Methods for Modeling of Product Lines for Safety-critical Systems

DVA407-Master Thesis

Mälardalen University
Department of Innovation, Design and Engineering
Box 883, 721 23 Västerås, Sweden

Volvo Construction Equipment
Electrical & Electronics System Architecture Department
Box 55, 631 02 Eskilstuna, Sweden

Start date: Feb. 13, 2013 Finishing date: July 13, 2013
Studying pace: full-time studies

Xiaodi Zhang
Student type: Master Stud 2yrs Student number: 870527-P164
xzg11002@student.mdh.se

Advisor at Mälardalen University and Supervisor at Volvo Construction Equipment:
Stephan Baumgart
stephan.baumgart@mdh.se

Examiner at Mälardalen University:
Sasikumar Punnekkat
sasikumar.punnekkat@mdh.se
Acknowledgement

I would like to present my heart-felt thanks to my supervisor Stephan Baumgart, for the patience and continuous encouragement. He provided me the chance to study in a new field, which broaden my horizon and enriched my knowledge. I also want to thank my examiner Sasikumar Punnekkat, who gave me useful suggestions in the thesis work.

I want to thank the colleagues from Volvo Construction Equipment functional safety group and software development group, thank their help and kindness. During the thesis work, I talked with many fellow colleagues who helped me gaining useful industrial experience. I wish the colleagues all the best for their future work.

I want to thank my friends for supporting me, sharing their thesis work experience and giving me valuable suggestions. Additionally, I want to thank the Mälardalen University Innovation, Design and Engineering (IDT) department for providing me the chance to work in the company, which helped me understanding the knowledge I learn from the university better.

Finally, I want to thank my family for their support and encouragement during my study.
Abstract

Software product line engineering is a proposed methodology that enables software products and software-intensive systems to be developed at lower cost, higher quality and less time to market. The structured and managed artifacts reuse among different products in development is the main target of software product line engineering. As a key-method of the product line engineering approach, the commonality and variability analysis is a technique that identifies the potential artifacts for reuse. But the reuse poses challenges for delivering safety-critical products from the product line and achieving product line functional safety. In order to analyze the product line and provide more valuable information for its safety analysis, we make use of established product line modeling techniques, which model the product line commonality and variability from different perspectives.

In this report, we investigate the product line modeling techniques. The product modeling analysis process covers two aspects:

1. Study different product line modeling techniques and find the ones suitable for product line modeling. We choose the modeling techniques that can be implemented to discuss in detail.

2. We implement the industrial wheel loader product line with two modeling techniques. Comprehensive models and detailed modeling process explanation are presented.

The product line functional safety analysis covers three aspects:

1. Investigate the different safety analysis techniques and choose the fault tree analysis as the main technique.

2. Extend the single system fault tree to the product line fault tree.

3. Investigate the contributions of the product line modeling techniques to the product line functional safety analysis. Specifically, we map the product line models to the product line fault tree.

Furthermore, we evaluate the product line modeling techniques from their performance in domain analysis and safety analysis.
List of Figures

1. Feature tree symbols [1]. ......................................................... 7
2. Use case modeling example: wheel loader CDC supervision function. . 8
3. Feature modeling example: wheel loader steer by wire features. ........ 8
4. Static modeling example: wheel loader product line context diagram. . 9
5. Dynamic modeling example: CDC activate use case communication diagram. 10
6. Dynamic modeling example: CDC supervision control statechart. . . 10
7. Fault tree symbols [3]. ............................................................ 16
8. System fault tree layers [2]. ................................................... 16
9. Wheel loader software product line concept diagram. ..................... 19
10. CDC function overview. ....................................................... 20
11. Boom function overview. ..................................................... 21
12. Wheel loader software product line feature diagram. .................... 23
13. Wheel loader software product line CDC function use case model. . 26
14. Wheel loader software product line CDC supervision function use case model. 27
15. Wheel loader software product line CDC calibration use case model. . 27
16. Wheel loader software product line feature model. ....................... 28
17. Wheel loader software product line conceptual static model. .......... 31
18. Wheel loader software product line context model. ...................... 31
19. Communication diagram for the use case: CDC Activate. ............... 33
20. Communication diagram for the use case: Joystick Control in Four Directions. 34
21. Communication diagram for the use case: Gear Selection Button Control. 35
22. Communication diagram for the use case: Low(High) Speed Button Control. 37
23. Communication diagram for the use case: End Damping of joystick control. 37
24. Communication diagram for the use case: CDC Error Handling. ....... 38
25. Statechart for CDC Supervision Control and CDC Function Control objects. 40
26. Wheel loader product line fault tree. ..................................... 50
27. Wheel loader product line fault tree (cont’d). ............................ 51

List of Tables

1. CDC feature/use case dependencies of the wheel loader software product line. 30
2. CDC feature/class dependencies of the wheel loader software product line. 44
3. PHA worksheet for detailed and hazard analysis. The table is specified for the hazard analysis of function Steer by Wire. ................. 46
4. Modeling technique evaluation matrix. The models built from the corresponding modeling techniques are mapped to each layer of the wheel loader product line fault tree. ................................. 52
Product line modeling technique evaluation matrix for safety analysis. The criteria are proposed based on the computer system safety analysis criteria, with concern in the product line safety analysis.

Product line modeling technique evaluation matrix for domain analysis. The criteria are modified from the domain analysis methods evaluation criteria, which mainly consider the evaluation from product line modeling perspective.
## Contents

List of Figures ........................................... C
List of Tables ............................................ C

1 Introduction ........................................... 1
   1.1 Background ....................................... 1
   1.2 Problem Description .............................. 2
   1.3 Project Goals and Contributions ................. 2
       1.3.1 Project Goals ............................... 2
       1.3.2 Contributions .............................. 3
   1.4 Overview ........................................ 3

2 State of Literature .................................... 5
   2.1 Software Product Line Engineering ............... 5
   2.2 Product Line Modeling Techniques ................. 6
       2.2.1 Feature-oriented Modeling .................. 6
       2.2.2 UML-based Modeling ....................... 7
       2.2.3 Decision-oriented Modeling ............... 11
       2.2.4 SysML-based Modeling ..................... 12
       2.2.5 Modeling Techniques Discussion .......... 12
   2.3 Functional Safety Analysis ....................... 13
       2.3.1 Preliminary Hazard Analysis ............... 14
       2.3.2 Fault Tree Analysis ....................... 15

3 Industrial Use Case .................................... 18
   3.1 Preliminary Approach for Design Embedded Real-Time Product Line ... 18
   3.2 Wheel Loader Comfort Drive Control Function .......... 19
   3.3 Wheel Loader Boom Lifting (Lowering) Function ....... 20

4 Software Product Line Modeling Technique Evaluation .......... 22
   4.1 Software Product Line Modeling Example .......... 22
       4.1.1 Wheel Loader Software Product Line Feature Modeling .... 22
       4.1.2 Wheel Loader Software Product Line CDC Function PLUS Modeling 25
       4.1.3 PLUS-based Embedded Real-Time Software Product Line Design .. 44
       4.1.4 Functional Safety Analysis ................. 45
   4.2 Modeling Techniques Evaluation ................... 52
       4.2.1 Evaluation Matrix for Safety Analysis .......... 52
       4.2.2 Evaluation Matrix for Domain Analysis ...... 55

5 Conclusions ............................................ 58
   5.1 Safety-critical System Product Line Modeling .......... 58
1 Introduction

1.1 Background

In order to produce large number of cars at lower cost and unified design, Ford invented the production line in 1913, which enabled the production for mass market [4]. Software product line approach has been applied to software engineering domain to meet the increasing complexity of software development and software reuse in industrial context. Software product line methodology is a development paradigm that allows companies to meet business drivers like reduce the time to market and cost, improve the productivity and quality [5].

