant state. It monitors the events to find any noncompliance with the standard. Figure 11 illustrates the method for defining the standard compliant in Emmerich’s approach.
Dynamic Object-Oriented Requirements System (DOORS) [23] used to implement the approach based on a document management system. DOORS, using a Dynamic eXtension Language (DXL), which enables to implement the concept of triggers. Triggers starts the process to respond to an event in associate with an action. This feature is used to prevent developers accomplish those actions which are forbidden based on the standard specification.

### 3.3.3 Tailoring & verifying Software Process

In [12] a clear, formal approach specification has been proposed as a method for tailoring process modules. In this approach, each process module is considered as an *encapsulated reusable unit of process fragment*. Activity and artefacts have been noticed as process entities. Each Activity entity has some properties such as Name, Type, Input or Output. Similarly, an artefact has the same properties and some additional values which represents if the artefact is an input, output or pre-existence, etc. A simple modelling notation called Activity- Artefact Graph, AAG, has been used to represent the process modules and their relationships.

To compliance of the actual process with the standard process, four tailoring operations have been considered in this approach: addition, deletion, splitting and merging.

After the Static Process Verification, Syntactic Correctness Checking and Standard Compliance Evaluation will be applied to the tailored process to guarantee a reasonable level of compatibility with
the standard process. The tailoring and verification operations of this approach implemented in a software management tool, is called SoftPM [24].

3.3.4 Process Line Approach
A process line can be considered as a well-defined reference process introduced by Ternite [33]. The process line includes the basis process elements as well as the dynamic work flow of the general process. The new process can be derived from this base line by reusing the identified process work flow and elements as they are, or as a variant of the existing element by adopting the variation mechanism, for example, to add or remove attributes or even tailoring a new element to the base process. These mechanisms allow to define new processes which vary the original process line by add, remove or change an excited element or sequence of the work flow.

The skeleton of a process line defines by two major concepts; the static presentation of the process elements and dynamic demonstration of these identified elements within the process workflow.

- **Static view**
The initial activity of defining a process line is to construct the static presentation of it. This static presentation is the bank of the identified elements derived from the reference process. The original predefined elements which construct the Baseline can be reused to define new standard, company or project process specific elements, as they are or as a variation of the original element regarding the new process requirements and policies. In this work, the category consists of the original elements called Base Elements and the standard/company/project varied elements, defined in a category named Variants. More details about Process-line’s Static view will be presented in Section 5.1.4.

- **Dynamic view**
The dynamic presentation of the process line defines the sequence of the identified activities in each specific process which diverges from the process line. In this work, these dynamic views of the processes have been called delivery process. More details about Process-line’s dynamic view will be presented in Section 5.1.5.

3.3.5 Safety Oriented Process line approach
To develop safety oriented systems, the development process should comply with special safety oriented process specification as standard guidelines. The verification phase to certify the process is very time and cost consuming. Considering the level of commonality between safety standards and how each standard specification varies from the other one, the Process line approach is specified and extended to safety-oriented process-line approach with a definite focus on the safety oriented development processes [15].
There are some modelling languages exist to represent such a model. UML4SPM, Litlle-JIL, SPEM 2.0, X-SPEM and V-SPEM which will be reviewed in Section 3.6.

3.4 Development Processes within Safety Standards

As mentioned earlier, many systems that are related to the everyday life can influence our safety; but has the concept of safety been considered during the lifecycle of developing these systems?

As the first step, it is important to distinguish if the system is a safety critical system. The answer to the following question would lead to identifying a safety critical system: “Can this system’s failure result in a hazard? And if yes, can the hazard cause harm?” If the answer is No, so it is Not a safety critical system. If the answer is Yes, then the system counts as a safety critical system and the next questions will be “How much the failure of system impacts the safety? And “How the hazardous result of the system malfunction can be reduced?” Safety standards have been raised up to answer these questions. Safety standards assigned 5 different safety integrity levels to the systems from level 0 which means system failure causes no safety issue, so it is a Not safety related system to level 4 which is the highest level and means system failure can cause death or serious injury [10]. Table 1 gives a set of safety standards within different domains and their descriptions.

<table>
<thead>
<tr>
<th>Standard</th>
<th>Context</th>
</tr>
</thead>
<tbody>
<tr>
<td>ISO9001/EN29001/BS5750 part1</td>
<td>This is the recommended minimum standard of quality system for software with a safety integrity level of 0, and an essential prerequisite for higher integrity levels.</td>
</tr>
<tr>
<td>EN50128</td>
<td>Railway industry</td>
</tr>
<tr>
<td>ISO26262</td>
<td>Road vehicles — Functional safety</td>
</tr>
<tr>
<td>IEC880</td>
<td>Nuclear Powers Stations software</td>
</tr>
<tr>
<td>DO-178C</td>
<td>Avionics and airborne systems.</td>
</tr>
<tr>
<td>Defence Standard 00-56</td>
<td>Defence Systems Containing Programmable Electronics</td>
</tr>
</tbody>
</table>

Table 1. Set of Safety Standards [10]

This thesis deliberated three well known prescriptive safety system development process specifications within the transportation filed by considering different sub domains, called: EN 50128 Railway applications — Communications, signalling and processing systems — Software for railway control and protection systems which has the status of a British Standard. International standard ISO26262 Road vehicles — Functional safety. DO-178C Software Considerations in Airborne Systems and Equipment Certification. These standards are the results of documenting the classic system development process’s
activities together with the safety-oriented activities to assure that the result meets all specified systems and safety requirements.

The following sections provide a short review of each safety standard. Also, some examples\(^1\) have been provided to clarify how a structured process and its elements have been refined and reorganized from the standard abstract specification. Without doing this, it was not possible to create a structured process for each standard and compare those together to find out the further **Reuse** possibilities of processes and elements in a systematic approach. It has been strictly noticed to keep the exact content of the standard’s process while refining the process elements of compliance with standards.

### 3.4.1 ISO 26262

ISO 26262 regulates the automotive domain and more specifically it is intended to be applied to safety-related systems that include one or more electrical and/or electronic (E/E) systems and that are installed in series production passenger cars with a maximum gross vehicle mass up to 3500 Kg. [2].

This thesis focused on the Software architectural design clause of ISO-26262 Road vehicles — Functional safety — Part 6: Product development at the software level document which issued in 2011.

### 3.4.2 EN50128

EN50128 regulates the railway domain, and more specifically, it is intended to be applied in the field of signalling and control. EN50128 is a European standard provided by the European Committee for Electro-technical standardization (CENELEC) [3]. It concentrates on the methods which need to be used to provide software which meets the demands for safety, which are placed upon it by these wider considerations. This process is constituted of ten normative phases. Those that are related to the software design are: Software requirements specification, Software architecture, Software design and implementation. This work focused on the phase Software design and implementation and specifically Software design activity.

### 3.4.3 DO-178C\(^2\)

Considerations in Airborne Systems and Equipment Certification, DO-178C [4] provides guidelines to produce software for airborne systems and equipment that performs its intended function with a level of confidence in safety that complies with airworthiness requirements. The provided guidelines are in three forms as: objectives for software life cycle processes, descriptions of activities and design considerations for achieving those objectives, and

---

\(^1\) All considered processes and identified process elements will be presented in details Appendix 1.

\(^2\) This research and provided solution is valid for DO-178B and DO-178C since no significant changes were introduced concerning the portion of the process that has been considered. Case study modelled based on DO-178B.
descriptions of the evidence that indicate that the objectives have been satisfied. In this report, we focus on the guidelines regarding the descriptions of activities and design considerations. More specifically, we emphasise on software design process.

3.5 (Re) Certification
In the Safety Critical System sake of art, re-certification applies to the act of certifying a certified system under a different standard or another version of the current standard. Re-certification enables the use of the system under the new requirement or environment. Re-certification as an intermittent reassessment is used to verify that competencies are maintained the specified requirements. On the other hand, Recertification could act to maintain the current certified system regarding the new requirement or deficiency of the existed certified system.

3.6 Process Modelling Languages
Process Models can be presented by using different notations (e.g. Graphical, textual [natural language, machine-readable]) in different abstract levels (e.g., life cycle level, engineering process level, atomic step level). Generally, a development process is represented as a model by using a process modelling language. Formalization of the software development process by using an appropriate process modelling language provides some significant results [7] for instance:

- Explicit specification of development processes
- Enabling Editing and change managing the PLs under an appropriate tool support
- Ability to generate a machine-readable model of the specified process
- Ability to systematic verification and maintenance of the system/software process

It is important to choose a proper language to represent a development process as a model. More precisely, to choose an appropriate language to provide a safety oriented cross domain process line model, which is the main concern of this report, a language which makes it able to provide reusability of process elements and support the different kind of relation types between elements regarding the Process-Line approach was needed. To select such a process modelling language, some attributes should be considered: Expressiveness, flexibility, the existence of a functional mechanism to enabled re-use of elements and execute ability are attributes which are the most important to select a proper language. In addition, the existence of an applicable tool to support the language is also very important.

Following paragraphs defines these target attributes briefly:

- **Expressiveness and Flexibility**: This attribute shows the ability of the language that how good it can express a process through its notation. How many possibilities map the process conceptual elements
into language constructs. A good process modelling language is expected to define the complex development processes in a manner that model being easy to understand and follow.

A fixable modelling language provides possibilities to apply ongoing changes to the model and it provides the facility to modify the process elements on the fly and change their relationships accordingly. This criterion is also about how rich is tailoring rules of the modelling languages.

- **Execute** and **Exchange ability**: It is important if the modelling language can represent the development process as an interpretable model. Nowadays companies are more and more to automating the development processes as much as possible, which brings up the requirement for executable process models [16]. The exchangeability is the possibility to use the interpreted code from the model as an input to another machine. In the new environment, the code can be processed based on the specific requirements.

- **Reuse ability mechanism**: specifically, for this work it should be evaluated if the modelling language provides any possibility to model the variability relationships between the process elements which defined in Process-line approach to support the reuse aspect.

- **Tools Support**: has any specific tool been developed to support the modelling language? If yes, how good these tools are developed to support each specific modelling language.

Following subsections present a review of several modelling languages which have been studied to select an appropriate language modelling the considered process line in this thesis work. The method of the compression and the result will be presented in more details in Chapter 5 and 6.

### 3.6.1 Little-JIL

Little-JIL is based on the JIL programming language XML (Extensible Markup Language) and SGML (Standard Generalized Markup Language) [25]. This modelling language is visual and the model is presented with graphical symbols as shown in Figure 12. Little-JIL is an agent coordination language. An Agent is the actor of a Step within the process. Step is a work unit which assigned to the related Agent. In other words, it is representing the concept of Role as a process element.
Figure 12. Little-JIL legend

The process, flows from a step to another one regarding the different Sequencing and Continuation badge, which are assigned to each step due to the process requirements. Steps can be defined as a composition of sub steps. Figure 13 presents an example process modelled in Little-JIL. Little-JIL semantic is a rich language which allows proactive and reactive control flow features. The reactive control ability includes reaction to error situations and external events in the form of reaction badges. Little – JIL also empowered by rich exception handling with ability of recovery from failures.
As it is mentioned above Little-JIL is based on the JIL which is a high-level programming language. This makes it possible to execute the Little-JIL. The JIL’s interpreter maps the graphical model to the textual notation.

Regarding reuse ability, no formal specific procedure, tailoring rules or variety types is defined to make it possible to reuse the process elements.

Despite the fact that there is some research-based tools, developed to support Little JIL, no formal commercial tool has been implemented yet.

### 3.6.2 SPEM 1.1

Software Process Engineering Meta model, SPEM version 1.1 [27] specifically defined to represent the software development process and related documents. To model the relation and interaction between elements, SPEM 1.1 adopts the UML 1.4 rules and notation. There are several tools developed to implement a SPEM 1.1 model such as IBM Rational Workbench and Osellus IRIS Process Author. However, the implemented model is not executable and it is not possible to convert the model to a machine-readable code. We don’t discuss more details about SPEM 1.1 since a new version of the Software Process Engineering Meta model, SPEM 2.0, is already developed by OMG and will be reviewed in Section 3.6.4.
3.6.3 UML4SPM; UML for Software Process Modelling

UML4SPM [26] is based on UML 2.0 and has been introduced to overcome the existing lack of SPEM1.1 which addressed in Subsection 2.6.2. UML4SPM consists of two packages. First is SPEM_Foundation, which includes all UML2.0 packages that are required to define the basic process features such as Activities and Behaviours. The second one is SPEM_Extension, which includes Process Structure and Work Product packages to extend UML2.0 with required semantic for software process modelling [26] UML4SPM enables to model the basic process elements: Activity, Work Product, Role, Human resource and Tool. Regarding the evolution of the process, UML4SPM focuses on the static view of the process. Figure 14, presents a portion of an example process which modelled with UML4SPM.
Thanks to UML2.0, tools developed to support UML2.0 can be used to implement the UML4SPM as well. In addition, as described in [26], thanks to specific UML2.0 packages, *Activities* and *Actions*, the UML4SPM models are executable.
3.6.4 SPEM 2.0

Software & Systems Process Engineering Meta-Model Specification, version 2.0 [5], known as SPEM 2.0 tries to provide more abilities and enriched structure by adopting UML 2.0 to prevail the existing limitations of SEPM 1.1.

SPEM 2.0 is not going deep to specify domains or disciplines as project management rules, etc. So, it can be used widely in different domains. SPEM 2.0 structure contains higher level abstract packages. Each of these logical units have been extended to subunits to provide more abilities and possibilities to model the complex system and different levels of that system. More details about SPEM 2.0 can be found in [5].

SPEM 2.0 provides a way to engineer the development process with the effective use of conceptual frameworks and to do an efficient modelling, documenting and managing during the development process. Table 2 presents a subset of SPEM2.0 process elements’ symbols.

<table>
<thead>
<tr>
<th>SPEM 2.0 Symbol</th>
<th>Description</th>
<th>SPEM 2.0 Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Activity" /></td>
<td>Activity</td>
<td><img src="image" alt="Tool" /></td>
<td>Tool</td>
</tr>
<tr>
<td><img src="image" alt="Role" /></td>
<td>Role</td>
<td><img src="image" alt="Guideline" /></td>
<td>Guideline</td>
</tr>
<tr>
<td><img src="image" alt="Task" /></td>
<td>Task</td>
<td><img src="image" alt="Role in use" /></td>
<td>Role in use</td>
</tr>
<tr>
<td><img src="image" alt="Step" /></td>
<td>Step</td>
<td><img src="image" alt="Task in use" /></td>
<td>Task in use</td>
</tr>
<tr>
<td><img src="image" alt="Work product" /></td>
<td>Work product</td>
<td><img src="image" alt="Work product in use" /></td>
<td>Work product in use</td>
</tr>
</tbody>
</table>

Table 2. A subset of SPEM 2.0 process elements

SPEM 2.0 also provides good facilities to model different types of relationships between process elements. This variation mechanism provides a possibility to reuse the existing elements to model new defined processes or tailor new identified elements to an existing structure to meet newly defined requirements. Regarding variability types, two elements are included, the base element and the variable element. Each variability type defines a different principle to vary the variable element from the base element. Table 4 overviews these Variability types and their principles. These variability types will be used while implementing the case study of this thesis work to enable a mechanism to re-use the identified elements and create customized processes. These will be discussed in detail in Section 5.1.4 and 5.1.5.
<table>
<thead>
<tr>
<th>Variability Type</th>
<th>Principle</th>
</tr>
</thead>
<tbody>
<tr>
<td>Contributes</td>
<td>Only add the attributes to the base element and never overwrite the information from the base.</td>
</tr>
<tr>
<td>Extends</td>
<td>The Variability element inherits the extra properties from the base element while keep its own properties.</td>
</tr>
<tr>
<td>Replace</td>
<td>The base element being logically replaced by the variability element’s properties.</td>
</tr>
<tr>
<td>Extends-Replaces</td>
<td>Integrate the effect of Extend and Replace variances at the same time.</td>
</tr>
</tbody>
</table>

Table 3. SPEM 2.0 Variability types [5]

Figure 15 illustrates a simple process that is modelled with SPEM 2.0.