The outcome of software product line engineering methodology is cheaper and higher quality software products and software-intensive systems [6]. The software product line engineering paradigm is proposed to be separated into two processes: domain engineering and application engineering. A structured and managed reuse of different development artifacts is the main aspect of software product line engineering. In order to enable such reuse, the commonality and variability analysis is applied for finding what is common and what is different to specific set of products.

Though reuse is crucial for the software product line concept, it poses challenges in the safety domain [7]. Unmanaged and unstructured reuse may lead to accident. Because of insufficient impact analysis, in 1996, the European Space Agency (ESA) suffered the well-known launch failure of ARIANE 5, which reused features from ARIANE 4 that did not fit for its acceleration and trajectory. The reuse of unnecessary features resulted in the launch failure [8]. The National Aeronautics and Space Administration (NASA) suffered from the Mars Climate Orbiter (MCO) loss in 1999. The NASA’s team reused the software from the Mars Global Surveyor (MGS), a metric mishap for a key spacecraft operation led to the loss [9]. For domains where failures of software may lead to accident and loss of mission, the system safety needs to be taken into account when designing the product line. System safety requires the preventing or reducing accidents in the whole system life cycle [10]. System safety covers both the technical and organizational aspects of system engineering. The approach to build the safety-critical product line should keep the balance between the two aspects [7]. Habli [7] proposes that for each artifact that is reused in the products derived from the product line, the artifact safety analysis needs to cover the domain engineering where the artifact is developed, and the application engineering where the artifact is instantiated.

One part of system safety is the functional safety that aims to ensure the failure of functionality implemented in E/E systems do not introduce hazards to the complete system. Functional safety standards are created for guiding the safety-critical products development. When developing the product, the functional safety standards are used for arguing functional safety and assessing the chosen technical solution. There are different functional...
safety standards with different domain specific concerns. For instance, the functional safety standard IEC 61508 [11] is a generic standard and applicable for several domains, while the functional safety standard ISO 26262 [12] is specific for the automotive domain and EN 50128 [13] is specific for the railway domain.

1.2 Problem Description

Current functional safety standards do not support the use of product lines, instead they only concern the development of a safety-critical function in one specific product. When safety-critical function is developed for a product line, all different instantiations of product line members need to be considered in the safety life cycle required by the functional safety standard.

The identified hazards in the hazard analysis are traced through development and their adequate closure needs to be shown. Since the hazards relate to the products and their application, the hazard analysis for a product line should investigate all possible product line members.

It is necessary to provide sufficient information for the hazard analysis. In the case that important information for the possible products are unavailable, i.e., possible hazards, failures or faults might be missed. The lack of crucial information may lead to: (i) wrong design, (ii) delayed change identification in the software development or products failure due to the delayed failure identification. All the cases will lead to higher cost for a company - either in (i) when the design is changed at a later stage or in (ii), the delayed change and failure identification easily lead to accident.

1.3 Project Goals and Contributions

1.3.1 Project Goals

Based on the premise that: compared with other development methods, the model-based development provides more information to conduct the product line safety analysis. Consequently, the goal is to prove that the model-based development provides the evidence for safety certification of all products derived from the product line.

From product line modeling perspective, different modeling techniques are developed to model a software product line. The first goal of this work is to investigate different product line modeling techniques. As our second goal, we apply the chosen product line modeling techniques in an industrial case to gain practical information. The third goal is to compare all modeling techniques that are proposed in the first goal.

We are aiming to investigate how the used modeling techniques support the product line safety analysis, in particular, the Fault Tree Analysis. In this report the possibility to
derive necessary information from the product line model when building the fault tree shall be evaluated.

1.3.2 Contributions

This report contributes a deeper understanding regarding the methods of modeling software product line for safety-critical products. Different software product line modeling techniques are demonstrated and evaluated in terms of their usage in the functional safety analysis:

- This report presents different safety analysis techniques and investigate the possible technique for product line safety analysis.
- This report provides examples for building a software product line model with concern in the functional safety analysis. Elaborate information is provided for the industrial use case being modeled.
- The detailed description for building a product line fault tree is presented in this report. Moreover, how the modeling techniques support the fault tree is investigated.
- Different modeling techniques are evaluated by their contributions in both the product line functional safety analysis and domain analysis.

Finally, the possible extensions for a specific modeling technique is proposed, which makes the modeling technique more suitable for product line safety analysis.

1.4 Overview

The report is structured as follows:

In Section 2 both the background information about the product line modeling and safety analysis techniques are presented. The information on product line modeling cover some proposed techniques. Elaborate information for the safety analysis techniques - the preliminary hazard analysis (PHA) and the Fault Tree Analysis (FTA) are presented.

In Section 3, two machine functions implemented in the embedded system from a Volvo wheel loader are described. The description covers the purpose of the functions and the way how those functions are realized. Later on, those industrial cases are modeled by using some of the proposed methods and applied for functional safety analysis.

Section 4 presents the process to build the software product line model. By using two of the modeling techniques described in Section 2, we build the software product line models for the industrial cases provided in Section 3. Thereafter, the safety analysis is performed on those cases and the modeling techniques are evaluated from their contributions to the functional safety analysis.
In the end, Section 5 summarizes different product line modeling techniques, and state the further research directions.
2 State of Literature

Software product line engineering provides a way to develop different kinds of products or software-intensive systems at lower cost and a shorter time. We concern the functional safety aspects of a software product line, i.e., how to analyze the functional safety of the product family in an efficient and useful manner.

The section is organized as following: Section 2.1 introduces the concepts related to software product line engineering. Section 2.2 describes some of the software product line modeling techniques: the UML-based modeling, the feature-oriented modeling, the decision-oriented modeling, the SysML-based modeling. The discussion covers the concepts of those modeling techniques and their application in different phases of software product line engineering. Section 2.3 introduces instances of safety assessment analysis methods and discuss some of the methods in detail.

2.1 Software Product Line Engineering

In order to develop software products with higher quality and lower cost, improve the business efficiency and productivity, (a) the concept software product line engineering is proposed by Weiss et al. in 1999 [14].

“Software product line engineering is a paradigm to develop software applications (software-intensive systems and software products) using platforms and mass customization” [6].

The software product line engineering covers both the software products, software-intensive systems, and software that is embedded into a software-intensive system [6]. The main topic of the software product line is reuse. For a software product line, the strategic, planned reuse yields predictable results, i.e., business goals as efficiency and productivity, reduced time to market and higher quality [15].

In order to enable reuse, software product line engineering must exploit commonalities and variabilities. The definition of commonality and variability vary. Coplien et al. [16] define commonality and variability in terms of sets: "left A commonality is an assumption held uniformly across a given set of objects (S)" and “a variability is an assumption true of only some elements of S, or an attribute with different values for at least two elements of S.” Weiss and David [17] define commonality considering the domain engineering process: Family-oriented Abstraction, Specification and Translation (FAST). The FAST process performs commonality analysis in its early step, which decide the members of a family. Differ from the commonality and variability analysis performed in [16], Weiss and David [17] deem that “Variabilities define the scope of the family by predicting what decisions about family members are likely to change over the lifetime of the family”. Pohl et al. [6] define variability considering all types of artifacts. As the first one introduce the Feature-Oriented Domain Analysis (FODA), Kang et al. [18] analyze commonality and variability in terms
of features. The FODA is further applied in the Feature-Oriented Reuse Method (FORM) to develop a product line’s architectures and components [19].

The software product line engineering development process is divided into domain engineering and application engineering.

“Domain engineering is the process of software product line engineering in which the commonality and variability of the product line are defined and realized” [6].

“Application engineering is the process of software product line engineering in which the applications of the product line are built by reusing domain artifacts and exploiting the product line variability” [6].

In other words, domain engineering creates the base and develops the artifacts for application (i.e., the set of components and the mechanism to support their reuse) while application engineering “consists in analyzing, designing, building, customizing, or testing one product by reuse” [20].

A platform is a base of technologies where the other technologies can build on top of it. The software product line engineering platform provides the base for the development of software-intensive system and software products. For products that share the same platform and have similar features, they are grouped together as a product line or a product family.

2.2 Product Line Modeling Techniques

2.2.1 Feature-oriented Modeling

Feature-Oriented Domain Analysis (FODA) is firstly proposed by Kang et al. [18]. The aim of FODA is to distinguish the common and variable features so that they can be reused between different products. Domain analysis determines the requirements of the system and the way to implement the system. FODA provides the process of domain analysis. The FODA process includes three phases: context analysis, domain modeling, architecture modeling [18]. However, those three phases are deduced to: feature analysis and domain modeling in [1], where the architecture modeling is categorized into domain design phase.

Kang et al. [18] define a feature as a distinctive characteristic of a system, which is visible only to the end user. Czamecki [1] defines a feature as a distinguishable characteristic of a concept, which is relevant to some concept stakeholders. Later on, Lee and Muthig [21] specify the features as services, operations, non-functional characteristics and technologies of a product line. A feature model contains features of a software system, whereas the feature model is depicted by a feature diagram.

The symbols used in the feature diagram is shown in Figure 1. A feature tree is composed of many leaf nodes and a root node. The leaf nodes are feature nodes, the root node is
Methods for Modeling of Product Lines for Safety-critical Systems

a concept node. For the feature diagram examples shown in Figure 1, the leaf node is denoted as \( f \) and the root node is denoted as \( C \). The nodes are connected by edges, some of the edges are combined with edge decorations. Those edge decorations show the variant relationship between feature nodes, i.e., alternative features, or-features.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Name</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="symbol" alt="Mandatory Feature" /></td>
<td>Mandatory Feature</td>
<td>The feature is included if its parent is included</td>
</tr>
<tr>
<td><img src="symbol" alt="Optional Feature" /></td>
<td>Optional Feature</td>
<td>The feature may be included if its parent is included</td>
</tr>
<tr>
<td><img src="symbol" alt="Alternative Feature" /></td>
<td>Alternative Feature</td>
<td>Exact one of the feature from the alternative feature set is selected if the parent is included</td>
</tr>
<tr>
<td><img src="symbol" alt="Or-Feature" /></td>
<td>Or-Feature</td>
<td>At least one of the feature from the or-feature set is selected if the parent is included</td>
</tr>
</tbody>
</table>

Figure 1: Feature tree symbols [1].

The model proposed by Kang et al. [18] only considers the end user as the stakeholder while the model proposed by Czarnecki [1] considers more stakeholders, i.e., any parties involved in the system. Moreover, Czarnecki [1] re-categorizes the phases in feature-oriented analysis.
process, which is similar to the software product line engineering phases depicted in [22].

2.2.2 UML-based Modeling

Gomaa [22] proposes a method to design software product lines with UML, which is defined as Product Line UML-based Software engineering (PLUS) process. The PLUS process provides the concepts and techniques for building the software product line model. The model depicts the commonality and variability explicitly, which improves the reuse significantly.

Gomaa categorises the software product line engineering phases as: software product line requirement modeling; software product line analysis modeling; software product line design modeling; increment component implementation; product line testing [22]. Since requirement and analysis modeling are essential for developing safety critical product lines, we describes the first two phases in detail [22]:

2.2.2.1 Software Product Line Requirements Modeling

Use Case Modeling

In the requirement modeling for a single system, the system is regarded as a black box. The use case model is built based on the functional requirements of the system, i.e., the system response towards the stakeholder requests. The functional requirements are addressed in terms of the actors and the use cases. The actors are external entities that interact with the system, which can be categorized as: primary actor and secondary actor. Each use case defines part of the external requirements of the system.

For a product line, some use cases are required by all members of the product line. And some use cases are required by specific members of the product line. To distinguish the use cases, PLUS introduces stereotypes, like: kernel, optional and alternative use cases. The kernel use case is required by all the members of the product line, the optional use case is required by some members of the product line, the alternative use cases are normally mutually exclusive. An use case example is presented in Figure 2. We choose the wheel loader Comfort Drive Control (CDC) function as the example for explaining different PLUS models. The detailed wheel loader CDC PLUS model will be presented in the later chapter of this report. The CDC supervision function is one of the optional functions implemented with the wheel loader, the CDC Supervision Request includes the optional functional requirements: CDC Supervision and CDC Disable.
Feature Modeling

A feature is a requirement or characteristic that is provided by one or more members of the product line [22]. In our work, one feature is related to one characteristic of a product.

A PLUS feature model is shown in Figure 3, the Machine Movement feature is a common feature realized by the steer by wire system, which includes the common feature Steer by Steering Wheel or the optional feature CDC Function.

Figure 3: Feature modeling example: wheel loader steer by wire features.

The use case model analyses the functional requirements of a product line, the feature model maps the functional requirements to the features, which are modeled for identifying the common and variable features of the products. Features and use cases have the many-to-many association relationship: a functional feature can be modeled as a group of use cases that are reused together; feature dependencies can be depicted by the dependency of use cases that belong to the feature; the variation points in a use case can be modeled as variable features or parameterized features.
2.2.2.2 Software Product Line Analysis Modeling

Static Modeling

The static model describes the static structure of a product line, which can be used for the commonality and variability analysis of the product line real-world classes in problem domain. The software product line boundary is determined from the static model with real-world physical classes. The boundary between the software product line and the external environment is referred to as a software product line context [22]. It is important to distinguish the context of a software product line, because the system needed to be interpreted in the system context to find the hazards [23]. There are two types of software product line context model: (i) model from the system perspective: the physical devices and software are part of the system, the user is external to the system; (ii) model from the software perspective: the physical devices are external to the system [22]. The static model is described by the UML class diagram. The external system is depicted as external classes. The external class is classified as an external user, an external device, an external system or an external timer. The external device is further categorized as external input device, depicting the device providing input to the system; external output device, depicting the device receiving output from the system; external input/output device, depicting the device receiving output and providing input from the system.

A context model is shown in Figure 4. The context model extracts information from the wheel loader software product line context model. In Figure 4, the context of the product line system includes one external input device, one external output device, one external input/output device and one external system. From reuse point of view, all those devices are kernel.

![Figure 4: Static modeling example: wheel loader product line context diagram.](image-url)
Dynamic Modeling

Dynamic model describes the control and sequencing of a system. The dynamic model is composed of two types of model: the dynamic interaction model address the interaction between objects.