Figure 15. Simple SPEM 2.0 model
Same as SPEM1.1 there are several tools available to implement the SPEM 2.0 models. Eclipse Process Framework, EPF [6] is an open source project within the Eclipse foundation which is mainly based on OMG SPEM standard. RMC (Rational Method Composer) from IBM and IRIS Process Author from Osellus are also available to support SPEM2.0 model implementation. On the other hand, there are some efforts to convert SPEM 2.0 model to an executable model [29] but there is no built-in interpreter or language feature to fully address the Execute-ability.

### 3.6.5 vSPEM

vSPEM [29] modelling language is the result of an empirical study toward modelling software process variability. SPEM can be extended by adding some new variability construct that enriches it with the understand ability of its notation. To aim this goal, a new package called Process Line Components has been added to SPEM Meta model. This new package provides a new process variability mechanism based on variation points and variants; “a process is created when variant occupies variation points”. The detailed description of this mechanism can be found in [29]. Based on this mechanism and to make the variant-rich models more understandable, some new icons have been defined by the authors. All these new icons are based on the original SPEM icons and represent the variants and variation points elements. Figure 16, shows these vSPEM icons versus SPEM icons.

![vSPEM icons versus SPEM icons](image)

**Figure 16. vSPEM variants and variation points icons based on SPEM icons [29]**

The relationship between the variables and the variation points in vSPEM is defined by adopting the concept of the variability relation Replace from SPEM. To represent the other variability relation, Extension, Contribution and Extend-Replace, vSPEM defined new icons [30]. Figure 17 illustrates a simple vSPEM variability model.
Figure 17. Simple vSPEM variability example: (a) configurable model, and (b) a valid configuration of this model [30]

It should be mentioned that vSPEM is a research based language and there are ongoing studies to improve the semantics of vSPEM as well as the aspects of variants, variation points and the variability mechanism. And no tool has been implemented to support specific vSPEM icons and relationships.

3.6.6 S-TunExSPEM

S-TunExSPEM [28] is another SPEM extension research towards the model and exchange the safety oriented processes. The authors put effort to enrich SPEM by adding specific attributes and icons to make it more specific supportive to model safety oriented processes. They also presented a mapping between the S-TunExSPEM concepts and XPDL 2.2 [28] which provide the exchangeability of safety related processes. Figure 18 shows the core new defined icons of S-TunExSPEM.

Figure 18. Set of S-TunExSPEM icons [28]

Table 4 represents the concept mapping of S-TunExSPEM concepts and XPDL 2.2
S-TunExSPEM is also a research based SPEM extension and still there is no tool developed to implement the specific icons and semantic of it. Figure 19 illustrates a simple process modelled by S-TunExSPEM.

### Table 4. Concept mapping [28]

<table>
<thead>
<tr>
<th>S-TunExSPEM</th>
<th>XPDL 2.2</th>
</tr>
</thead>
<tbody>
<tr>
<td>SafetyRoleDefinition</td>
<td>Participant + extended attribute</td>
</tr>
<tr>
<td>SafetyTaskUse</td>
<td>Activity + extended attribute</td>
</tr>
<tr>
<td>SafetyWorkProductUse</td>
<td>DataObject + extended attribute</td>
</tr>
<tr>
<td>SafetyRoleUse</td>
<td>Lane in a pool + extended attribute</td>
</tr>
<tr>
<td>SafetyGuidance</td>
<td>Annotation + extended attribute</td>
</tr>
<tr>
<td>SafetyTool</td>
<td>Application + extended attribute</td>
</tr>
<tr>
<td>SafetyWorkSequence</td>
<td>Transition + extended attribute. Remark: from-activity or to-activity must be an activity representing a SafetyTaskUse</td>
</tr>
</tbody>
</table>

Figure 19. A portion of S-TunExSPEM safety oriented process model [28]

### 3.6.7 Supportive Tool: EPF 1.0

Eclipse Process Framework [6], known as EPF Composer is a platform to implement the process models. It helps process engineers and project managers to tailor, manage, author and publish the development processes. Since EPF is the tool used to implement the case study models of this work, following sections provide brief information regarding its structure and specifications. EPF detail specification can be found in [6]. The EPF composer consists of 12 main components. Following paragraph briefly describes some of them to provide a basic understanding of the EPF structure:

- **Common** component provides all EPF composer components with common infrastructure services like, XML parsing, file Input/Output and error handling.
• **UMA** or Unified Method Architecture defines the structure of the EPF Method Contacts and Processes which are the main categories consist of UMA model classes. Method Contacts describe roles, related tasks and produced work products as well as the supportive guidance’s. Method contents can be considered as the process elements pool where these elements can be reused to develop different lifecycles. Figure 20 represents a Method Content class diagram.

![Method Content Class Diagram](image)

**Figure 20. Method Content, class diagram [32]**

• **Delivery Process** defines the dynamic aspect of the model: the development lifecycle. The process presents the workflows and/or breakdown structures which define the sequence of tasks over time.

• **Plug ins, Method contents**: these are categorized units which let organize the model in a proper way.

Figure 21 shows an example process and related work breakdown structure modelled in EPF Composer.
Figure 21. A process and related breakdown structure

- **Export/Import** component allows method contents, plugins and libraries transfer as XMI files.
- **XML Export/Import** component empowers the EPF models with exchange ability. It makes it possible to Import and Export the method library content to and from an XML file. This component allows the third party to exchange and convert their method content to a UMA-based method library.

Several open standards such as XML Meta Interchange (XMI 2.1), Unified Modelling Language (UML 2.0) and DHTML has been used to design and implement the EPF 1.0 composer. Each standard empowers the EPF 1.0 with specific abilities, for example:

- **XMI 2.1**, is a standard for interchanging and storing the metadata in XML format. The XMI lets EPF to store the method libraries in an XMI file. Method Plug-ins and Configuration use the XMI file format as well.
- **UML 2.0** has been used to store and present the graphs and information related to Activity and Work Product Dependency diagrams.
- **DHTML** which includes several web related specifications, such as HyperText Mark-up Language (HTML), Java Script and Document Object Model (DOM) which is used for parsing and accessing HTML and XML documents [6].
3.7 Related Works

Parallel to the expansion of the complexity of the systems, specifically softwares, as well as the size of the organizations, the need of a systematic way to organize and lead the huge amount of data grows up every day. Efforts to define a structure to manage and represent data and resources leads to invention of different methods to specify the system and software development process [7] [12] [15] [19] [22].

Different methods are suitable to define different types of system and software development processes. This thesis is focused on safety oriented systems in the software level. Due to the safety aspect of these development processes, specific methods have been defined to describe a safe development process which assures the result complies with all the predefined safety related requirements [9]. For different context, different standards have been specified such as ISO 26262, EN 50 128 and DO-178C [2] [3] [4].

The next step is to define a notation to represent these process specifications in an expressive model. As discussed in Section 3.6, many studies have been conducted to define a powerful notation to model the development processes. These notations which are known as Modelling Languages aim to represent the development process in a more understandable way and help the process engineers and project managers to manage the data and resources more efficiently within a structured framework, apply the change management requirements to the ongoing project and even verify or maintain the process regarding new defined requirements [7]. Some of the Process modelling languages (PML) have been reviewed in section 2.6 and its subsections. More over these general features of the process modelling languages some studies have been done to enrich the syntax and semantics of the PMLs to be more structured and provide the re-usability, exchangeability and Execute-ability of the model or a portion of it [16] [12] [30] [28].

For example, SPEM 2.0 has been defined by the Object Management Group (OMG) that is not-for-profit technology standards consortium industry standards consortium in 2007. SPEM 2.0 provides process engineers and project managers ability to model simple to complex processes. SPEM 2.0 supports process core elements and provide a systematic way to re-use the process elements and more. Even make it possible to convert the models to and from XML files [27].

More specifically regarding the Safety Oriented process, there are some researches to extend the PMLs with Safety process specific elements and routines. As an example, a research based safety-oriented modelling language has been presented in [28].

Execute-ability of the modelling language is another concept in this domain for researchers to work on. Gallina, et al. in [28] present S-TunExSPEM which is an extension to SPEM 2.0 enriched with Execute-ability.
4 Method

This section describes *How* to reach the solution to cope with problems that are mentioned in Section 1.1. Subsection 4.1 shortly discusses the steps need be taken to select a suitable modelling method for this work. The adopted method to select the proper sets of process to investigate is presented in -. How to drive out process elements from selected processes discussed in Subsection 4.3 and finally 4.4 presents how to select an appropriate PML.

4.1 Selecting a Reuse-based Modelling Method

To address the first question from the problem analysis: *What is the proper approach to model a safety oriented process by considering the concept of Reuse?*

To answer this question several modelling approaches in context of Safety Oriented system development have been studied and 3.3 presented a brief description of each one. A comparison assessment took place to select the most appropriate approach which fit the requirements of this thesis. The most important factors in this assessment are:

- The process modelling method should support the concept of *Reuse* in an acceptable level. It should provide the possibility of reusing processes and process elements in a systematic way.
- The process modelling method should be *Flexible* enough to apply tailoring rules and on-going changes to the model.
- An expressive modelling language should exist to represent the modelling approach.

Briefly, below steps followed to select a proper reuse-based modelling approach in context of safety oriented which is presented in Chapter5, *Solution.*

1. Take a modelling approach to assess.
2. Study the approach.
3. Is it supporting the concept of reuse?
   - If Yes: Continue
   - If No: Back to step 1.
4. Is it flexible to apply ongoing changes and tailoring rules?
   - If Yes: Continue
   - If No: Back to step 1.
5. Does any expressive process modelling language exist to represent this approach?
   - If Yes: Select this approach.
   - If No: Back to step 1.
4.2 Selecting a Proper Set of Processes

Similarity is the key concept in the matter of Reuse. Therefore it is essential to select appropriate sets of processes which are expected to have reasonable amount of similarities in case of process elements. It is not necessarily means that processes must come from the same product development life cycle or same project, but also processes can be selected from a great range of choices:

*Intra Domain Processes:*

- Processes which all come from the same domain such as avionic industry ‘s software development processes.
- Processes to be certified under the same standard process.
- A process to be recertified under a different version of the standard process.

*Cross Domain processes:*

- Processes which come from the same domain (e.g. Transportation) but different sub domains (e.g. Avionics, Rail Industries.)

Briefly, below steps followed to select a proper set of processes:

1. Take a set of processes to assess.
2. Review each process.
3. Have them reasonable amount of similarities?
   - If Yes: Select this set of processes.
   - If No: Back to step 1.

4.3 Identification of Process Elements

A process models is a structured representing of the development process elements and their relations. So, it is essential to study the process description and driving out the process elements and their relations from the documented specification. It should be considered how these elements are constructed and what are their dependencies. In this report, we focused on the basic but impartible elements of each development process: Activity, Task, Step, Role, Work product and Guideline.

4.4 Selecting an Appropriate Process Modelling Language (PML)

It is important to implement a clear and expressive model of the development process by an appropriate PML. Foremost, a candidate modelling language to model a reuse based safety oriented process should be rich in syntax and semantics to represent the safety oriented process elements and their relations. It should provide a systematic mechanism to model specific variability relations types as described in Section 3.6.4. Expressiveness, as it is described in Section 3.6 is another important factor, the model
should be easy to understand. It should also provide possibilities to apply ongoing changes to the process and being well tool supported with the possibility of generating machine readable files from the models which is stated in Section 3.6 as Exchangeability. These factors, lead to select SPEM 2.0 modelling language to model the considered approach in this thesis.

Briefly, below steps followed to select a proper process Modelling language (PML).

1. Take a PML to assess.
2. Study the PML specification.
3. Is it supporting the selected approach to model reusable safety oriented process/process elements?  
   If Yes: continue
   If No: back to step 1.
4. Is it Expressive?  
   If Yes: continue
   If No: back to step 1.
5. Does any tool support this PML to implement a machine-readable model? 
   If Yes: select this PML.
   If No: back to step 1.

5 Reuse-based solution: Safety Oriented Process Line Approach

This chapter presents the proposed solution to cope with the problems analysed in Section 1.1, by following the cited method in Chapter 4. More specifically, this chapter presents a systematic approach to enable reuse of process and process elements in the context of safety oriented system development and (re)certification.

As it was mentioned earlier, there is no explicit method exists to model a set of safety oriented development processes. More precisely, there is no systematic approach to model a safety oriented development process by reusing or tailoring an existed process or process elements. There is no concrete approach to reusing a specific part of a predefined process or process elements to overcome a new problem or set of requirements and create a new development process. Likely there are some on-going researches regarding the concept of reusing whole or part of a standard process; 4 Reuse-base methods, called the ProcPT approach, Emmerich’s approach, Process Tailoring using Process Module and finally Process line approach which was reviewed in Section3.3, have been assessed to find out which method provides a better possibility to present a systematic solution for reusing safety critical process elements during the process of developing new systems and/or (re) certification. Below briefly recalls the deficiencies of each investigated modelling approach.

- The ProcPT approach, as it was briefly reviewed in Section 3.3.1, doesn’t modulate the process to reusable sections. So, it is not possible to reuse a specific part of the model in other processes. The elements that are considered in this approach are limited to Activity and Product. The
current version of ProcTP tool is not stable and there is some on-going development effort to implement a more efficient version. No information has been provided about the exchange ability of the implemented model. [19]

- The Emmerich approach was also briefly reviewed in Section 3.3.2, doesn’t define the standard process as a composite structure and it makes it difficult to identify the process units. Therefore, it is not possible to re-use a specific part of the process.
- The major lack of the Process Tailoring that is using Process Module, which summarized in 3.3.3, is that the basic process element Role as the performer of activities is not defined in the method specification. Also, there is no tool support for this method but a prototype tool which proposed by the authors in [12].
- The Process line approach and more specifically its extension safety-oriented process line has been investigated in Section 3.3.4 and an illustrative example of how the relative model is implemented in this thesis is given in Chapter 6.

These Reuse-based methods have been compared by their advantages and disadvantages. The results of this assessment summarized in Table 5.