The dynamic interaction model is based on the use case depicted in the requirement model. In the dynamic interaction model, objects\(^1\) communicate with each other to address the functions required by the use cases. A specific message path through a use case is called a scenario. The objects involved in the use case are determined by the object structuring criteria. There are two approaches for determining the classes that the objects instantiated: the kernel first approach, with which the dynamic analysis is performed on kernel use cases, which are needed by every product line member; the evolution approach, with which the optional and alternative use cases are depicted, those use cases are needed by some product line members. The classes involved in each of the approach is categorized by the class structuring criteria.

A communication diagram is presented in Figure 5. The communication among objects realize the use case CDC activate. The outside devices interact with the system via the device interfaces. The CDC Data entity contains the information relates to the CDC function.

The other type of PLUS dynamic model is finite state machine. A finite state machine is a conceptual machine that contains a finite number of states. The change between states is referred to as state transition, which is caused by an input event. The state transition diagram depicts the finite state machine. In object-oriented model, some of the objects are state dependent control objects, indicating the objects encapsulate state transition diagrams. The state transition diagram depicts control and sequencing view of a system. In UML notation, a state transition diagram is referred to as a statechart diagram, we simplify it as statechart from now on.

An object with stereotype state dependent control in a communication diagram hides the details of a state machine, which is depicted by a statechart. The statechart needs to be consistent with the communication diagram, i.e., an input on the communication diagram must be identical to an input event on the statechart, alternatively, an output on the communication diagram must be identical to an output event on the statechart. The equivalent communication diagram message and the statechart event must be given the same name.

A statechart example is presented in Figure 6. The statechart depicts the state transition during the wheel loader CDC supervision control. The wheel loader supervision function monitors the wheel loader CDC function operation, i.e., if error happens during CDC operation, the supervision control disables the CDC function. The statechart transits

\(^1\)An object is a real-world physical or conceptual entity. An object is a single “thing” while a class is a collection of objects [22].
from the CDC Monitor Available state to the CDC Monitor Activated state after CDC activating. If error happens, the statechart then transits to the CDC Disabled state. When the user restarts the CDC, the statechart transits back to the CDC Monitor Available state.

Figure 5: Dynamic modeling example: CDC activate use case communication diagram.

Figure 6: Dynamic modeling example: CDC supervision control statechart.
2.2.2.3 Software Product Line Feature/Class Dependency Modeling

The feature model groups the functional requirements together, it is used for enabling the reuse. The static model is used for analyzing the real-world classes in problem domain, i.e., the static structure of the classes and their associations, whereas the product line functionality is realized by those classes. The feature/class dependency model addresses the relationship between the feature model and the static model, it shows the classes that support each product line feature. In order to analyze the feature/class dependency, the relationship between features and objects have to be determined in advance. In the communication diagram, the use case is supported by objects. Since a feature can support one or more use cases, the objects that support the feature are determined from those use cases, a feature-based communication diagram is introduced to model the relationship between features and objects, i.e., the objects provide the functionality specified by the feature. For the objects in the feature-based communication diagram, the classes that instantiate the objects are determined. Consequently, the feature/class dependency is derived from the feature/object dependency, the relationship is depicted in a feature-based communication diagram, where the features are depicted in the form of packages containing the classes support each feature.

Riebisch et al. [24] extend UML to model the system families. The concept is based on the feature diagram. The model depicts the variants during system analysis and design. Compare the model proposed by Riebisch et al. with the PLUS model, the former one confuses the variants and lack the variants relations. Moreover, the former model misses depicting requirement and parameter.

2.2.3 Decision-oriented Modeling

Decision-oriented modeling is used for modeling the product line variability. A decision is usually an information-intensive value, making a decision is the same as choosing the ways for achieving a goal at a certain time [25]. Unlike feature-oriented modeling, decision-oriented modeling has not yet reached a common notion for the decision, researchers proposes different approaches for building the decision model:

Kaeding [26] introduces the decision table: the product line variability information is captured in a table. Each row of the table represents the variability, the information is presented in textual form, which make the table understandable. However, Maga and Jazdi [27] point out that the table cannot support formal proofs. Moreover, the table cannot model the interdependencies between different variants, and the table becomes confusing when it contains complex information.

The decision tree is an improvement to the decision table. The decision tree depicts the variants in with graphical symbols. Maga and Jazdi [27] mentions that the decision tree is intuitive and understandable. However, the decision tree may yield redundancy when
there are numerous variants. On the other hand, decision tree cannot capture some crucial interdependency between variants [27, 28].

Schmid and John [29] introduce the variability meta-model to manage the variability across different life cycle stages. The model is independent of the specific notation. Dhungana et al. [30] introduce the product line variability model which is suitable for different domains, the approach is supported by the Decision-Oriented Product Line Engineering for effective Reuse (DOPLER) meta-tool.

Czarnecki et al. [31] compares the feature model with the decision model. They mention that the main difference between the feature model and the decision model is: the feature model supports both the commonality and variability modeling, while the decision model only supports the variability modeling. In that sense, the feature model provides the base for the decision model, i.e., the feature model is used in the domain engineering for providing the artifacts for the application, the decision model is used in the application engineering which deliver the specific activities.

### 2.2.4 SysML-based Modeling

Habli et al. [32] uses SysML to model the complex safety-critical systems. We take the modeling technique into account because it is similar to the PLUS, meaning that it has the potential to be used for modeling the software product line. The SysML model is composed of: context and feature models, which describe the environment where the system executes and the functions provided by the system; system requirements models, which model the functionality of the system; system structure models, which describe the components and their communication; system behavior models, which depict the behaviors of the system, specified in the behavior of each function.

Maga and Jazdi [27, 28] introduce an approach to model the variants of industrial automation systems in a product line by using SysML. The main concept of the technique is inheritance and package modeling [27]. The system elements are modeled as packages. Those packages needed by each product member is defined as mandatory packages, while the packages needed by some of the product members are defined as optional packages. with the SysML-based modeling, and get the conclusion that the SysML-based modeling provides a comprehensive support for the variants modeling.

### 2.2.5 Modeling Techniques Discussion

In the feature model proposed by Kang et al. [18] and Czarnecki [1], variation points of product line are limited to features, which can not cover all the assets of the product line. The issue is addressed by the PLUS for it extends the UML stereotypes. As a result, the PLUS can cover the feature, requirement and implementation aspects of the product line,
which in turn show the static and behavior aspects of the models and the way they are affected by the variation points.

On the other hand, compare to the SysML-based modeling, the PLUS process is more suitable for software product line modeling as well, the reasons are: (i) the SysML-based modeling technique proposed by Habli et al. [32] covers most of the system modeling aspects. However, from the software product line modeling perspective, their model does not provide enough information as the PLUS does. (ii) According to [27], UML-based model proposed in [24] is rooted from the concept of the feature diagram. Therefore, the model has the flaws like: for numerous variants, the model becomes confusing and hard to maintain consistency; the model cannot depict the relations between variants; requirement and parameter diagrams are not supported. All those disadvantages have been conquered by the PLUS [22]. (iii) The SysML-based modeling method proposed by Maga and Jazdi is suitable for variability modeling. Nevertheless, a product line model should cover both the variabilities and commonalities. From this perspective, the PLUS process provides better support for the software product line modeling.

Based on the preliminary discussion above, we decide to model the product line with the PLUS. As the comparison, we choose the feature-oriented modeling, which is depicted by the feature diagram. The detailed modeling process with those two techniques will be presented in the later section.

Moreover, we evaluate the four modeling techniques mentioned above from two aspects: (i) From the safety analysis point of view, we evaluate the modeling techniques with the criteria: clear role in safety analysis, valuable result for safety analysis, modeling process is time-benefit, modeling starts on product line level, traceability, model contains context information, model contains hardware information and model contains software information; (ii) From the domain analysis point of view, we evaluate the modeling techniques with the criteria: extraction of information, ease to build model, tool support, flexibility, maintenance, the role the product line model play in the software development process, i.e., requirements modeling, analysis modeling, design modeling, implementation, testing. The criteria assess the modeling techniques considering the required information for modeling, the product line modeling process and the outcome of the product line modeling. The detailed comparison follows after the discussion of product line modeling process.

### 2.3 Functional Safety Analysis

Safety is a property of a system, which is “the expectation that a system does not, under defined conditions, lead to a state in which human life or the environment is endangered” [23]. Functional safety is one kind of safety, which concerns the malfunctioning behavior of E/E systems [12].

The main principle of functional safety concept is to eliminate, reduce or control hazard. A hazard is “a physical situation, often following from some initiating event, that can lead to
an accident” [23]. Hazard may lead to accident, which is “an unintended event or sequence of events that causes death, injury, environmental or material damage” [23]. A fault is a “adjudged or hypothesized cause of an error” [33], which may lead to hazard.

In ISO 26262, the functional safety concept is verified by its compliance with the safety goals, and its competence in mitigating or avoiding the hazardous events [12].

In a specific environment, a failure could lead to hazards. A failure is an “event (transition) that occurs when the delivered service deviates from correct service (the system specification)” [33].

A fault may lead to errors, which in turn may lead to failures [34]. An error is defined as a “part of the total state of the system that may (in case the error succeeds, by propagating itself, in reaching the external system state) lead to its subsequent service failure” [33].

A risk is defined in [23] as “the combination of the frequency, or probability, and the consequence of an accident” [23]. In ISO 26262, the Automotive Safety Integrity Levels (ASILs) for determining the risk classes with respect to the severity, exposure and controllability of the hazard [12] is specified. In IEC 61508, this level is defined as SIL.

The type of safe analysis techniques vary. A broad range of hazard analysis techniques are proposed in literatures and by practitioners. Generally the safety analysis methods can be categorized as: top-down analysis techniques, bottom-up analysis techniques, Common Cause Analysis (CCA) and Common Mode Analysis (CMA) [3].

In industry, the Preliminary Hazard Analysis (PHA) is widely used. Moreover, the top-down analysis technique - the Fault Tree Analysis (FTA) and the bottom-up analysis technique - the Failure Modes and Effects Analysis (FMEA) are commonly used in industry as well. We only study the PHA and FTA in detail in this report.

2.3.1 Preliminary Hazard Analysis

The PHA identifies basic information for hazards, hazards cause factors, hazard casual factors, consequences and relative risk associated in the design phase of system development [2]. The purpose of the PHA is to identify the hazards and effect on the design as early as possible.

The PHA process is provided by [2]:

1. Define system. Define the system boundary. Define the activity and activity phases, determine the system environment.

2. Plan PHA. Identify the system functions.

3. Establish safety criteria. Identify the design safety criteria, the applicable safety percepts and safety guidelines.
4. **Acquire data.** Acquire data needed by the PHA. Those data includes: design, operational, and process data, hazard information, regulatory data etc.

5. **Conduct PHA.** Construct list and worksheet for analysis, compare the system status with the corresponding check lists. Consider functional relations within the system when identifying hazards, utilize the hazard/mishap information obtained from other systems. Normally, the PHA worksheet includes information needed by the analysis, like the name of the function being identified, the condition where the product works, the hazard being evaluated etc. An example of the worksheet is presented in the later section.

6. **Evaluate risk.** For each identified hazard, determine the mishap risk with and without hazard mitigations in the design phase.

7. **Recommend corrective action.** Recommend the crucial action to mitigate or eliminate the identified hazards.

8. **Monitor corrective action.** Ensure the safety recommendation action is efficiently mitigating the hazards by reviewing the test result.

9. **Track hazards.** Identify new hazards.

10. **Document PHA.** Make PHA report, which includes the entire PHA process and PHA worksheets, conclusions and recommendations.

Among all those PHA analysis steps, the most important step is step 1 and step 4, which defines the boundary of the system and data acquired by the system. In our work, we derive the information needed by those steps from the product line model.

### 2.3.2 Fault Tree Analysis

The Fault Tree Analysis (FTA) is a widely used deductive safety analysis technique [23], which tracks the specific causes of a undesired problem. We will describe the application of FTA in detail in this report.

The FTA provides a method for analyzing and evaluating failure path on system level, where the fault tree is used to detect the possible cause of an undesired event. In the system development process, the FTA can be used in the design phase of system development, which is applied to find the possible problems as early as possible [2].

The FTA covers the quantitative and qualitative aspects of the safety requirement. In ISO 26262, the functional safety requirement is defined as a:

“Specification of implementation-independent safety behavior, or implementation-independent safety measure, including its safety-related attributes” [35].
The qualitative safety requirement covers the relationship between events and their contributions to the top event; the qualitative safety requirement derives the probability a top event happen from the probabilities of the sub-events happen. In the safety projects performed for Ford, the FTA is applied in the safety concept phase and the corresponding safety requirements [36].

Considering the structure of a fault tree, the fault tree includes a *top event* which is the undesirable event, the statement of top event should include: what the fault is, where and when the fault occurs [3]. Except the top event, there are other events and conditions, which lead to the occurrence of the top event. Those intermediate events or conditions must have immediate effect to the top event. Moreover, if the intermediate event cannot be decomposed any more, it is denoted as a *basic event* [23]. Events in fault tree are connected by logic symbols. Different fault tree symbols are shown in Figure 7: the box, events and logical gates are mainly used for building a fault tree. The top event is depicted in the description box, and the intermediate events as well. Basic Event and Undeveloped Event are usually located at the bottom layer of the fault tree, indicating that the event cannot be further decomposed. The boxes and events are connected by gates, including AND-Gate, Priority AND-Gate, OR-Gate. Those gates indicating the logic relationship between events. If the fault tree is divided into different parts, those parts are connected by the Transfer symbol.

The fault tree is developed in layers, levels and branches [2], the main layer of a system fault tree is shown in Figure 8. For a single system fault tree, the top layer usually depicts the system functions and phases, the hazard described by the root node on this layer is derived from the PHA; the middle layers depict the different mission phases, i.e., dynamic aspect of the fault tree, and the subsystem fault flows; the bottom layer models the system component fault flows.