<table>
<thead>
<tr>
<th>Method aspect</th>
<th>Tailoring &amp; verifying SP</th>
<th>Safety oriented Process- Line approach</th>
<th>Emmerich’s approach</th>
<th>Process PT approach</th>
</tr>
</thead>
<tbody>
<tr>
<td>Modularity/ Structure richness</td>
<td>Partially supported</td>
<td>Highly Structured</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Supporting the aspect of re-use</td>
<td>Supported</td>
<td>Supported</td>
<td>No</td>
<td>Only Deletion</td>
</tr>
<tr>
<td>Representation method/ language</td>
<td>Activity Artefact Graph (AAG)</td>
<td>SPEM2.0</td>
<td>Dynamic eXtension Language (DXL)</td>
<td>GV-Model</td>
</tr>
<tr>
<td>Tool support</td>
<td>Prototype tool developed</td>
<td>Supported by several tools</td>
<td>DOORS</td>
<td>Process PT developed in PROLOG</td>
</tr>
</tbody>
</table>

Table 5. Reuse-based methods comparison

After all, Process Line approach has been chosen to implement the solution model as it is powerful to model the safety critical development process elements and their relationships. It is easy to apply
changes whenever it is needed with a reasonable amount of effort. The Process line approach is easy to understand and its most important capability is providing the possibility to reuse process and process elements in future if needed.

In this chapter, the selected method will be studied in detail. Section 5.1 reviews the selected reused based method: Process Line Approach in details. It presents how the approach lets to model static and dynamic dimensions of processes to create a cross domain process line. Section 5.2 presents in which mechanism the proposed approach provides the possibility to customize and select different portion of the model and more details about the exportation and machine readability features of the model.

More specifically, Section 5.1.1 discusses the Process-line modelling method itself, the language and the corresponding tool which used to implement the model. Section 5.1.2 specifies the work scope. Section 5.1.3 describes how to identify the process elements. Section 5.1.4 presents the work flow of Domain Engineering: organizing the identified process elements in proper categories and defines the relationship between elements to enable a strong systematic reuse possibility. Section 5.1.5 describes the work flow of Process Engineering: how to reuse the pre-identified process elements to define the new single processes.

Figure 22 presents a work flow overview of how to create a Process line.
5.1 Model the Process Line: SPEM 2.0

This section explains the motivation underlying the selection of the SPEM2.0 process modelling language (PML). As it was mentioned in Section 5.4, it is a challenge to select a proper modelling language to provide a clear presentation of the development process models. Specifically, there is no specific language defined to model a safety oriented process line yet. In context of this work, six PMLs called UML4SPM, SPEM 1.1, SPEM 2.0, vSPEM, S-TunExSPEM and Little JILL have been studied. Then four criteria have been considered to perform a comparative evaluation: expressiveness, flexibility to apply ongoing changes to the model and providing good support to model the concept of reuse, exchangeability and tool support. The motivation for the selection of these criteria was given in Section 4.4.
Section 3.6 briefly reviewed these PMLs and the assessment criteria of this comparison. Table 6 presents the result of the comparison. As the SPEM 2.0 is the upgraded version of SPEM 1.1, the later one is not considered in to the comparison.

<table>
<thead>
<tr>
<th>Criteria</th>
<th>UML4SPM</th>
<th>Little-JIL</th>
<th>SPEM 2.0</th>
<th>vSPEM</th>
<th>S-TunExSPEM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Expressiveness</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>Flexibility</td>
<td>1</td>
<td>Not flexible</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>Tool Support</td>
<td>1</td>
<td>1</td>
<td>Not Supported</td>
<td>Not Supported</td>
<td></td>
</tr>
<tr>
<td>Exchangeability</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

*Legend*

1: Partially Supported, 2: Supported

Table 6. Process modelling languages comparison result

Based on the comparison results shown in Table 6, SPEM 2.0 has been selected to model this case study safety oriented process line.

There are different tools that provide support for SPEM 2.0 notation, like MagicDraw UML (SPEM plug-in), Rational Method Composer, Eclipse Process Framework (EPF) and Enterprise Architect [18] but there is no special tool yet to support the SPEM 2.0 capabilities at hundred percent. Among these, EPF provides better facilities to implement the SPEM 2.0 features and possibilities to model the under-consideration safety oriented process line. It provides varieties of process elements as well as facilities to model the variability types which were defined in SPEM 2.0. These variability types make it possible to reuse the existed elements as they are or even customize and reuse them based on the newly defined requirements. EPF has been reviewed briefly in 3.6.7.

5.1.1 Selection of a Proper Set of Processes

By observing the concept of reuse and due to the nature of the safety oriented Process-Line approach, it should be considered that selected safety-oriented processes should have a reasonable amount of similarities. Otherwise creating a process line and reusing process elements become meaningless.

Considering the original objectives of this thesis and assessed sub problems mentioned in 1.1 and due to the focus of this work on safety critical development process, three standards from the field of transportation, called: ISO26262, EN-50128 and DO-178C have been chosen for investigation. These standards come from a common field of transportation and they exhibit some common processes. Nevertheless, as they specified for different domains; each standard presents some variabilities from other standards as well.
5.1.2 Work Scope
This work has been limited to study the design-related activity of the software development processes specification of the selected safety standards. For sake of clarity, it should be noted that this activity is called:

- Software Architectural Design clause 3 in ISO 26262 (Part 6);
- Software design activity in EN50128 (Software design and implementation);
- Software Design Process 4 in DO178C.

The aim of this work is to present a solution to enable the reuse of process elements during the development process life cycle and (re)certification. Analysing the selected portion of process has been done by the author as an example.

This study has been limited to the provision of a company/project-independent interpretation of each above-mentioned activity. The provided report aims to show the usefulness of the approach and has no ambition to represent a skilled and certifiable interpretation of the standard specifications. Presented model is limited to a set of process elements, which are: Role, Work product, Activity, Task, and Step. Also, there has been a brief look to Guidelines and Tools.

5.1.3 Identifying Process Elements
To create a safety oriented process line model, first the process elements should be identified. Due to the limitation of the time and scale of this work, Activity, Task, Step, Work product and Role are the main process elements that are considered; there is also a glance at Guideline and Tool.

It should be mentioned that the activity of identifying process elements should be done by a professional analyser of the related process. This guarantees that identified elements cover the actual process specification.

To clarify how to identify the process elements from the process specification, some examples are provided in what follows:

Activity: body of a process consists of several activities which are structured elements built from smaller components, called tasks which in turn can include some steps, and are related to the responsible Roles and input and output work product.

For example, Software Architectural Design clause in ISO 26262 (Part 6) has been considered as an Activity.

3 Within this report this clause has been considered as an activity.
4 Within this report this process has been considered as an activity.
**Role** is responsible to perform an Activity/Task. For instance, the roles which called *Software Supplier* and *Developer* derived out from the portion of the EN50128 presented in Figure 23. A portion of EN 50128:2001-Section 9.4.

![Figure 23. A portion of EN 50128:2001-Section 9.4](image)

**Tasks** and **Steps** construct an Activity. Performance of the Tasks and Steps results in implement one or a set of work product. Following is an example of a task including steps results from ISO26262 standard process specification and shows how the process specifications transfers to the process elements.

![Figure 24. A portion of ISO26262 standard specification: ISO/FDIS 26262-6:2011, section 7.](image)

This portion of the standard process specification is converted to the following process elements:

**Task:** Ensure that software partitioning implement freedom from interface between software components.

This task consists of following **steps**:

**Step 1:** Ensure the shared resources

**Step 2:** Ensure if the software partitioning is supported by dedicated hardware or equivalent means
Step 3: Consider ASIL for partitioning part of software development

Step 4: Software partitioning verification

Work products are clearly defined in most process specifications as they are the tangible result of the process. Otherwise the outcome of an activity will be considered as a Work product. Figure 25. Work products defined at ISO/FDIS 26262-6:2011 presents an example in which how ISO26262 address a set of output work products at ISO/FDIS 26262-6:2011, section 7. These can be imported to the process model as they defined in the process specification.

Figure 25. Work products defined at ISO/FDIS 26262-6:2011

Tools and guidelines are other elements which can be taken from the process specification and added to the process model. In EPF a task can be considered as a structured element which consists of steps and it is related to its responsible role, the related input and output work products as well as the related guideline which could be a tutorial, suggested technic or any other type of guideline document. But it is not possible to directly add a Tool to this composition. A proposition for this is to define a Guideline of type Tool-Mentor for the related task and then tailoring the related tool to this tool mentor. In Chapter 6, all the process elements related to this thesis will be presented in detail.

5.1.4 Domain Engineering
This subsection explains the work flow of Domain Engineering. Domain Engineering can be considered as the first phase to define the safety oriented process line model. Identifying process elements that has been described in Section 5.1.3 is the first activity in Domain Engineering; since it is a significant activity, a separate section has been assigned to it.

In this phase, all the identified process elements will be classified under two main categories: Base Elements and Variant Elements. Base Elements itself includes three categories, Full-commonality
elements, Partial-commonality elements and Optional elements. These categories conduct the static aspect of the model. The static part builds a pool of elements to be reused to implement the dynamic process specifications. Within this solution, this static aspect of the model called Base Line. Figure 26 presents a high-level view of the Base Line implemented in EPF.

![Figure 26. Base Line created in EPF](image)

While performing the activity of Domain engineering it is very important to scope the tolerant of commonalities and variabilities. It should be clear that in which level of details, the process elements should be investigated. In this work, Steps are the most detailed aspect which has been considered from the Activity decomposition.

The process elements from different processes can be described in different words and even organized in an altered way as different set of activities, tasks and steps; but technically they might have the same objective. So, during the comparing and categorizing the process elements, it should be noticed that different process specification can define the same subject with different words. This is the analyst/profession obligation to study and compare the process elements and understand that they belong to which category.

1. **Base Elements:**

This category can be counted as the process line element depository. Base elements would be organized in 3 different categories as follows:

1. **Full-Commonality(s)** consists of those elements which are completely the same in all processes.
   
   It should be considered that a structured element can be counts as a full commonality when all its included parts are the same in all processes.
   
   As an example, regarding Table 8 the role *Designer* exists in all processes so it can be considered as a full commonality in the Base Line called **BR1: Designer Base**. Table 7 presents an example of how to create a full commonality element based on information given in Table 10.
2. **Partial-Commonality** consists of elements, which are common between some processes, or any part(s) of structured elements, which are common between some processes, the common part itself will be assumed as an individual new element.

Table 8 presents an example of how to create a Partial commonality element.

3. **Optional** includes process specific elements which vary from one process to another. Table 9 presents an example of how to create an Optional element based on Table 8 information.
Table 9. An example of how to create an Optional element

II. Variants:

Each element under Variants category counts as a variable element which can add or inherit attributes to or from base elements or even replace an element with a new element or attributes to the base elements. These interactions between elements are represented by using the variability types Contributes, Replace, Extend and Extend-Replace which were discussed in Section 3.6.4. These are enable the mechanism to reuse the Base elements.

4. Variants Includes all process elements which vary from the base element with a special variability type. These elements in relation to Partial-commonality and optional elements plus Full commonality elements will be reused at the dynamic part of the model to create the Delivery Processes.

Tables 6-8 show an example of how to domain engineering and categorize the identified process elements as Full-Commonality, Partial-Commonality, Optional, Variants and their relationships to model the Base Line. While creating and categorising the Variants it is so important to connect the element with a correct variability type to the related Base element. Following, briefly recalls each variability type and some possible cases which have been recognized under this thesis.

- Contributes:

The Contribute variability type is used when the variant element should be related to the Base element in an additive way. As a result, the Base element will be logically affected by the additional properties of the variant elements. Figure 27 illustrates an example how Contributes is effecting contributor and the Base elements.
Figure 27. SPEM 2.0 diagram shows a contribution between Base and Variant roles.

An instance case of using Contributes is when the Base line element should be updated for probable future reuse by the latest upgraded version of the variant element in an additive way. At the upper diagram, the role “Safety Analyst_Base” which is responsible for task “Develop Safety requirements specification” is contributed with the role “Analyst Base”. The result of this contribution illustrated at the lower diagram: The role “Safety Analyst” has been removed and Analyst Base enriched with the task and related work products has been formerly assigned to the Safety Analyst.

- Extend

As it described earlier in Section 3.6.4, Extend variably type makes it possible for variant element to inherit all Base element attribute and extends it with its own attributes without changing the Base element. Figure 28 illustrates an example of Extend usage.
Figure 28. SPEM 2.0 diagram illustrate an example of Extend variability type

- **Replace**

As it was described in Section 3.6.4, the Replace variability type makes it possible to replace the Base element with a completely new element or an element with major varieties from the base element. One of the advantages of Replace that we have discovered is that an Activity or Work product can be replaced with a variant element, which is done with a different method or its result presented in a different template/pattern, other than the base element.

Figure 29 presents an example of using replace relation between work products.
Extend-Replace

Extend-Replace is a composition of Extend and Replace variability types. It enables variant element to not only inherit the base element’s attributes but to extend it with its own additional properties. The Extend-Replace makes it possible to manipulate some attributes of the base elements without the need to replace it completely with a new element. It means only those attributes which are needed will be redefined and others will stay the same. Figure 30 shows an example of using Extend-Replace.
Figure 30. SPEM 2.0 diagram illustrate an example of Extend-Replace variability type taken from [5]
<table>
<thead>
<tr>
<th>Process</th>
<th>Process Elements</th>
<th>Task &amp; Steps</th>
<th>Work products</th>
</tr>
</thead>
</table>
| Process 1 | P1R1: Designer | P1T1: Architectural design  
  P1T1S1: High level design  
  P1T1S2: Detailed component design | P1W1: System Architecture & design specification:  
  - High level design specification  
  - Low level component specification |
|         | P1R2: Safety Engineer | P1T2: Safety Analysis  
  Step1: Safety requirement analysis  
  Step2: Provide Safety cases | P1W2: Safety analysis specification |
| Process 2 | P2R1: Designer | P2T1: Design  
  P2T1S1: Architecture Design  
  P2T1S2 Architecture Design Validation | P2W1: System design specification |
| Process 3 | P3R1: Designer | P3T1: System design  
  Step1: design specification | P3W1: Design specification |
|         | P3R2: Developer | P3T2: Implementation | P3W2: Codes |

![Domain Engineering Diagram](image)

<table>
<thead>
<tr>
<th>Base Line</th>
<th>Role</th>
<th>Task &amp; Steps</th>
<th>Work products</th>
</tr>
</thead>
</table>
| Base Elements | Full Commonality | BR1: Designer Base | BT1: Architectural design Base  
  BT1S1: High level design Base | BW1: Design specification Base |
|            | Partial Commonality | BR2: Safety Engineer Base  
  BT2: Safety Analysis Base  
  BT2S1: Safety requirement analysis  
  BT2S2: Provide Safety cases | BW2: Safety analysis specification Base  
  BW3: Codes Base |
|            | Optional | BR3: Developer Base | BT3: Implementation Base |
| Variable Elements | Variables | | P3W2: Codes in Replace relation with BW3 |

Table 10. Creating a Base Line as a result of Domain Engineering
Figure 31 illustrates a high-level view of a Base Line implemented in EPF. Plug-ins and Content packages in EPF provide a way to organize the process line. It means it could be defined in a different way depend on many factors such as the size of the process line, its level of details etc. In Chapter 6 a safety oriented process line model has been implemented as a case study which tries to give a clear understanding of how the process line model can be implemented by consuming real data.

![Figure 31. Base Line implemented in EPF](image)

### 5.1.5 Process Engineering

Dynamic part of the process model consists of sets of single process that is created by reusing the predefined process elements from Full Commonality and Variants categories which are related to the Partial commonality and Optional elements. Contrary to the domain engineering; sequence of the activities are important features in Process engineering. In EPF by setting the proper Predecessors to each task makes it possible to define the priority of the tasks.

Each process will be created by reusing predefined process elements that come one after another in a specific order which their performance result full fill the process specification requirements. In this work, these dynamic single processes called Delivery Process and is implemented in EPF under Delivery process plug in as Delivery process content. Figure 32 presents a simple example of a delivery process work breakdown structure. It is noticeable that depend on the variability relation type between the variant element and the base element, either one of them can show up in the delivery process work breakdown. For example, if the variability type contributes to the Base element, it is the Base element which shows up in the work break down structure. On the other hand, if the variability type is Replaced, Extended or Extend-Replaced, the Variant element shows up in the work break down structure. Please see Section 6.1.2 to recall how each variability type takes effect on associated process elements.
EPF makes it possible to draw the activity diagram for each delivery process based on its work breakdown structure. This is a simple activity diagram which can be enriched by a limited set of tools provided by EPF. Figure 33 shows the activity diagram of the Delivery Process1 which is presented in Figure 32.

5.2 Export the Model: Machine Readability

As it is discussed earlier in Section 3.6, it is important that the selected process modelling language be exchangeable and be supported by a powerful tool which provides good facilities to export machine readable models. On the other hand, it is important to have this possibility to select a customized portion of the model that needs to be exported. Following sections describe how EPF facilities used in this thesis to export machine readable models.
5.2.1 Customized Configuration

EPF defined a concept called Configuration, which enables to select and save customized portion of different parts of a model, for example, a single process, delivery processes or any other composition of the model’s units. Yet, it should be observed that the smallest units that can be selected, are included elements in each method contents.

Figure 34 shows how a configuration defined and customized in EPF.

![Figure 34. A customized configuration in EPF](image)

As it is illustrated in Figure 34, a Configuration modelled in EPF, specifically presents the selected packages and their relationship from the whole process line model including all base elements, process 1 variants and Delivery process 1. This customized configuration can be exported and reused in other processes or as input to other machine for further processes.

5.2.2 Export the Model as a XML File

Automation and exchangeability are important points in context of process modelling. Being able to automate the modelling process by importing proper data, changing and maintaining the model systematically, would result in saving a big amount of time and cost. But this is not possible without
having a machine-readable version of the model. As it pictured in Figure 35, EPF makes it possible to export the model as a whole Library, selected Configuration, Method plug-ins, Microsoft Projects and as a XML file.

![Figure 35. EPF provides possibility to export the library in different formats](image)

An obvious advantage of EPF, which empowers the model exchange ability, is to be able to export the model as a XML file. Being able to export the whole or a portion of a model in a known format such as XML, makes it possible to use the exported data as an input of other machines to process and fulfil the related requirements.

### 6 Implementation: A Case Study

In this case study, the proposed solution has been applied to a real set of data to verify its capabilities to address the thesis objectives which were mentioned in Chapter 1.

The “Safety Critical Systems Development Baseline_Design” model that is implemented in EPF within case study to prove the usage of the propose approach to enable the reuse of processes and process elements is attached to this work.

#### 6.1 Identifying the process elements

In this scope of work, author studied the software development portion of three safety oriented standard specifications: ISO 26262, EN50128, DO178C. More precisely, activities with concern of software design have been investigated. A detailed list of identified process elements is available in Appendix 1.

Section 5.1.3 described an example of how to identify process elements from the safety standard prescriptive specification. It is obvious that analysing the standard specification and modulate it to the
process elements should be done by a professional standard process analyst. What this work presents an example to clarify how safety oriented process line approach is adopted to accomplish this case study.

6.2 Process-related Domain Engineering

After identifying the process and the process elements from the safety standard specifications, it is time to *Domain engineering*. Elements that identified in Section 6.1 have been assorted to two main categories; the Base element includes Full commonality, Partial commonality, Optional and Variants categories. Following paragraphs recall the attributes of each category and describes how the described solution in Chapter 5 applied to this case study.

Figure 36, shows an abstract view of the modelled Base line includes Design_Base and Design_Variants.

![Figure 36. Design_Base, a portion of the EPF model](image)

The Design_Base itself includes:

1. **Full Commonality**: These elements are the same in all investigated safety standard specifications. It is possible to re-use these elements, as they are to create the customized delivery process. Figure 37 shows the modelled content Design_Base_Commonality_Full. In this case the role, Designer_Base is considered as full commonality.
Partial Commonality: To re-use these elements to create the delivery processes, it is needed to enrich the elements in an additive way. These elements contain the common part of partial commonalities. Figure 38 shows the related EPF content, Design_Base_Commonality_Partial.

As an example, to show how this approach helps to specifically model safety oriented processes, in [17] an additional partial commonality category has been defined that specifically contains the safety oriented partially common elements. Figure 39 shows Design_Base_Commonality_Partial_Safety category and the involved elements.
3. **Optional**: It is optional to reuse these elements to create a customized process. These can be replaced by an empty element as well. As described in section 5.1.4, an optional element includes the standard specific part of partially common original elements. Figure 40 shows the implemented: Design_Base_Optional.
The second Baseline main category is Design_Variants. This category includes reusable standard-specific element which varies from the Base elements in a specific contribution type. Figure 41 presents an abstract view of Design Variants categories.

![Figure 41. Design_Variants, a portion of the EPF model](image)

These elements are in different type of relations with the Design_Base elements. In 5.1.4, it has been described how to identify and assign the proper variability type between these variants and the Design_Base elements. Figure 42 presents an example of how to model a variant element and assign a proper variability type to it.
Appendix 1 contains a detailed mapping of the modelled elements to an Excel table.

### 6.3 Process Engineering

This section represents how to implement the dynamic phase of the process-line approach, *Process engineering*, in EPF. The concept of Process Engineering which has been described in 5.1.5. Concisely, whereas Domain engineering results to implement a structured static pool of elements, Process engineering leads to implement the dynamic processes by reusing those elements. EPF enables to select from the Baseline elements in order to reuse them and create a new customized development process.

First step to create a development process is to select the involved activity from the static skeleton of the Base line. Figure 43 shows a set of activities that are selected to create the DO178B _Design_Delivey process. The view of the delivery process at the left column called *Work breakdown structure view*. 

![Figure 42. How to assign a proper variability type to a task in EPF.](image-url)
After selecting the proper elements, to create a dynamic process, activities are needed to be sorted out by a specific sequence.

EPF made it possible to choose predecessors for each activity. Setting the predecessors results to have each activity in a correct place according to the process timeline. Figure 44 shows an example of how the predecessors have been set for the DO1778B Design Delivery process in this case study.

EPF also has an ability to create a basic sequence diagram based on which predecessors has been set. Then it provides a set of UML tools so we could make the diagram more advanced and organized. Figure 45 shows a sequence diagram created based on the DO178B Design Delivery Process activities and the related predecessors.
As it was mentioned in 5.2, EPF allows to export the whole or a portion of the implemented model in different formats. In this case study, we have focused on the Library Configuration and XML formats. Figure 46 shows different formats which are possible to export a EPF model in.
6.4.1 Library Configuration

To export a Library Configuration, one needs to be created first. As it was described in section 5.2.1 EPF allows to select a set of modelled process elements and delivery processes to create a customized unit called Library Configuration. Figure 47 shows how a set of categories have been selected to create the DO178B Library Configuration. This creates the possibility to export a required portion of the model instead of the whole modelled process line and its details.

Figure 46. Different formats to export a EPF model

Figure 47. DO178B Library Configuration implemented in EPF
### 6.4.2 XML

EPF also permits to export the model on its whole or a selected part of it in XML format. As Figure 48 presents below, it is possible to select one or several method plugin(s), a specific method configuration or the entire method library to export as a XML file.

![Select export type](image)

**Figure 48.** Export a portion or entire model as a XML file from EPF

With concern of exchangeability, sharing and presence of meta information and properties, XML is a common format to store complicated data in a structure form. XML human readable and it is possible to open a XML file in an Internet browser or a simple text editor. Also, the XML creates a specific tag for each element in the data file which make the exported file structured and exchangeable. This also make it easy to trace the relations between desired element or find a specific element within the exported file.

For an instance, as it is shown in Figure 49, the method plugins Design_Baseline and Design_Delivery process_DO178B have been selected to be exported as XML file.

![Select method plugins](image)

**Figure 49.** Selecting one or more plug ins to be exported as XML

The XML snippets given in Figure 50, shows the summarized structure of the exported file. As mentioned above each element in selected portion of the model got its own tag which make it possible to refer to the tag or follow up its relation to the other elements.
Figure 50. Briefed XML file exported from an EPF model

This is a structured detailed presentation of what has been modelled in EPF as Figure 51 exposes it below. XML text above, gives the information that the Task called 01_Design description from the method content DO178B_Design_Variants from the main category Design_variants is in a Contributes variability type with the task called Software architectural design_Base from the Design_Base Commonality_Partial belongs to the Design_Base main category.

Since XML assign a specific tag to each element, it is possible to search and locate any specific element and its relation to the other elements effectively.
6.5 Case Study Evaluation

In the presented case study, the Safety-Oriented Process Line approach has been adopted and applied to selected portions of safety standards ISO 26262, EN50128 and DO-178C in order to implement the “Design_Process Line” model in SPEM 2.0/Eclipse Process Composer (EPF). The findings of this case study are briefly listed as follow:

- Systematic reuse of modelled process elements has been enabled by adopting Safety-Oriented Process Line approach in SPEM 2.0/EPF in an expressive and organized way.
- The amount of commonality between the selected set of processes has been the key feature to enable the reuse of the process elements.
• Identifying the process elements in compliance with the relative safety standard has been an important activity to implement such model. A clear and traceable relation between the identified process elements and the safety standard specification has to be stated. This activity should be done by a professional safety standard analyst.

• Systematic use of the notation of Full-Commonalities, Partial-Commonalities and Variants helped to define the commonalities and variabilities of the identified process elements.

• Within this case study, examples of Contribute, Replace and Extend variability types have been identified and modelled according to the investigated set of data.

• SPEM2.0/EPF has provided a reasonable support to implement the case study by adopting the Safety-Oriented Process Line approach. It was possible to model:
  - Static and dynamic aspects of each process.
  - Different types of relationships (Variability types) between process elements that helped modelling the Line process and Variables.
  - Both atomic and structured elements.

All these have enabled the systematic reuse of modelled elements to define new processes. Nevertheless, some limitations have been remarked as well:

  - The architecture of the model has to be designed considering the EPF predefined structure which has not been flexible.

  - Even though SPEM2.0 provided possibilities to model different types of process elements, yet no attribute/feature has been defined to specify the safety-oriented process elements. In SPEM2.0/EPF it has been possible to add textual explanations to the element to clarify any specific attribute.

7 Conclusion

This chapter summarises this thesis report and its achievements. Also, suggests a few potential leads for the future work in this subject.

7.1 Summary

This thesis has adopted the safety oriented process line approach to model safety-oriented development processes in which the modelled processes can be reused in process of (re-)certification. Thesis studied and compared several process modelling languages and picked up SPEM 2.0 as a proper modelling language to represent the safety oriented process line approach. This report extended the existed works by considering additional variability types and process elements.

To aim thesis objectives, four process modelling approaches have been investigated; ProcPT, Managing Standards Compliance, Tailoring & verifying software process and Process line approach. To find the
best approach that enables modelling re-useable safety oriented process, following aspects have been considered: modularity, supporting the aspect of re-use, representative language and tool support; Safety Oriented Process line approach has been selected to conduct this thesis. The most important feature of the adopted approach is definition of variability types which enables modelling the re-useable safety oriented processes.

To model a reusable safety oriented process line, several safety-oriented standard development process specifications have been reviewed to select a proper set of safety oriented processes; according to the concept of Reuse, it was appropriate to choose a set of specifications with the most common features while having their own specific properties. This investigation ended up with selecting a set of safety standard specifications from the transportation domain called: DO178C, ISO26262 and EN50128.

The selected set of the safety standard specifications have been analysed to identify the clear development processes. Identified processes and their elements have been categorized to the Base-line and Variants main categories. Then, proper variability types have been applied to the Variant elements according to safety oriented process line approach policies. This phase which called Domain engineering, creates a static view of consists of reusable elements. Next phase is to reuse the elements to create new, customized delivery processes. This phase called Process Engineering.

To model the re-useable safety oriented process by following the safety oriented process line approach, six Process Modelling Languages have been investigated. Little-JIL, SPEM 1.1, UML4SPM, SPEM 2.0, vSPEM and S-TunExSPEM have been studied and compared. With reference to their expressiveness, flexibility, tool support and exchangeability. The other important feature to consider was how the PML could support the process line approach and enable the concept of process reusability. Investigations results in selecting SPEM2.0 and its supportive tool EPF to implement this thesis case study.

In Chapter 6, an illustrative example has been implemented to verify the usage of the proposed solution to model reusable process and process elements in context of safety oriented system development and (re) certification. This proposed approach has been also adopted and extended in [17] and [31] to support those researches objectives by considering different process elements and other model design architecture that enabled some other facilities of SPEM2.0/EPF to model reusa ble safety oriented process and process elements.

7.2 Future work

Below are few leads which could potentially be considered as future works.

Based on the approach presented in this thesis, different architectures can be designed to model the reusable safety oriented processes within EPF by considering different safety critical process elements
or different categorized structure. The model that has been implemented to support [17] is an example of different architecture with focus on Tools.

The other path to extend and customize this approach is to apply it to other safety-oriented inter domain or cross domain set of processes than transportation. The solution is valid for different sets of processes that have several aspects in common but also have their own specific properties.

There is space for more research to gain benefits of SPEM2.0/EPF exchange ability that enables to export the entire or a portion of the reusable model as XML files in order to automate the transformation of data from the model as an input to other machines.

Research, examine and extend the existing tools that support the Safety Oriented Process-line Approach and extend the model by considering other safety oriented process elements and variability types is another path that can be covered by further implementing.

References


Appendix 1

The Excel file contains all identified safety critical standard specification elements is attached to this report. An image of this Excel file is presented below.
<table>
<thead>
<tr>
<th>P3.1. Software design</th>
</tr>
</thead>
<tbody>
<tr>
<td>P3.1.1. Design description</td>
</tr>
<tr>
<td>P3.1.1.1. Software architecture description</td>
</tr>
<tr>
<td>P3.1.1.2. Low level requirement specification</td>
</tr>
<tr>
<td>P3.1.2. Software components specification</td>
</tr>
<tr>
<td>P3.1.2.1. Software components identification</td>
</tr>
<tr>
<td>P3.1.2.2. Software component categorization</td>
</tr>
<tr>
<td>P3.1.2.3. Software component safety integrity level determination</td>
</tr>
<tr>
<td>P3.1.3. Design description verification</td>
</tr>
<tr>
<td>P3.1.4. Safety specification</td>
</tr>
<tr>
<td>P3.1.4.1. Safety analysis</td>
</tr>
<tr>
<td>P3.1.4.2. System responses to failure specification and verification</td>
</tr>
<tr>
<td>P3.1.5. Monitoring the safety-related requirements</td>
</tr>
<tr>
<td>P3.1.5.1. Monitoring control flow</td>
</tr>
<tr>
<td>P3.1.5.2. Monitoring data flow</td>
</tr>
<tr>
<td>P3.1.5.3. Design test modifiable software</td>
</tr>
<tr>
<td>P3.1.5.4. Insensitive protection of non-modifiable components from modifiable components</td>
</tr>
<tr>
<td>P3.1.5.5. If the protection is provided by software, it should be designed and verified at the same software level as the non-modifiable software</td>
</tr>
<tr>
<td>P3.1.5.6. The method provided to change the modifiable component should be shown to be the only means by which the modifiable component can be changed (Guideline?)</td>
</tr>
<tr>
<td>P3.1.7. Verification</td>
</tr>
</tbody>
</table>

P3.1.5.1. Software architecture

P3.1.5.2. Low level requirements
[P2] EN50128
P2A1. Software Design
P2A2. Requirement Specification
P2A2.1. Software Requirements Specification
P2A4. System Requirement Specification
P2A5. System Architecture Description
P2A6. Software Quality Assurance Plan

P2T1. Software Architecture Specification
P2T1.S1. Software architectural specification by considering the Software Requirements Specification at the required software safety integrity level.
P2T1.2. Hardware-Software interface specification
P2T1.3. Software components specification
P2T1.3.1. Identify and describe software components
P2T1.3.2. Categorizing components
P2T1.3.3. Identifying if component are validated if so, their validation conditions
P2T1.3.4. Identifying SILs of the components
P2T1.3.5. Specifying COTS components if applicable by considering SIL
P2T1.4. COTS software identification and verification considering related safety integrity level
P2T1.5. Previously developed software identification and suitability justification considering the required software safety integrity level
P2T1.6. Safety specification
P2T1.7. Identifying software development strategy considering SIL
P2T1.8. Verification
P2T1.8.1. Justifying that the selected measures and techniques satisfies the Software Requirements Specification at the required SIL

Software Architecture Specification
Appendix 2

*Process Line* _Design Method Library_ model, which is implemented in EPF is attached to this report.