![Figure 8: System fault tree layers [2].](image-url)
<table>
<thead>
<tr>
<th>Symbol</th>
<th>Name</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>□</td>
<td>Description Box</td>
<td>Description of an output of a logic symbol or of an event</td>
</tr>
<tr>
<td>□</td>
<td>AND-Gate</td>
<td>Boolean Logic gate - event can occur when all the next lower conditions are true</td>
</tr>
<tr>
<td>□</td>
<td>Priority AND-Gate</td>
<td>Boolean Logic gate - event can occur when all the next lower conditions occur in a specific sequence (sequence is usually represented by a conditional event)</td>
</tr>
<tr>
<td>△</td>
<td>OR-Gate</td>
<td>Boolean Logic gate - event can occur if any one or more of the next lower conditions are true</td>
</tr>
<tr>
<td>□</td>
<td>Inhibit</td>
<td>Output fault occurs if the (single) input fault occurs in the presence of an enabling conditional event.</td>
</tr>
<tr>
<td>□</td>
<td>Transfer</td>
<td>Indicates transfer of information</td>
</tr>
<tr>
<td>□</td>
<td>Basic Event</td>
<td>Event which is internal to the system under analysis, requires no further development</td>
</tr>
<tr>
<td>□</td>
<td>House</td>
<td>Event which is external to the system under analysis, it will or will not happen (Pf=1 or Pf=0)</td>
</tr>
<tr>
<td>□</td>
<td>Undeveloped Event</td>
<td>Event which is not developed further because it has little impact on the top level event or because the details necessary for further event development are not readily available</td>
</tr>
<tr>
<td>□</td>
<td>Conditional Event</td>
<td>A condition which is necessary for a failure mode to occur</td>
</tr>
</tbody>
</table>

Figure 7: Fault tree symbols [3].
In order to model different types of systems and to fulfill the different demand of safety analysis, there are many extensions to the FTA: Software Fault Tree Analysis (SFTA). SFTA is a traditional safety analysis technique:

“SFTA is a method for identifying and documenting the combinations of lower-level software events that allow a top-level event (or root node) to occur” [37].

SFTA can be used for the product line safety analysis, once a product line SFTA is established, it can identify the additional product line constraints, which minimize the effort to derive a SFTA [7]. To be specific, the reuse of SFTA improve the product line safety analysis by deriving the reusable assets for new product line member [38, 39, 40].

The dynamic FTA is developed intentionally for the computer-based system, in [41], the fault-tolerance computer system includes “a fault-tolerant parallel processor, a mission avionics system” and “a fault-tolerant hypercube”. The dynamic fault tree combines the FTA with Markov analysis for sequence-dependent analysis [2]. Dugan et al. [41] describe the dynamic FTA for handling reliability of fault-tolerant computer systems, and complex fault & error recovery techniques. In [42], the authors use a fault tree to model failure probability and feature dependencies of systems, in which the Markov method is used to solve the dynamic and dependent sections of the fault tree. Dynamic FTA is accurate in calculating the final probability, however, it still has the shortcomings that it is hard to understand, it is software dependent and not fit for fault-tolerant computer system analysis [43].

The software product line engineering delivers the safety-critical software products or software-intensive systems, thus the safety analysis should be applied in the context of product line. The product line is modeled by using the modeling techniques, whereas the product line model provides information for the safety analysis. We will present the application of the product line modeling techniques and the safety analysis techniques in detail in the later section.
3 Industrial Use Case

In this section we show some industry cases which are further used as the objective of software product line modeling and safety analysis. Those cases depict the functions performed by Volvo wheel loader\(^2\).

The section is organized as following: Section 3.1 presents the preliminary idea for design the embedded real-time product line. Section 3.2 describes the Comfort Drive Control (CDC) function. Section 3.3 describes the Boom Lifting (Lowering) function, which is one of the basic functions provided by the boom. In both sections, first we present the purpose of the function; then we explain the general principle of each function. The CDC function and the Boom Lifting(Lowering) function presented in this report are hypothetical functions implemented with the wheel loader.

3.1 Preliminary Approach for Design Embedded Real-Time Product Line

For the product line member derived from the wheel loader product line, each product line member has identical E/E architecture and software architecture. The product line software concept diagram is shown in Figure 9. Normally, a product line member has five computer nodes, i.e., ECUs. Due to the variability, some of the product line members have more than five ECUs. The extra ECUs are used for handling the extra requirement, e.g. air conditioning. Some of the product line members have less than five ECUs because they combine some of the ECU functions. The product line member shares the same AUTOSAR based platform. The application is built on top of the platform. The platform and the application software compose of the system main software.

The main software is realized in the form of software components. The product features are mapped to those software components, the components attributes are specified in configuration files. The software component configuration files include the communication and synchronization of the software components. The time budget should be considered in software components implementation. The component time budget constraints the maximum execution time of the software component, normally, it is decided from experience [44].

The product member has identical common configuration files and source code files. The product individual behavior is defined by datasets, which are sets of parameters controlling the program flow. The dataset is built on top of the main software, different parameter settings determine different functions.

\(^2\)Neither Volvo nor the author shall be responsible for the accuracy or liable of the information provided by the case study. Neither Volvo nor the author shall be responsible for any indirect or consequential loss related to the information contained in the case study.
3.2 Wheel Loader Comfort Drive Control Function

In order to make the wheel loader operation more comfortable and increase the productivity, the CDC function is introduced. Instead of using the steering wheel, a joystick is used to steer the machine. In that case, the user can steer the machine by easily moving the joystick, which releases the user from tedious arm movement when using the steering wheel. The CDC function fits for the short cycle working very well, e.g. operating on the work site. Figure 10 shows an overview of the CDC function, which is realized by the communication between different functional blocks. Considering design the real-time embedded system, the functional blocks can be assigned with tasks and the communication is organized by the scheduling protocol. The CDC function described in this report is a hypothetical function, which is based on the Volvo wheel loader CDC function, but extra functions are added to the existed CDC function. The presented hypothetical CDC function could be a development direction of Volvo wheel loader.

In order to use the CDC function, a user has to activate the CDC function first. The activation process is composed of two steps: first the user puts down the armrest, then pushes the activation button. The CDC can be successfully activated only when both of the steps are performed properly. The activation request and armrest status are fed into a CDC Function Control (in this work, CDC Function Control and Function Control are interchangeable), which sends out the armrest status and activation request to the Drive Line Monitor. This monitor responds the CDC Function Control with the gearbox status. If the gearbox is in the right status for activating CDC function, the CDC function is activated.

After activating the CDC function, the user controls the machine by either the joystick or the left (right) control lever. The left (right) control lever and the joystick are mutually
exclusive implementation choices, meaning that if the CDC function is installed, the machine can only have either the left (right) control lever or the joystick. In the case that the left (right) control lever is implemented, there are extra speed buttons needed for control the speed of the machine. There are two types of speed buttons: the gear selection button, with which the user can switch between different gears by simply press the button; the low (high) speed button, which is the simplified version of the gear selection button, i.e., instead multiple gears, there are only low (high) speed gear. Since the joystick control covers the information flow in both the joystick control case and the left (right) control lever case, we choose to present the principle of joystick control only.

There are two kinds of operations involved in the CDC function with joystick control: one is steering and the other is forwarding (reversing). As shown in Figure 10, the blue line represents the signal flow for CDC steering and the red line represents the signal flow for CDC forwarding (reversing). When the user starts using the joystick to control the machine, the CDC Function Control gets the joystick input and generates the corresponding signal (either CDC Steering Output or CDC Forwarding (Reversing) Output in Figure 10). An extra control block is needed to monitor the CDC input and output signal, which is the CDC Supervision Control here. The control block compares the CDC Forwarding (Reversing) signal with the CDC Forwarding (Reversing) Output signal, the CDC Steering signal with the CDC Steering Output signal. Once those signals do not match with each other, indicating an error happens, the CDC Supervision Control cuts the Relay or Logic circuit which is enabled in CDC activation step. On the other hand, it is also possible that even though no input is detected from the joystick, the CDC Function Control block generates the control signal. In that case, the relay or the logic circuit is cut by the CDC Supervision Control as well and the user shall be informed via the Display Unit.