Appendix 3

The exported XML file form the selected parts of the implemented case study, Section 6.4.2.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<uma:MethodLibrary xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:uma="http://www.eclipse.org/epf/uma/1.0.6" name="Desktop" briefDescription=""
id="_3sAFIYonEeKthuzKkyyAbg" orderingGuide="" presentationName="" suppressed="false"
authors="" changeDescription="" version="" tool="epf=1.5.0"/>
```
<MethodElementProperty name="library_synFree" value="true"/>

<MethodElementProperty value="0"/>

<MethodElementProperty value="_S3x58IooEeKthuzKkyyAbg"/>

<MethodElementProperty value="Design_Baseline"/>

<MethodElementProperty value="_NWjbIoqEeKthuzKkyyAbg"/>

<MethodElementProperty value="DO178B_Design_Delivey process"/>

<MethodElementProperty value="_Rr2AYIoqEeKthuzKkyyAbg"/>

<MethodElementProperty value="ISO26262_Design_Delivey process"/>

<MethodElementProperty value="_UJWkoioqEeKthuzKkyyAbg"/>

<MethodElementProperty value="EN50128_Design_Delivey process"/>

<MethodPlugin name="Design_Baseline" briefDescription="" id="_S3x58IooEeKthuzKkyyAbg" orderingGuide="" presentationName="" suppressed="false" authors="Shaghayegh Kashiyarandi" changeDate="2013-03-11T10:19:10" changeDescription="" version="" supporting="false" userChangeable="true"/>

<MethodElementProperty name="plugin_synFree" value="true"/>

<MethodPackage xsi:type="uma:ContentCategoryPackage" name="ContentCategories" id="_sPXlJt3EeapT9Qu_oq2uA">

  <ContentCategory xsi:type="uma:Tool" name="Range checks of input and output data" briefDescription="" id="_iA4GUODAEeOMvsZxx52zg" orderingGuide="" presentationName="Range checks of input and output data" suppressed="false" isAbstract="false" variabilityType="na">

    <MethodElementProperty name="me_edited" value="true"/>

    <ToolMentor>_WzKQsODAEeOMvsZxx52zg</ToolMentor>

  </ContentCategory>

  <ContentCategory xsi:type="uma:Tool" name="Plausibility checka" briefDescription="" id="_mZigaODAEeOMvsZxxxy52zg" orderingGuide="" presentationName="Plausibility checka" suppressed="false" isAbstract="false" variabilityType="na">

    <MethodElementProperty name="me_edited" value="true"/>

    <ToolMentor>_WzKQsODAEeOMvsZxx52zg</ToolMentor>

  </ContentCategory>

  <ContentCategory xsi:type="uma:Tool" name="Detection of data errorsb" briefDescription="" id="_pmZtQODAEeOMvsZxxxy52zg" orderingGuide="" presentationName="Detection of data errorsb" suppressed="false" isAbstract="false" variabilityType="na">

    <MethodElementProperty name="me_edited" value="true"/>

    <ToolMentor>_WzKQsODAEeOMvsZxx52zg</ToolMentor>

  </ContentCategory>

</MethodPackage>
<MethodElementProperty name="me_edited" value="true"/>

<ToolMentor>_WzKQsODAEeOMvsZxxxy52zg</ToolMentor>

</ContentCategory>

<ContentCategory xsi:type="uma:Tool" name="External monitoring facility" briefDescription="" id="_sqM1cODAEeOMvsZxxy52zg" orderingGuide="" presentationName="External monitoring facility" suppressed="false" isAbstract="false" variabilityType="na">

<MethodElementProperty name="me_edited" value="true"/>

<ToolMentor>_WzKQsODAEeOMvsZxxxy52zg</ToolMentor>

</ContentCategory>

<ContentCategory xsi:type="uma:Tool" name="Control flow monitoring" briefDescription="" id="_vo0isODAEeOMvsZxxy52zg" orderingGuide="" presentationName="Control flow monitoring" suppressed="false" isAbstract="false" variabilityType="na">

<MethodElementProperty name="me_edited" value="true"/>

<ToolMentor>_WzKQsODAEeOMvsZxxxy52zg</ToolMentor>

</ContentCategory>

<ContentCategory xsi:type="uma:Tool" name="Diverse software design" briefDescription="" id="_yqpU8ODAEeOMvsZxxy52zg" orderingGuide="" presentationName="Diverse software design" suppressed="false" isAbstract="false" variabilityType="na">

<MethodElementProperty name="me_edited" value="true"/>

<ToolMentor>_WzKQsODAEeOMvsZxxxy52zg</ToolMentor>

</ContentCategory>

</MethodPackage>

<MethodPackage xsi:type="uma:ContentPackage" name="Design_Base" briefDescription="" id="_fD-xEllooEeKthuzKkyyAbg" orderingGuide="" presentationName="Design_Base" suppressed="false" global="false">

<MethodPackage xsi:type="uma:ContentPackage" name="Design_Base_Commonality_Partial" briefDescription="" id="_wjbDcIooEeKthuzKkyyAbg" orderingGuide="" presentationName="Design_Base_Commonality_Partial" suppressed="false" global="false">

<ContentElement xsi:type="uma:Task" name="Software architectural design _Base" briefDescription="" id="_OD0UUiorEeKthuzKkyyAbg" orderingGuide="" presentationName="Software architectural design _Base" suppressed="false" isAbstract="false" variabilityType="na">

<Presentation xsi:type="uma:TaskDescription" name="_OD0UUiorEeKthuzKkyyAbg" briefDescription="" id="_OMwde7IUwNw_FHp2GGg" orderingGuide="" presentationName="" suppressed="false" authors="" changeDescription="" version="" externalId=""/>
<ContentElement name="Safety specification_Base" id="_wSrxYIorEeKthuzKkyyAbg" presentationName="Safety specification_Base" suppressed="false" isAbstract="false" variabilityType="na">

<Section name="Safety analysis" briefDescription="" id="_zX2LglorEeKthuzKkyyAbg" orderingGuide="" presentationName="" suppressed="false" sectionName="" variabilityType="na">

<Description/>

</Section>

<Alternatives/>

</Section>

<Alternatives/>

</Section>

</ContentElement>

<ContentElement name="Verification_Base" id="_2Ar2AIorEeKthuzKkyyAbg" presentationName="Verification_Base" suppressed="false" variabilityType="na">

<Section name="Software architectural design verification according to related standard" briefDescription="" id="_sNpvQI1YEeK95dFtIcj1-g" orderingGuide="" presentationName="" suppressed="false" sectionName="" variabilityType="na">

<Description/>

</Section>

<Alternatives/>

</Section>

<Alternatives/>

</Section>

</ContentElement>
<ContentElement xsi:type="uma:Task" name="Contracts definition_Base" briefDescription="" id="_vKMkIosEeKthuzKkyyAbg" orderingGuide="" presentationName="Contracts definition_Base" suppressed="false" isAbstract="false" variabilityType="na"/>

<ContentElement xsi:type="uma:Task" name="Certificate preparation_Base" briefDescription="" id="_Gyi0MIotEeKthuzKkyyAbg" orderingGuide="" presentationName="Certificate preparation_Base" suppressed="false" isAbstract="false" variabilityType="na"/>

<ContentElement xsi:type="uma:Artifact" name="Architectural design specification_Base" briefDescription="" id="_edCTkIotEeKthuzKkyyAbg" orderingGuide="" presentationName="Architectural design specification_Base" suppressed="false" isAbstract="false" variabilityType="na"/>

<Fulfill>_xdYbMIoxEeKthuzKkyyAbg</Fulfill>

<Fulfill>_oWPX4IouEeKthuzKkyyAbg</Fulfill>

<Fulfill>_YkKmQlouEeKthuzKkyyAbg</Fulfill>

</ContentElement>

<ContentElement xsi:type="uma:Artifact" name="System configuration_Base" briefDescription="This is a part of Architectural design specification_Base" id="_YkKmQlouEeKthuzKkyyAbg" orderingGuide="" presentationName="System configuration_Base" suppressed="false" isAbstract="true" variabilityType="na"/>

<ContentElement xsi:type="uma:Artifact" name="Single contract specification_Base" briefDescription="This is a part of Architectural design specification_Base" id="_oWPX4IouEeKthuzKkyyAbg" orderingGuide="" presentationName="Single contract specification_Base" suppressed="false" isAbstract="true" variabilityType="na"/>

<ContentElement xsi:type="uma:Artifact" name="Single component specification" briefDescription="This is a part of Architectural design specification_Base" id="_xdYbMIoxEeKthuzKkyyAbg" orderingGuide="" presentationName="Single component specification" suppressed="false" isAbstract="true" variabilityType="na"/>

<ContentElement xsi:type="uma:Task" name="Argumentation_Base" briefDescription="" id="_rorJMI_lEeKrTIQgX7A_CQ" orderingGuide="" presentationName="Argumentation_Base" suppressed="false" isAbstract="false" variabilityType="na"/>

<ContentElement xsi:type="uma:Artifact" name="Safety case fragment_Base" briefDescription="" id="_BRGNsI_nEeKrTIQgX7A_CQ" orderingGuide="" presentationName="Safety case fragment_Base" suppressed="false" isAbstract="false" variabilityType="na"/>
<MethodPackage xsi:type="uma:ContentPackage" name="Design_Base_Optional" briefDescription="" id="_4UZscIooEeKthuzKkyyAbg" orderingGuide="" presentationName="Design_Base_Optional" suppressed="false" global="false">
  <MethodPackage xsi:type="uma:ContentPackage" name="Design_Base_Optional_DO178B" briefDescription="Base optional work products related to DO 178B" id="_qDx68IoyEeKthuzKkyyAbg" orderingGuide="" presentationName="Design_Base_Optional_DO178B" suppressed="false" global="false">
    <ContentElement xsi:type="uma:Artifact" name="Software Requirements Data_Base_Optional_Input" briefDescription="" id="_U4PU0IozEeKthuzKkyyAbg" orderingGuide="" presentationName="Software Requirements Data_Base_Optional_Input" suppressed="false" isAbstract="false" variabilityType="na"/>
    <ContentElement xsi:type="uma:Artifact" name="Software Development Plan_Base_Optional_Input" briefDescription="" id="_aZFHMIozEeKthuzKkyyAbg" orderingGuide="" presentationName="Software Development Plan_Base_Optional_Input" suppressed="false" isAbstract="false" variabilityType="na"/>
    <ContentElement xsi:type="uma:Artifact" name="Software Design Standards_Base_Optional_Input" briefDescription="" id="_eyvPEIozEeKthuzKkyyAbg" orderingGuide="" presentationName="Software Design Standards_Base_Optional_Input" suppressed="false" isAbstract="false" variabilityType="na"/>
    <ContentElement xsi:type="uma:Artifact" name="Software architecture_Base_Optional_Output" briefDescription="" id="_3K4eolozEeKthuzKkyyAbg" orderingGuide="" presentationName="Software architecture_Base_Optional_Output" suppressed="false" isAbstract="true" variabilityType="na"/>
    <ContentElement xsi:type="uma:Artifact" name="Low level requirements _Base_Optional_Output" briefDescription="" id="_PRHYIoozEeKthuzKkyyAbg" orderingGuide="" presentationName="Low level requirements_Base_Optional_Output" suppressed="false" isAbstract="true" variabilityType="na"/>
    <ContentElement xsi:type="uma:Task" name="Design description verification_Base_Optional" briefDescription="According to: DO178B, section 5.2.2, b" id="_RwLiQIo0EeKthuzKkyyAbg" orderingGuide="" presentationName="Design description verification_Base_Optional" suppressed="false" isAbstract="false" variabilityType="na"/>
    <ContentElement xsi:type="uma:Task" name="Monitoring the safety related requirements_Base_Optional" briefDescription="According to: DO178B, section 5.2.2, d" id="_aZFHMIozEeKthuzKkyyAbg" orderingGuide="" presentationName="Monitoring the safety related requirements_Base_Optional" suppressed="false" isAbstract="false" variabilityType="na"/>
  </MethodPackage>
</MethodPackage>
<Presentation xsi:type="uma:TaskDescription" name="_gak6Ilo0EeKthuzKkyyAbg" briefDescription="" id="_X_Vug3AszOI8CwMg-xnHuw" orderingGuide="" presentationName="" suppressed="false" variabilityType="na">
  <MainDescription></MainDescription>
  <KeyConsiderations></KeyConsiderations>
  <Section name="Monitoring control flow" briefDescription=""
    id="_n7EQEIo0EeKthuzKkyyAbg" orderingGuide="" presentationName="" suppressed="false"
    sectionName="" variabilityType="na">
    <Description></Description>
  </Section>
  <Section name="Monitoring data flow" briefDescription=""
    id="_onLIIIo0EeKthuzKkyyAbg" orderingGuide="" presentationName="" suppressed="false"
    sectionName="" variabilityType="na">
    <Description></Description>
  </Section>
  <Alternatives></Alternatives>
  <Purpose></Purpose>
</Presentation>

<ContentElement xsi:type="uma:Task" name="Design user modifiable software _Base_Optional" briefDescription="According to: DO178B, section 5.2.3"
  id="_uUkIwIo0EeKthuzKkyyAbg" orderingGuide="" presentationName="Design user modifiable software _Base_Optional" suppressed="false" isAbstract="false" variabilityType="na"/>

</MethodPackage>

<MethodPackage xsi:type="uma:ContentPackage" name="Design_Base_Optional_ISO26262"
  briefDescription="" id="_y-SX0Io0EeKthuzKkyyAbg" orderingGuide="" presentationName="Design_Base_Optional_ISO26262" suppressed="false" global="false">
  <ContentElement xsi:type="uma:Artifact" name="Design and coding guidelines for modeling and programming languages_Base_Optional_Input" briefDescription=""
    id="_endhQIo1EeKthuzKkyyAbg" orderingGuide="" presentationName="Design and coding guidelines for modeling and programming languages_Base_Optional_Input" suppressed="false" isAbstract="false" variabilityType="na"/>
<ContentElement xsi:type="uma:Artifact" name="Guidelines for the application of methods_Base_Optional" briefDescription="" id="_LkvclIo5EeKthuzKkyyAbg" orderingGuide="" presentationName="Guidelines for the application of methods_Base_Optional" suppressed="false" isAbstract="false" variabilityType="na"/>

<ContentElement xsi:type="uma:Artifact" name="hardware-software interface specification_Base_Optional_Input" briefDescription="" id="_vRqOIo1EeKthuzKkyyAbg" orderingGuide="" presentationName="hardware-software interface specification_Base_Optional_Input" suppressed="false" isAbstract="false" variabilityType="na"/>

<ContentElement xsi:type="uma:Artifact" name="Qualified software components available_Base_Optional_Input" briefDescription="" id="-_f4MIo4EeKthuzKkyyAbg" orderingGuide="" presentationName="Qualified software components available_Base_Optional_Input" suppressed="false" isAbstract="false" variabilityType="na"/>

<ContentElement xsi:type="uma:Artifact" name="Safety plan (refined)_Base_Optional_Input" briefDescription="" id="_TuudIIo1EeKthuzKkyyAbg" orderingGuide="" presentationName="Safety plan (refined)_Base_Optional_Input" suppressed="false" isAbstract="false" variabilityType="na"/>

<ContentElement xsi:type="uma:Artifact" name="Software safety requirements specification_Base_Optional_Input" briefDescription="" id="_6hXnYIo1EeKthuzKkyyAbg" orderingGuide="" presentationName="Software safety requirements specification_Base_Optional_Input" suppressed="false" isAbstract="false" variabilityType="na"/>

<ContentElement xsi:type="uma:Artifact" name="Software verification plan (refined)_Base_Optional_Input" briefDescription="" id="_iY1Zwo4EeKthuzKkyyAbg" orderingGuide="" presentationName="Software verification plan (refined)_Base_Optional_Input" suppressed="false" isAbstract="false" variabilityType="na"/>

<ContentElement xsi:type="uma:Artifact" name="Software verification report_Base_Optional_Input" briefDescription="" id="_tOcIo4EeKthuzKkyyAbg" orderingGuide="" presentationName="Software verification report_Base_Optional_Input" suppressed="false" isAbstract="false" variabilityType="na"/>

<ContentElement xsi:type="uma:Artifact" name="System design specification_Base_Optional_Input" briefDescription="" id="_0KYIwo4EeKthuzKkyyAbg" orderingGuide="" presentationName="System design specification_Base_Optional_Input" suppressed="false" isAbstract="false" variabilityType="na"/>

<ContentElement xsi:type="uma:Artifact" name="Tool application guidelines_Base_Optional_Input" briefDescription="" id="_EfXaMlo5EeKthuzKkyyAbg" orderingGuide="" presentationName="Tool application guidelines_Base_Optional_Input" suppressed="false" isAbstract="false" variabilityType="na"/>

<ContentElement xsi:type="uma:Task" name="Embedded software ASIL measurement and qualifying in accordance with ISO 26262_Base_Optional" briefDescription="According to: ISO26262, Section 7.4.10" id="_5YMol05EeKthuzKkyyAbg" orderingGuide="" presentationName="Embedded software ASIL measurement and qualifying in accordance with ISO 26262_Base_Optional" suppressed="false" isAbstract="false" variabilityType="na"/>
Ensure that software partitioning implement freedom from interface between software components _Base_Optional_ according to: ISO26262, Section 7.4.11

Ensure about shared resources

Ensure if software partitioning support by dedicated hardware or equivalent means

ASIL consideration about partitioning part of software development

Software partitioning verification
<ContentElement xsi:type="uma:Task" name="Software safety mechanisms specification at the software architectural level_Base_Optional" briefDescription="According to: ISO26262, Section 7.4.14" id="_h1qwYo6EeKthuzKkyyAbg" orderingGuide="" presentationName="Software safety mechanisms specification at the software architectural level_Base_Optional" suppressed="false" isAbstract="false" variabilityType="na"/>

<ContentElement xsi:type="uma:Task" name="Hazard analysis and risk assessment in accordance with the change management process in ISO 26262_Base_Optional" briefDescription="This task is needed if new hazards introduced.&amp;xD;&amp;xA; According to ISO26262, section 7.4.16" id="_srCo6EeKthuzKkyyAbg" orderingGuide="" presentationName="Hazard analysis and risk assessment in accordance with the change management process in ISO 26262_Base_Optional" suppressed="false" isAbstract="false" variabilityType="na"/>

<ContentElement xsi:type="uma:Task" name="Providing Upper estimation of required resources for the embedded software_Base_Optional" briefDescription="According to ISO26262, section 7.4.17" id="_XE11oIo7EeKthuzKkyyAbg" orderingGuide="" presentationName="Providing Upper estimation of required resources for the embedded software_Base_Optional" suppressed="false" isAbstract="false" variabilityType="na"/>

<ContentElement xsi:type="uma:Artifact" name="Safety plan (refined)_Base_Optional_Output" briefDescription="" id="_milWAlO7EeKthuzKkyyAbg" orderingGuide="" presentationName="Safety plan (refined)_Base_Optional_Output" suppressed="false" isAbstract="false" variabilityType="na"/>

<ContentElement xsi:type="uma:Artifact" name="Software safety requirements specification (refined)_Base_Optional_Output" briefDescription="" id="_udCrIolo7EeKthuzKkyyAbg" orderingGuide="" presentationName="Software safety requirements specification (refined)_Base_Optional_Output" suppressed="false" isAbstract="false" variabilityType="na"/>

<ContentElement xsi:type="uma:Artifact" name="Safety analysis report_Base_Optional_Output" briefDescription="" id="_3hSS4Io7EeKthuzKkyyAbg" orderingGuide="" presentationName="Safety analysis report_Base_Optional_Output" suppressed="false" isAbstract="false" variabilityType="na"/>

<ContentElement xsi:type="uma:Artifact" name="Dependent failures analysis report_Base_Optional_Output" briefDescription="" id="_86zMl07EeKthuzKkyyAbg" orderingGuide="" presentationName="Dependent failures analysis report_Base_Optional_Output" suppressed="false" isAbstract="false" variabilityType="na"/>

<ContentElement xsi:type="uma:Artifact" name="Software verification report (refined)_Base_Optional_Output" briefDescription="" id="_ExRIMIo8EeKthuzKkyyAbg" orderingGuide="" presentationName="Software verification report (refined)_Base_Optional_Output" suppressed="false" isAbstract="false" variabilityType="na"/>

<ContentElement xsi:type="uma:Task" name="Allocation of software safety requirements to software components_Base_Optional" briefDescription="" id="_ya4olsMEeKq7142qVWWw" orderingGuide="" presentationName="Allocation of software safety requirements to software components_Base_Optional" suppressed="false" isAbstract="false" variabilityType="na"/>

</MethodPackage>
<MethodPackage xsi:type="uma:ContentPackage" name="Design_Base_Optional_EN50128" briefDescription="" id="_8oAEElOyEeKthuzKkyyAbg" orderingGuide="" presentationName="Design_Base_Optional_EN50128" suppressed="false" global="false">

<ContentElement xsi:type="uma:Role" name="Developer_Optional" briefDescription="According to: EN 50128, section 9.4.1" id="_758yoIovEeKthuzKkyyAbg" orderingGuide="" presentationName="Developer_Optional" suppressed="false" isAbstract="false" variabilityType="na"/>

<ContentElement xsi:type="uma:Role" name="Supplier_Optional" briefDescription="According to: EN 50128, section 9.4.1" id="_LtpU4IowEeKthuzKkyyAbg" orderingGuide="" presentationName="Supplier_Optional" suppressed="false" isAbstract="false" variabilityType="na"/>

<ContentElement xsi:type="uma:Artifact" name="Software Requirements Specification_Base_Optional_Input" briefDescription="" id="_TsmSAI08EeKthuzKkyyAbg" orderingGuide="" presentationName="Software Requirements Specification_Base_Optional_Input" suppressed="false" isAbstract="false" variabilityType="na"/>

<ContentElement xsi:type="uma:Artifact" name="System Safety Requirements Specification_Base_Optional_Input" briefDescription="" id="_c4RcIo8EeKthuzKkyyAbg" orderingGuide="" presentationName="System Safety Requirements Specification_Base_Optional_Input" suppressed="false" isAbstract="false" variabilityType="na"/>

<ContentElement xsi:type="uma:Artifact" name="System Architecture Description_Base_Optional_Input" briefDescription="" id="_h-aKwJo8EeKthuzKkyyAbg" orderingGuide="" presentationName="System Architecture Description_Base_Optional_Input" suppressed="false" isAbstract="false" variabilityType="na"/>

<ContentElement xsi:type="uma:Artifact" name="Software Quality Assurance _Base_Optional_Input" briefDescription="" id="_nfQkMI08EeKthuzKkyyAbg" orderingGuide="" presentationName="Software Quality Assurance _Base_Optional_Input" suppressed="false" isAbstract="false" variabilityType="na"/>

<ContentElement xsi:type="uma:Task" name="Hardware- Software interaction specification_Base_Optional" briefDescription="According to: EN50128, section 9.4.4" id="_x_eaoIo8EeKthuzKkyyAbg" orderingGuide="" presentationName="Hardware- Software interaction specification_Base_Optional" suppressed="false" isAbstract="false" variabilityType="na"/>

<ContentElement xsi:type="uma:Task" name="COT software identification and verification considering related safety integrity level_Base_Optional" briefDescription="According to: EN50128, section 9.4.6" id="_8X8qUlo8EeKthuzKkyyAbg" orderingGuide="" presentationName="COT software identification and verification considering related safety integrity level_Base_Optional" suppressed="false" isAbstract="false" variabilityType="na"/>

<ContentElement xsi:type="uma:Task" name="Identifying software development strategy considering SIL_Base_Optional" briefDescription="According to: EN50128, section 9.4.10" id="_IPNzQIo9EeKthuzKkyyAbg" orderingGuide="" presentationName="Identifying software development strategy considering SIL_Base_Optional" suppressed="false" isAbstract="false" variabilityType="na"/>
Previously developed software identification and suitability justification considering the required software SIL_Base_Optional

Designer_Base

DO178B_Design_Variants

DO178B_Design_CS_Variants

DO178B_Design_CPS_Variants

01_Design description
<ContentElement xsi:type="uma:Task" name="03_Design description verification" briefDescription="" id="_qUqecIsVEeKq7p142qVWWw" orderingGuide="" presentationName="03_Design description verification" suppressed="false" isAbstract="false" variabilityBasedOnElement="_RwLiQIo0EeKthuzKkyyAbg" variabilityType="replaces"/>

<ContentElement xsi:type="uma:Task" name="04_Safety specification" briefDescription="" id="_vzOY4IsVEeKq7p142qVWWw" orderingGuide="" presentationName="" suppressed="false" isAbstract="false" variabilityBasedOnElement="_wSrxYIorEeKthuzKkyyAbg" variabilityType="contributes">

</Presentation>

</ContentElement>

<ContentElement xsi:type="uma:Task" name="05_Monitoring of the safety related requirements" briefDescription="" id="_0fG5MIsVEeKq7p142qVWWw" orderingGuide="" presentationName="05_Monitoring of the safety related requirements" suppressed="false" isAbstract="false" variabilityBasedOnElement="_gak6Ilo0EeKthuzKkyyAbg" variabilityType="replaces">

</ContentElement>
If the protection is provided by software, it should be designed and verified at the same software level as the non-modifiable software.

The method provided to change the modifiable component should be shown to be the only means by which the modifiable component can be changed.
<Presentation xsi:type="uma:TaskDescription" name="01_Software architectural design specification,_I4mWqIsMEeKq7p142qVWWw" briefDescription="" id="-rqSa8ZLA2U_1ulZUsakjja" orderingGuide="" presentationName="" suppressed="false" authors="" changeDescription="" version="" externalId=""/>

<MainDescription></MainDescription>

<KeyConsiderations></KeyConsiderations>

<Section name="Specifying Software architectural design considering software safety integrity level." briefDescription="" id="_13FsI1YEeK95dfTcjc1-g" orderingGuide="" presentationName="" suppressed="false" sectionName="" variabilityType="na">

<Description></Description>
</Section>

<Alternatives></Alternatives>

<Purpose></Purpose>
</Presentation>
</ContentElement>

<ContentElement xsi:type="uma:Task" name="02_Software components specification" briefDescription="" id="_ocUL0IsMEeKq7p142qVWWw" orderingGuide="" presentationName="" suppressed="false" isAbstract="false" variabilityBasedOnElement="_fz5DsIorEeKthutzKkyyAbg" variabilityType="contributes">

<Presentation xsi:type="uma:TaskDescription" name="02_Software components specification,_ocUL0IsMEeKq7p142qVWWw" briefDescription="" id="-weUrPj0KTkxs0xPAzr1Zg" orderingGuide="" presentationName="" suppressed="false" authors="" changeDescription="" version="" externalId=""/>

<MainDescription></MainDescription>

<KeyConsiderations></KeyConsiderations>

<Section name="Components Static aspects design" briefDescription="" id="_nEgykJ_GEeK0XIf1uYLVA" orderingGuide="" presentationName="" suppressed="false" sectionName="" variabilityType="na">

<Description></Description>
</Section>

<Section name="Components Dynamic aspects design" briefDescription="" id="_osJewI_GEeK0XIf1uYLVA" orderingGuide="" presentationName="" suppressed="false" sectionName="" variabilityType="na">

<Description></Description>
</Section>
<Section name="Components included in New or reuse with modification categories shall developed by considering ISO 2662" briefDescription="" id="_qYRypYl_GEeK0XIf1uIYLVA" orderingGuide="" presentationName="" suppressed="false" sectionName="" variabilityType="na">
</Section>

<Section name="Qualifying components included in reuse without modification category in accordance with ISO 26262" briefDescription="" id="_thVFAI_GEeK0XIf1uIYLVA" orderingGuide="" presentationName="" suppressed="false" sectionName="" variabilityType="na">
</Section>

<Alternatives></Alternatives>

<Purpose></Purpose>

</Presentation>

<ContentElement xsi:type="uma:Task" name="03_Allocation of software safety requirements to software components" briefDescription="" id="_imgYwlsMEeKq7p142qVWWw" orderingGuide="" presentationName="03_Allocation of software safety requirements to software components" suppressed="false" isAbstract="false" variabilityBasedOnElement="_y-a4oIsMEeKq7p142qVWWw" variabilityType="replaces"/>

<ContentElement xsi:type="uma:Task" name="04_Embedded software ASIL measurement and qualification" briefDescription="If any embedded software identified." id="_8BH0MIsMEeKq7p142qVWWw" orderingGuide="" presentationName="04_Embedded software ASIL measurement and qualification" suppressed="false" isAbstract="false" variabilityBasedOnElement="_5YMoIo5EeKthuzKkyyAbg" variabilityType="replaces"/>

<ContentElement xsi:type="uma:Task" name="05_Ensuring that software partitioning implement freedom from interface between software components" briefDescription="" id="_Lh-uAlSNEeKq7p142qVWWw" orderingGuide="" presentationName="05_Ensuring that software partitioning implement freedom from interface between software components" suppressed="false" isAbstract="false" variabilityBasedOnElement="_QNy6wIo6EeKthuzKkyyAbg" variabilityType="replaces"/>