During the machine operation, before reaching the maximum steering angle, the machine should reduce the speed so that it can arrive the end position smoothly. The operation that the machine reduces its speed before reaching the maximum angle is called end damping or end position retardation.
3.3 Wheel Loader Boom Lifting (Lowering) Function

Boom Lifting (Lowering) function is one of the mandatory functions implemented on every wheel loader product. The boom carries loads in the vertical direction. The user manipulates the boom to lift or lower the loads. The boom function aims at controlling the boom to lift or lower. Figure 11 shows an overview of the boom function.
The user uses levers to control the boom. There are two types of levers: multi-lever and single lever. Multi-lever is the default implementation and single-lever is alternative. The lever request is fed into the Boom Function Control (since it is identical with the CDC Function Control, the Boom Function Control and Function Control are interchangeable in this work). The control block generates the Lifting Order and Lowering Order, which are passed to the Hydraulic System. The boom lifting (lowering) signal (invisible in Figure 11) is the same type of signal as the steering signal of the CDC function. Like the CDC function, boom performs end damping so that the machine reaches the maximum angle smoothly, whereas the angle information is provided by the Boom Angle Detector.

In the boom lifting (lowering) operation, if no input signal detected in the situation that error happens, the Boom Function Control disables the boom function and the user shall be informed via the Display Unit. Since the CDC steering signal is the same type as the boom lifting (lowering) signal, it is possible that the CDC steering functional safety is effected by the boom function, we will address this issue further.

Except the boom lifting (lowering) function, some other functions are implemented with the boom as well, like boom float, boom suspension, boom kick-out. As the example, we only describe the boom lifting (lowering) function in detail.

Both cases depicted in this section present the functions of the Volvo wheel loader. In Section 4, we will present the process of building the wheel loader software product line model in detail and apply the safety analysis on those functions as well.
4 Software Product Line Modeling Technique Evaluation

We have presented two industrial cases in Section 3, which are in the following applied for building a software product line model by using the modeling techniques presented in Section 2.2.

Section 4.1.1 and Section 4.1.2 present the industrial cases software product line model. The embedded real-time software product line design is described in Section 4.1.3. Thereafter, the functional safety analysis is applied for the industrial cases based on the information provided by the model, as described in Section 4.1.4.

Section 4.2 provides the detailed discussion on modeling techniques evaluation. In Section 4.2.1, an evaluation matrix is presented for comparing different modeling techniques from safety analysis perspective. Section 4.2.2 presents the modeling techniques evaluation matrix from domain analysis perspective.

4.1 Software Product Line Modeling Example

4.1.1 Wheel Loader Software Product Line Feature Modeling

By using the feature description in [1], we build the feature diagram based on the two cases described in Section 3: Boom Lifting (Lowering) function and CDC function. For those two functions, the Boom Lifting (Lowering) function is a mandatory function which is default implemented during the wheel loader manufacturing; the CDC function is an optional function which is implemented based on the customer’s choice. The feature diagram of the wheel loader software product line considering both functions is shown in Figure 12.
Figure 12: Wheel loader software product line feature diagram.
The root node of the feature diagram is the concept node Wheel Loader Software Product Line.

The first layer below the root node describes the general functions of the product family, see Figure 12. In the wheel loader case, the first layer contains three functions: Steering Wheel, Boom Function and CDC Function. The steering wheel feature relates to the default characteristic that the user can control the machine by using the steering wheel. The functions are represented as features in the feature diagram. Among all those three features, the Steering Wheel and Boom Function are mandatory features, the CDC Function is optional feature. Each member of the product line has the characteristics Steering Wheel and Boom Function, some of the members have the characteristic CDC Function.

For the wheel loader software product line, the second layer contains the product family functions relate to each of the parent feature, see Figure 12. The functions are represented in the form of feature nodes:

- For the Boom Function node, the feature nodes belong to it are: Boom Float, Boom Suspension, Boom Lifting(Lowering) and Boom Kick-Out. The Boom Float is provided when the machine travels on the public road, other features are provided when the machine operates on the work site.

  For the features belong to the Boom Function feature, all of them have the end user as the stakeholder.

- For the CDC Function node, the features belong to it are: Activate, Supervision, Maintenance, Steering, Moving Forwarding(Reversing), End Damping and Handle Error. Only when the CDC function is selected, those features are provided by the machine. Therefore, as features belonging to the parent node CDC function, the sub features are mandatory features for the parent node. But consider from the software product line perspective, those features should be optional features.

  One difference between the feature model proposed by [18] and the model prose by [1] is: the later one considers more stakeholders than the end user. For all the features belong to the CDC Function feature, the Activate, Steering, Moving Forwarding(Reversing), End Damping and Handle Error features have the end user as the stakeholder. The Maintenance feature has the maintenance engineer as the stakeholder. The Supervision feature has the supervision unit as the stakeholder.

On the third layer of the feature diagram, see Figure 12, there are features belonging to the second layer features. In the wheel loader case, similar to the second layer, the third layer models the product family functions.

- For the features relate to the boom function, we only present the features which connect to the Boom Lifting(Lowering) feature. Those features are: Activate, Lifting(Lowering), End Damping, Lever Function, Handle Error, with the
end user as the stakeholder; Maintenance with the maintenance engineer as stake- 
holder.

• For the CDC function, the parent node Maintenance contains the features: 
  Set Maximum Lever Current, Set Maximum Angle Range, CDC and CDC 
  Supervision Installed and Read Out Machine Status via Service Tool, those 
  features have the maintenance engineer as the stakeholder. The parent node 
  Steering has alternative sub features: Lever Control Left(Right) and Joystick 
  Control in Four Directions, which have the end user as the stakeholder. The 
  Moving Forwarding(Reversing) feature has alternative sub features: joystick 
  Control in Four Directions and Speed Button, the end user is the stakeholder 
  of those two features as well. Since the Speed Button is implemented with the 
  Lever Control Left(Right) feature, we extend the feature relationship provided 
  by [1], define such kind of relationship as requires. The End Damping parent 
  feature has alternative sub features: Small Machine End Damping and Big Machine 
  End Damping, which have the end user as the stakeholder.

In the wheel loader case, the fourth layer of the feature diagram describes the product 
family functions as well, see Figure 12. The features on the fourth layer of the feature 
diagram are as following:

• For the boom lifting (lowering) function. The End Damping feature on the third 
layer has alternative sub features on the fourth layer: Big Machine End Damping 
and Small Machine End Damping. Each of the feature is mandatory feature, both 
of the features have the end user as the stakeholder.

The Lever Function has alternative sub features Multi-lever with Lever Hold 
and Single-lever without Lever Hold. The former one is default implemen-
tation, the later one is alternative. Those features has the end user as the stakeholder.

The Maintenance feature has Set Maximum Lever Current, Set Maximum Angle 
Range and Read Out Machine Status via Service Tool sub features. Those 
features are mandatory features for the Maintenance feature, meaning that those 
features can coexist.