<Presentation xsi:type="uma:TaskDescription" name="05_Ensuring that software partitioning implement freedom from interface between software components" briefDescription="" id=".tche3K-149pbdz96svLwdg" orderingGuide="" presentationName="" suppressed="false" authors="" changeDescription="" version="" externalId=""/>

</MainDescription>

<KeyConsiderations></KeyConsiderations>
<Section name="Ensure about shared resources" briefDescription="" id="_yCXAMI_GEeK0XIf1uYLVA" orderingGuide="" presentationName="" suppressed="false" sectionName="" variabilityType="na">
  <Description/>
</Section>

<Section name="Ensure if software partitioning support by dedicated hardware or equivalent means" briefDescription="" id="_zk0RcL_GEeK0XIf1uYLVA" orderingGuide="" presentationName="" suppressed="false" sectionName="" variabilityType="na">
  <Description/>
</Section>

<Section name="ASIL consideration about partitioning part of software development" briefDescription="" id="_19YEQI_GEeK0XIf1uYLVA" orderingGuide="" presentationName="" suppressed="false" sectionName="" variabilityType="na">
  <Description/>
</Section>

<Section name="Software partitioning verification" briefDescription="" id="_45da4I_GEeK0XIf1uYLVA" orderingGuide="" presentationName="" suppressed="false" sectionName="" variabilityType="na">
  <Description/>
</Section>

<Alternatives/>

<Purpose/>

</Presentation>

<ContentElement xsi:type="uma:Task" name="06_Safety analysis" briefDescription="" id="_RQYbQIsNEeKq7p142qVWWW" orderingGuide="" presentationName="" suppressed="false" isAbstract="false" variabilityBasedOnElement="_wSrxeYyorEeKthuzKkkyyAbg" variabilityType="contributes">
  <Presentation xsi:type="uma:TaskDescription" name="06_Safety analysis_RQYbQIsNEeKq7p142qVWWW" briefDescription="" id="_s9HnVfd0WDL9esu97yjVfQ" orderingGuide="" presentationName="" suppressed="false" authors="" changeDescription="" version="" externalId="">  
    <MainDescription/>
  </Presentation>
  <KeyConsiderations/>
</ContentElement>
<Section name="Failure analysis accordance with ISO 26262" briefDescription=""
 id="_9W52kI_GEeK0XlIf1uYLVA" orderingGuide="" presentationName="" suppressed="false"
 sectionName="" variabilityType="na">
 <Description></Description>
 </Section>

<Section name="Identify or confirm the safety-related parts of the software"
 briefDescription="" id="_V-t8l_GEeK0XlIf1uYLVA" orderingGuide="" presentationName="" suppressed="false"
 sectionName="" variabilityType="na">
 <Description></Description>
 </Section>

<Section name="Support the specification and verify the efficiency of the safety mechanisms"
 briefDescription="" id="_o6M5wI_GEeK0XlIf1uYLVA" orderingGuide="" presentationName="" suppressed="false"
 sectionName="" variabilityType="na">
 <Description></Description>
 </Section>

<Alternatives></Alternatives>

<Purpose></Purpose>

</Presentation>

</ContentElement>

<ContentElement xsi:type="uma:Task" name="07_Software safety mechanisms specification at
 the software architectural level" briefDescription="" id="_caKzIIsNEeKq7p142qVWWw"
 orderingGuide="" presentationName="07_Software safety mechanisms specification at the software
 architectural level" suppressed="false" isAbstract="false"
 variabilityBasedOnElement="_h1qwYIo6EeKthuzKkyyAbg" variabilityType="replaces">
 <MethodElementProperty name="me_edited" value="true"/>

 <Presentation xsi:type="uma:TaskDescription" name="07_Software safety mechanisms
 specification at the software architectural level._caKzIIsNEeKq7p142qVWWw" briefDescription=""
 id="-3NTxBpKMKthErijwo5F1Q" orderingGuide="" presentationName="" suppressed="false"
 authors="" changeDescription="" version="" externalId="">
 <MainDescription></MainDescription>
 <KeyConsiderations></KeyConsiderations>

 <Section name="Error detection at Software architectural level" briefDescription=""
 id="_bfmmIOC_EeOMvsZxxxy52za" orderingGuide="" presentationName="" suppressed="false"
 sectionName="" variabilityType="na">
 <Description></Description>
</Presentation>
<ContentElement xsi:type="uma:Task" name="08_Hazard analyses and risk assessment in accordance with the change management process" briefDescription="" id="_0fRx8IsNEeKq7p142qVWWW" orderingGuide="" presentationName="08_Hazard analyses and risk assessment in accordance with the change management process" suppressed="false" isAbstract="false" variabilityBasedOnElement="_srC-gIo6EeKthuzKkyyAbg" variabilityType="replaces"/>

<ContentElement xsi:type="uma:Task" name="09_Provision of upper estimation of required resources for the embedded software" briefDescription="" id="_C60OcIsOEeKq7p142qVWWW" orderingGuide="" presentationName="09_Provision of upper estimation of required resources for the embedded software" suppressed="false" isAbstract="false" variabilityBasedOnElement="_XE11oIo7EeKthuzKkyyAbg" variabilityType="replaces"/>

<ContentElement xsi:type="uma:Task" name="10_Verification of software architectural design" briefDescription="" id="_SvDjQIsOEeKq7p142qVWWW" orderingGuide="" presentationName="" suppressed="false" isAbstract="false" variabilityBasedOnElement="_2Ar2AIorEeKthuzKkyyAbg" variabilityType="contributes">
  <Presentation xsi:type="uma:TaskDescription" name="10_Verification of software architectural design," briefDescription="" id="_jhar9y1h2hQ09VN3DzdDw" orderingGuide="" presentationName="" suppressed="false" authors="" changeDescription="" version="" externalId=""">
    <MainDescription></MainDescription>
    <KeyConsiderations></KeyConsiderations>
    <Section name="Verification of compliance with the software safety requirements" briefDescription="" id="_FREQwL_HEeK0XIf1ulYLVA" orderingGuide="" presentationName="" suppressed="false" sectionName="" variabilityType="na">
      <Description></Description>
    </Section>
    <Section name="Verification of compatibility with the target hardware" briefDescription="" id="_1sOEI_HEeK0XIf1ulYLVA" orderingGuide="" presentationName="" suppressed="false" sectionName="" variabilityType="na">
      <Description></Description>
    </Section>
  </Presentation>
</ContentElement>
<Section name="Verification of adherence to design guidelines" briefDescription=""
 id="_HwaoEl_HEeK0XH1uiYLVA" orderingGuide="" presentationName="" suppressed="false"
 sectionName="" variabilityType="na">
 </Section>

</alternatives>

<purpose></purpose>
</presentation>
</contentelement>

<contentelement xsi:type="uma:task" name="08_Application of software safety mechanism" 
 briefDescription="" id="_B_3A4ODAEeOMvsZxxy52zg" orderingGuide="" presentationName="08_Application of software safety mechanism" suppressed="false" 
 isAbstract="false" variabilityType="na">
 <methodelementproperty name="me_edited" value="true"/>
 <presentation xsi:type="uma:TaskDescription" name="_B_3A4ODAEeOMvsZxxy52zg" 
 briefDescription="" id="zkTN72L9sX14NnsUuulAA" orderingGuide="" presentationName="" 
 suppressed="false" authors="" changeDescription="" version="" externalId="">
 <maindescription></maindescription>
 <keyconsiderations></keyconsiderations>
 <section name="Error handling " briefDescription="" id="_OEDLgODAEeOMvsZxxy52zg" 
 orderingGuide="" presentationName="" suppressed="false" sectionName="" variabilityType="na">
 </section>
 </presentation>
 <toolmentor>_WzKQsODAEeOMvsZxxy52zg</toolmentor>
</contentelement>

<contentelement xsi:type="uma:ToolMentor" name="Mechanism for error detection at the software architectural level" 
 briefDescription="" id="_WzKQsODAEeOMvsZxxy52zg" orderingGuide="" presentationName="Mechanism for error detection at the software architectural level" suppressed="false" isAbstract="false" variabilityType="na">
 <methodelementproperty name="me_edited" value="true"/>
Mechanisms for error handling at the software architectural level

MethodElementProperty name="me_edited" value="true"/>

</ContentElement>

</ContentElement>

</MethodPackage>

</MethodPackage>

</MethodPackage>

</MethodPackage>

</MethodPackage>

</MethodPackage>

</MethodElement>

</ContentPackage>

</ContentPackage>

</ContentPackage>

</ContentElement>

</MethodElement>

</MethodElement>

</MethodElement>

</MethodElement>

</MethodElement>

</MethodElement>

</MethodElement>

</MethodElement>

</MethodElement>

</MethodElement>

</MethodElement>

</MethodElement>

</MethodElement>

</MethodElement>

</MethodElement>

</MethodElement>

</MethodElement>

</MethodElement>

</MethodElement>

</MethodElement>

</MethodElement>

</MethodElement>

</MethodElement>

</MethodElement>

</MethodElement>

</Presentation>

</MainDescription>

<KeyConsiderations>

</ContentElement>

</ContentElement>

</ContentElement>

</ContentElement>

</ContentElement>

</ContentElement>

</ContentElement>

</ContentElement>

</ContentElement>

</ContentElement>

</ContentElement>

</ContentElement>

</ContentElement>

</ContentElement>

</ContentElement>

</ContentElement>

</ContentElement>

</ContentElement>

</ContentElement>

</ContentElement>

</ContentElement>

</ContentElement>

</ContentElement>

</ContentElement>

</ContentElement>

</ContentElement>

</ContentElement>

</ContentElement>

</ContentElement>

</ContentElement>

</ContentElement>

</ContentElement>

</ContentElement>

</ContentElement>

</ContentElement>

</ContentElement>

</ContentElement>

</ContentElement>

</ContentElement>

</ContentElement>

</ContentElement>

</ContentElement>

</ContentElement>

</ContentElement>

</ContentElement>

</ContentElement>

</ContentElement>

</ContentElement>

</ContentElement>

</ContentElement>

</ContentElement>

</ContentElement>

</ContentElement>

</ContentElement>

</Presentation>

</MainDescription>

</KeyConsiderations>
<Section name="Identifying if component are validated if so, their validation conditions" briefDescription="" id="_RyUx4i_HEeK0XIf1uYLVA" orderingGuide="" presentationName="" suppressed="false" sectionName="" variabilityType="na">
  <Description></Description>
</Section>

<Section name="Identifying SILs of the components" briefDescription="" id="_T6W1YI_HEeK0XIf1uYLVA" orderingGuide="" presentationName="" suppressed="false" sectionName="" variabilityType="na">
  <Description></Description>
</Section>

<Section name="Specifying COTS components if applicable by considering " briefDescription="" id="_VwiT4I_HEeK0XIf1uYLVA" orderingGuide="" presentationName="" suppressed="false" sectionName="" variabilityType="na">
  <Description></Description>
</Section>

<Alternatives></Alternatives>

<Purpose></Purpose>

</Presentation>

</ContentElement>

<ContentElement xsi:type="uma:Task" name="04_COT software identification and verification considering related safety integrity level" briefDescription="" id="_m_egwIsUEeKq7p142qVWWw" orderingGuide="" presentationName="04_COT software identification and verification considering related safety integrity level" suppressed="false" isAbstract="false" variabilityBasedOnElement="_8X8qUlo8EeKthuzKkyyAbg" variabilityType="replaces"/>

<ContentElement xsi:type="uma:Task" name="05_Previously developed software identification and suitability justification considering the required software SIL" briefDescription="" id="_sVDeUIsUEeKq7p142qVWWw" orderingGuide="" presentationName="05_Previously developed software identification and suitability justification considering the required software SIL" suppressed="false" isAbstract="false" variabilityBasedOnElement="_7v-XwlsUEeKq7p142qVWWw" variabilityType="replaces"/>

<ContentElement xsi:type="uma:Task" name="06_Safety specification" briefDescription="" id="_8iwcAIsUEeKq7p142qVWWw" orderingGuide="" presentationName="" suppressed="false" isAbstract="false" variabilityBasedOnElement="_wSrxFY1orEeKthuzKkyyAbg" variabilityType="contributes"/>

<ContentElement xsi:type="uma:Task" name="07_Identification of software development strategy considering SIL" briefDescription="" id="_BrloIsVFeeKq7p142qVWWw" orderingGuide="" presentationName="07_Identification of software development strategy considering
SIL" suppress="false" isAbstract="false"
variabilityBasedOnElement="_IPNzQIo9EeKthuzKkyyAbg" variabilityType="replaces"/>

<ContentElement xsi:type="uma:Task" name="08_Verification" briefDescription=""
id="_Gx_QIIsVEeKq7p142qVWWw" orderingGuide="" presentationName="" suppressed="false"
isAbstract="false" variabilityBasedOnElement="_2Ar2AIorEeKthuzKkyyAbg"
variabilityType="contributes">

<Presentation xsi:type="uma:TaskDescription" name="_Gx_QIIsVEeKq7p142qVWWw"
briefDescription="" id="dlBOr0KVVi5yA0uJGoEi29Q" orderingGuide="" presentationName="" suppressed="false"
authors="" changeDescription="" version="" externalId="">

<MainDescription></MainDescription>

<KeyConsiderations></KeyConsiderations>

<Section name="Justification provision that the selected measures and techniques satisfies the
Software Requirements Specification at the required SIL" briefDescription=""
id="_Lwtz8IsVEeKq7p142qVWWw" orderingGuide="" presentationName="" suppressed="false"
sectionName="" variabilityType="na">

<Description></Description>

</Section>

<Alternatives></Alternatives>

<Purpose></Purpose>

</Presentation>

</ContentElement>

</MethodPackage>

</MethodPackage>

</MethodPlugin>

<MethodPlugin name="DO178B_Design_Delivey process" briefDescription=""
id="_NWjbllqEcKthuzKkyyAbg" orderingGuide="" presentationName="DO178B_Design_Delivey
process" suppressed="false" authors="" changeDescription="" version="" supporting="false"
userChangeable="true">

<MethodElementProperty name="plugin_synFree" value="true"/>

<ReferencedMethodPlugin>_S3x58IooEeKthuzKkyyAbg</ReferencedMethodPlugin>

<MethodPackage xsi:type="uma:ContentCategoryPackage" name="ContentCategories"
id="_sQiNQjt3EeapT9Qu_oq2uA"/>

<MethodPackage xsi:type="uma:ProcessComponent" name="DO178B_Design_Delivey process"
briefDescription="" id="_i3Jc0IsYEcKcfPN9Aq1m9g" orderingGuide="" presentationName=""
suppressed="false" global="false" authors="" changeDescription="" version=""></MethodPackage>
<Process xsi:type="uma:DeliveryProcess" name="DO178B_Design_Delivey process"
briefDescription="" id="_i3JcoYsYeeKcfPN9Aq1m9g" orderingGuide=""
presentationName="DO178B_Design_Delivey process" suppressed="false" isAbstract="false"
hasMultipleOccurrences="false" isOptional="false" isPlanned="true" prefix="" isEventDriven="false"
isOngoing="false" isRepeatable="false" variabilityType="na"
diagramURI="DO178B_Design_Delivey%20process/deliveryprocesses/DO178B_Design_Delivey%20process/diagram.xmi">

<Presentation xsi:type="uma:DeliveryProcessDescription" name="DO178B_Design_Delivey process._i3JcoYsYeeKcfPN9Aq1m9g" briefDescription="" id="-9PXU-cSwP9iLPQGsWkhHtw" orderingGuide="" presentationName="" suppressed="false" authors="" changeDescription=""
version="" externalId="" usageGuidance="">

<MainDescription/>

<KeyConsiderations/>

<Alternatives/>

<HowToStaff/>

<Purpose/>

<Scope/>

<UsageNotes/>

<Scale/>

<ProjectCharacteristics/>

<RiskLevel/>

<EstimatingTechnique/>

<ProjectMemberExpertise/>

<TypeOfContract/>

</Presentation>

<BreakdownElement xsi:type="uma:TaskDescriptor" name="Software architectural design Base" briefDescription="" id="_Yc1jclskEeKyBspY1Xltzw" orderingGuide=""
presentationName="Software architectural design Base" suppressed="false" isAbstract="false"
hasMultipleOccurrences="false" isOptional="false" isPlanned="false" prefix="" isEventDriven="false" isOngoing="false" isRepeatable="false" isSynchronizedWithSource="true">

<MethodElementProperty name="me_references" value="&lt;?xml version="1.0" encoding="UTF-8" standalone="no"?&gt;&lt;me_references udtList=""/&gt;"

<SuperActivity>_i3JcoYsYeeKcfPN9Aq1m9g</SuperActivity>

<Task>_OD0UUIorEeKthuzKkyyAbg</Task>
<Step name="Software architectural design specification _Base " briefDescription="" id="_T6U0sIorEeKthuzKkyyAbg" orderingGuide="" presentationName="" suppressed="false" sectionName="" variabilityType="na">
  <Description></Description>
</Step>
</BreakdownElement>

<BreakdownElement xsi:type="uma:TaskDescriptor" name="Software components specification _Base" briefDescription="" id="_ZLfZMIskEeKyBspY1XLtZw" orderingGuide="" presentationName="Software components specification _Base" suppressed="false" isAbstract="false" hasMultipleOccurrences="false" isOptional="false" isPlanned="false" prefix="" isEventDriven="false" isOngoing="false" isRepeatable="false" isSynchronizedWithSource="true">
  <MethodElementProperty name="me_references" value="&lt;?xml version="1.0" encoding="UTF-8" standalone="no"?&gt;&lt;me_references udtList=""/&gt;"/>
  <SuperActivity>_i3JcoYsYEeKcfPN9Aq1m9g</SuperActivity>
  <Predecessor id="_fm7j4IsqEeKyBspY1XLtZw" linkType="finishToStart">_rV8-glskEeKyBspY1XLtZw</Predecessor>
  <Predecessor id="_fm8K8IsqEeKyBspY1XLtZw" linkType="finishToStart">_Yc1jcIskEeKyBspY1XLtZw</Predecessor>
  <Task>_fz5DsIorEeKthuzKkyyAbg</Task>
  <Step name="Software components identification and description " briefDescription="" id="_i-xrUlorEeKthuzKkyyAbg" orderingGuide="" presentationName="" suppressed="false" sectionName="" variabilityType="na">
    <Description></Description>
  </Step>
  <Step name="Categorizing components" briefDescription="" id="_kvRYkIorEeKthuzKkyyAbg" orderingGuide="" presentationName="" suppressed="false" sectionName="" variabilityType="na">
    <Description></Description>
  </Step>
</BreakdownElement>

<BreakdownElement xsi:type="uma:TaskDescriptor" name="Contracts definition_Base" briefDescription="" id="_bhve4IskEeKyBspY1XLtZw" orderingGuide="" presentationName="Contracts definition_Base" suppressed="false" isAbstract="false" hasMultipleOccurrences="false" isOptional="false" isPlanned="false" prefix="" isEventDriven="false" isOngoing="false" isRepeatable="false" isSynchronizedWithSource="true"/>
<MethodElementProperty name="me_references" value="&lt;?xml version="1.0" encoding="UTF-8" standalone="no"?&gt;&lt;me_references udtList=""/&gt;">

</MethodElementProperty>

<SuperActivity>_i3JcoYsYEeKcfPN9Aq1m9g</SuperActivity>

<Predecessor id="_w-la8IskEeKyBspY1XLtZw" linkType="finishToStart">_ZLfZMIskEeKyBspY1XLtZw</Predecessor>

<Task>_vKMkIoEeKthuzKkyyAbg</Task>

</BreakdownElement>

<BreakdownElement xsi:type="uma:TaskDescriptor" name="03_Design description verification" briefDescription="" presentationName="03_Design description verification" suppressed="false" isAbstract="false" hasMultipleOccurences="false" isOptional="false" isPlanned="false" prefix="" isEventDriven="false" isOngoing="false" isRepeatable="false" isSynchronizedWithSource="true">

</BreakdownElement>

<MethodElementProperty name="me_references" value="&lt;?xml version="1.0" encoding="UTF-8" standalone="no"?&gt;&lt;me_references udtList=""/&gt;">

</MethodElementProperty>

<MethodElementProperty name="wbe_GlobalPresentedAfter" value="_bhve4IskEeKyBspY1XLtZw"/>

<SuperActivity>_i3JcoYsYEeKcfPN9Aq1m9g</SuperActivity>

<Predecessor id="_x3weoIskEeKyBspY1XLtZw" linkType="finishToStart">_bhve4IskEeKyBspY1XLtZw</Predecessor>

<Task>_qUqecIsVEeKq7p142qVWWw</Task>

</BreakdownElement>

<BreakdownElement xsi:type="uma:TaskDescriptor" name="Certificate preparation_Base" briefDescription="" presentationName="Certificate preparation_Base" suppressed="false" isAbstract="false" hasMultipleOccurences="false" isOptional="false" isPlanned="false" prefix="" isEventDriven="false" isOngoing="false" isRepeatable="false" isSynchronizedWithSource="true">

</BreakdownElement>

<MethodElementProperty name="me_references" value="&lt;?xml version="1.0" encoding="UTF-8" standalone="no"?&gt;&lt;me_references udtList=""/&gt;">

</MethodElementProperty>

<MethodElementProperty name="wbe_GlobalPresentedAfter" value="_lyKl0IskEeKyBspY1XLtZw"/>

<SuperActivity>_i3JcoYsYEeKcfPN9Aq1m9g</SuperActivity>

<Predecessor id="_XnGb0I_mEeKrTIqgX7A_CQ" linkType="finishToStart">_rrkCIIskEeKyBspY1XLtZw</Predecessor>

</Master Thesis>
<Task>_Gyi0MIotEeKthuzKkyyAbg</Task>

</BreakdownElement>

<BreakdownElement xsi:type="uma:TaskDescriptor" name="Safety specification_Base" briefDescription="" id="_qWg6wIskEeKyBspY1XLiZw" orderingGuide="" presentationName="Safety specification_Base" suppressed="false" isAbstract="false" hasMultipleOccurrences="false" isOptional="false" isPlanned="false" prefix="" isEventDriven="false" isOngoing="false" isRepeatable="false" isSynchronizedWithSource="true">
  <MethodElementProperty name="me_references" value=""/>
  <MethodElementProperty name="wbe_GlobalPresentedAfter" value=""/>
  <SuperActivity>_i3JcoYsYEeKcfPN9Aq1m9g</SuperActivity>
  <Predecessor id="_hWtUsI_mEeKrTIQgX7A_CQ" linkType="finishToStart">_lyKl0IskEeKyBspY1XLiZw</Predecessor>
  <Task>_wSrxYIorEeKthuzKkyyAbg</Task>
  <Step name="Safety analysis" briefDescription="" id="_zX2LgIorEeKthuzKkyyAbg" orderingGuide="" presentationName="" suppressed="false" sectionName="" variabilityType="na">
    <Description></Description>
  </Step>
</BreakdownElement>

<BreakdownElement xsi:type="uma:TaskDescriptor" name="05_Monitoring of the safety related requirements" briefDescription="" id="_rC9xEIskEeKyBspY1XLiZw" orderingGuide="" presentationName="05_Monitoring of the safety related requirements" suppressed="false" isAbstract="false" hasMultipleOccurrences="false" isOptional="false" isPlanned="false" prefix="" isEventDriven="false" isOngoing="false" isRepeatable="false" isSynchronizedWithSource="true">
  <MethodElementProperty name="me_references" value=""/>
  <MethodElementProperty name="wbe_GlobalPresentedAfter" value=""/>
  <SuperActivity>_i3JcoYsYEeKcfPN9Aq1m9g</SuperActivity>
  <Predecessor id="_2uxQ4IskEeKyBspY1XLiZw" linkType="finishToStart">_qWg6wIskEeKyBspY1XLiZw</Predecessor>
  <Task>_0fG5MIsVEeKq7p142qVWWw</Task>
</BreakdownElement>
<Step name="Monitoring control flow" briefDescription=""
    id="_10LPgI_HEeK0XIf1uYLVA" orderingGuide="" presentationName=""
    suppressed="false" sectionName="" variabilityType="na">
    <Description></Description>
</Step>

<Step name="Monitoring data flow" briefDescription=""
    id="_2wuXUI_HEeK0XIf1uYLVA" orderingGuide="" presentationName=""
    suppressed="false" sectionName="" variabilityType="na">
    <Description></Description>
</Step>

<BreakdownElement xsi:type="uma:TaskDescriptor" name="06_Design user modifiable software"
    briefDescription="" id="_rV8-glskEeKyBspY1XLTzW" orderingGuide="" presentationName="06_Design user modifiable software"
    suppressed="false" isAbstract="false" hasMultipleOccurences="false" isOptional="false" isPlanned="false"
    prefix="" isEventDriven="false" isOngoing="false" isRepeatable="false" isSynchronizedWithSource="true">
    <MethodElementProperty name="me_references" value="<?xml version="1.0" encoding="UTF-8" standalone="no"?>
        <me_references udtList=""/>
        
        <MethodElementProperty name="wbe_GlobalPresentedAfter"
            value="&_IXZIEskEeKyBspY1XLTzW"/>
        
        <SuperActivity>_i3JcoYsYEeKcfPN9Aq1m9g</SuperActivity>
        
        <Task>_8q0WYIsVEeKq7p142qVWWw</Task>
        
        <Step name="Interference protection of non-modifiable components from modifiable components" briefDescription=""
            id="_6nAFU1_HEeK0XIf1uYLVA" orderingGuide="" presentationName=""
            suppressed="false" sectionName="" variabilityType="na">
            <Description></Description>
        </Step>
        
        <Step name="If the protection is provided by software, it should be designed and verified at the same software level as the non-modifiable software" briefDescription=""
            id="_7Q1FcI_HEeK0XIf1uYLVA" orderingGuide="" presentationName=""
            suppressed="false" sectionName="" variabilityType="na">
            <Description></Description>
        </Step>
        
        <Step name="The method provided to change the modifiable component should be shown to be the only means by which the modifiable component can be changed." briefDescription=""
            id="_7Q1FcI_HEeK0XIf1uYLVA" orderingGuide="" presentationName=""
            suppressed="false" sectionName="" variabilityType="na">
            <Description></Description>
        </Step>
    </BreakdownElement>
<BreakdownElement xsi:type="uma:TaskDescriptor" name="Verification_Base"
briefDescription="" id="_rrkC1IskEeKyBspY1XLtZw" orderingGuide=""
presentationName="Verification_Base" suppressed="false" isAbstract="false"
hasMultipleOccurrences="false" isOptional="false" isPlanned="false" prefix=""
isEventDriven="false" isOngoing="false" isRepeateable="false" isSynchronizedWithSource="true">
  <MethodElementProperty name="me_references" value="&lt;?xml version="1.0" encoding="UTF-8" standalone="no"?"><me_references udtList=""/>
  </MethodElementProperty>
  <SuperActivity>_i3JcoYsYEeKcfPN9Aq1m9g</SuperActivity>
  <Predecessor id="_Olt34IslEeKyBspY1XLtZw" linkType="finishToStart">_rC9xEIskEeKyBspY1XLtZw</Predecessor>
  <Task>_2Ar2AIorEeKthuzKkyyAbg</Task>
  <Step name="Software architectural design verification according to related standard"
briefDescription="" id="_sNpvQI1YEeK95dPtjc1-g" orderingGuide=""
presentationName="" suppressed="false" sectionName="" variabilityType="na">
    <Description></Description>
  </Step>
</BreakdownElement>

<BreakdownElement xsi:type="uma:TaskDescriptor" name="Argumentation_Base"
briefDescription="" id="_QlzjUI_mEeKrTIQgX7A_CQ" orderingGuide=""
presentationName="Argumentation_Base" suppressed="false" isAbstract="false"
hasMultipleOccurrences="false" isOptional="false" isPlanned="false" prefix=""
isEventDriven="false" isOngoing="false" isRepeateable="false" isSynchronizedWithSource="true">
  <MethodElementProperty name="me_references" value="&lt;?xml version="1.0" encoding="UTF-8" standalone="no"?"><me_references udtList=""/>
  </MethodElementProperty>
  <SuperActivity>_i3JcoYsYEeKcfPN9Aq1m9g</SuperActivity>
  <Predecessor id="_YLQRwI_mEeKrTIQgX7A_CQ" linkType="finishToStart">_iXZIElskEeKyBspY1XLtZw</Predecessor>
  <Step name="Software architectural design verification according to related standard"
briefDescription="" id="_sNpvQI1YEeK95dPtjc1-g" orderingGuide=""
presentationName="" suppressed="false" sectionName="" variabilityType="na">
    <Description></Description>
  </Step>
</BreakdownElement>
<Task>_rorJMLI_leeKrTIQgX7A_CQ</Task>

</BreakdownElement>

</Process>

</MethodPackage>

</MethodPlugin>

<MethodConfiguration name="DO178B" briefDescription="" id="_20_rIlSYeEeKcfPN9Aq1m9g" orderingGuide="" presentationName="" suppressed="false" authors="" changeDescription="" version="">

<MethodElementProperty name="TouchedByConfigEditor" value="true"/>

<MethodElementProperty name="Config_doneLoadCheckPkgs" value=""/>

<MethodElementProperty name="me_edited" value="true"/>

<MethodPluginSelection>_NWjblIoqEeKthuzKkyyAbg</MethodPluginSelection>

<MethodPackageSelection>_i3JcoIsYEeKcfPN9Aq1m9g</MethodPackageSelection>

<MethodPackageSelection>_NWkCMloqEeKthuzKkyyAbg</MethodPackageSelection>

<MethodPackageSelection>_NWjblIoqEeKthuzKkyyAbg</MethodPackageSelection>

<MethodPackageSelection>_NWjblYoqEeKthuzKkyyAbg</MethodPackageSelection>

<MethodPackageSelection>_sQiNQIt3EeapT9Qu_oq2uA</MethodPackageSelection>

<MethodPackageSelection>_NWjblIoqEeKthuzKkyyAbg</MethodPackageSelection>

<MethodPackageSelection>_NWjblIoqEeKthuzKkyyAbg</MethodPackageSelection>

<MethodPackageSelection>_NWjblIoqEeKthuzKkyyAbg</MethodPackageSelection>

<MethodPackageSelection>_NWjblIoqEeKthuzKkyyAbg</MethodPackageSelection>

</MethodConfiguration>

</uma:MethodLibrary>