• For the CDC function, the Speed Button feature has sub features Gear Selection 
  Button and Low(High) Speed Button alternative sub features. In the case that 
  the left (right) control lever is implemented, one of the speed button should be 
  implemented as well.

4.1.2 Wheel Loader Software Product Line CDC Function PLUS Modeling

Based on the PLUS process [22], we build the wheel loader software product line model 
with concern the CDC function. The modeling tool we choose is the Enterprise Architect. 
The way to build the model is explained in detail in the following part.
### 4.1.2.1 Software Product Line Requirements Modeling

#### Use Case Modeling

The use case modeling describes the functional requirements of the product line system. All the product members provide the commonality (defined as kernel use case), while some product members provide the variability. The variability can be captured by either the variation points of the kernel use case or being defined as optional or alternative use case. In UML, use case model is depicted by the use case diagram.

In order to build the wheel loader product line CDC function use case diagram, we start by identifying the actors and the use cases. There are two types of actors needed to be considered: (i) the primary actor who initiates the use case of the system; (ii) the secondary actor who joins the use case later [22]. The use case describes the interactions between the actor and the system, therefore, the use case should provide a value to the actor. In the CDC case, we mainly consider the end to end functions the system provides to the actor. The use case diagrams are shown from Figure 13 to Figure 15.

![Use Case Model](image.png)

**Figure 13:** Wheel loader software product line CDC function use case model.
Methods for Modeling of Product Lines for Safety-critical Systems

Since the use cases are organized in compliance with the most common sequence of interactions between the actor and the system. In the CDC case, we start by analyzing the common sequence of the interactions between the actor and the system. In our analysis we have identified three types of actors: the User, the CDC Supervision Unit and the System Maintenance Engineer.

As shown in Figure 13, if the customer wants the CDC function, the Maintenance Engineer install the CDC function with the CDC supervision function, which is described by the use case CDC and CDC Supervision Installed. To use the CDC function, the user should activate the function first, as shown in the use case CDC Activate.
After activating, the User can steer the machine by using the lever. There are two types of lever: one is used only for left (right) steering, the other is joystick, with which the user can steer the machine or drive the machine forwarding (reversing). The corresponding use case relate to the lever are: Lever Control Left(Right) and Joystick Control in Four Directions. The left (right) control lever should be implemented together with the speed button. The use cases related to the speed buttons are: Gear Selection Button Control and Low(High) Speed Button Control. The machine movement can be either CDC Steering, depicting the steering movement of the machine, or CDC Moving Forwarding(Reversing), depicting the forwarding (reversing) of the machine. Once the machine reaches the maximum angle during steering, it should perform the End Damping so that the machine can reach the end positions smoothly. Other than using the optional CDC function, the User can use steering wheel or gear lever instead, those are default functions of the machine, as shown in the use case Steering by Steering Wheel and Gear Lever Operating. If some error happens during the CDC operating, the CDC Supervision Unit initiates the CDC Error Handling use case, at the same time, the User is informed. The mainly cause of error are: the input request cannot be detected, or the input does not match with the output. The User is the primary actor for all the use cases related to the basic CDC functions except the use case CDC Error Handling, for which the CDC Supervision Unit is the primary actor and the User is the secondary actor.

The use cases relate to CDC supervision are shown in the Figure 14. Besides the use cases presented in Figure 13, more use cases are needed by the CDC function. The CDC Supervision Unit initiates the supervision function, as shown in the use case CDC Supervision Request. The supervision request includes both the CDC Supervision which monitors the CDC operating, and the CDC Disable which disables the CDC function when error occurs. The Maintenance Engineer is the primary actor for all the use cases relate to CDC supervision.

The use cases relate to CDC calibration are show in Figure 15. The Maintenance Engineer involves in the Read Out Machine Status via Service Tool and CDC Maintenance. The CDC Maintenance includes CDC and CDC Supervision Installed, Check CDC Install Status and CDC Calibration. The calibration operating includes the CDC Steering Angle to Service Tool and CDC Input Request to Service Tool. The CDC Supervision Unit is the primary actor for all the use cases related to CDC calibration.

The PLUS use case diagram presents the functional requirements of the wheel loader steering system. We derive the functional requirements mainly from the end to end functions of Volvo wheel loader, but not limited to them. The functional requirements depicted by the use case model can be used as the input for safety analysis. For instance, the CDC use case depicts the wheel loader CDC function, the malfunction of the CDC function may lead to hazards for the machine. In the case that we identify the functional requirements from the use case diagram, we can also identify the possible hazards relate to the functional requirements from the use case diagram.
Feature Modeling

Feature model aims at modeling the commonality and variability of the product line. The software product line features are result of the commonality and variability analysis, thus, from the reuse perspective, features can mainly be categorized as common, optional and alternative. A common feature is provided by each product member, optional and alternative feature is provided by some of the product member. Alternative feature are normally mutually exclusive features. In UML, the feature model is depicted by the class diagram. The feature model is shown in Figure 16.

In the CDC case, from the whole software product line perspective, the feature Machine Movement is sub-divided into two features Steering by Steering Wheel and CDC Function. The Steering by Steering Wheel is a common feature while the CDC Function is an optional feature for the wheel loader software product line. More other...
features or feature groups relate to the CDC Function feature are described as following:

- **The CDC Function.** The feature depicts the basic characteristic of CDC. As we described before, the basic CDC function includes the use cases: CDC Activate, CDC Error Handling, CDC Supervision Request, CDC Supervision and CDC Disable. The feature is the group of all those use cases. Meanwhile, the CDC Function feature is sub-divided into other features provided by the CDC function.

- **The CDC Steering.** The feature group depicts the machine steering controlled by CDC. In order to steer the machine by using CDC, there can be two choices: Lever Control Left(Right), which is determined by the left(right) control lever; Joystick Control in Four Directions which is determined by the joystick. The two features are mutually exclusive features. The Lever Control Left(Right) feature is decided from the use case Lever Control Left(Right) and the variation point of use case CDC Steering. The Joystick Control in Four Directions feature is decided from the use case Joystick Control in Four Directions, the variation points of use cases CDC Steering and CDC Moving Forwarding(Reversing).

- **The CDC Moving Forwarding(Reversing).** The feature group depicts the machine moving forward(reversing). If joystick is selected, the machine moving forward(reversing) is controlled by joystick, which relate to the feature Joystick Control in Four Directions. The feature is determined from the use case Joystick Control in Four Directions, the variation points of use cases CDC Steering and CDC Moving Forwarding(Reversing). If the left(right) control lever is selected instead of the joystick, then the speed button should be selected as well. Consequently, there are two features: Gear Selections Button and Low(High) Speed Button, both of them have prerequisite feature Lever Control Left(Right). For all the features within the feature group, they are mutually exclusive features.

- **The End Damping.** The feature group depicts the machine end position retardation. The end damping differs for different size of machines, which can be categorized as Small Machine End Damping or Big Machine End Damping. Those features are mutually exclusive features and they can be determined from the variation point of use case End Damping.

- **The CDC Maintenance.** The feature group depicts CDC function maintenance operation. In the PLUS model proposed in [22], feature group contains variant features. However, in the CDC function, when the CDC function is selected, all the features belong to the CDC Maintenance feature group are essential for the product member. In order to describe the relationship, we introduce the stereotype zero-or-all-of feature group. For all the features: Set Lever Current Range, Set Angle Range and CDC and CDC Supervision Installed are parameterized features. The value of the feature Set Lever Current Range depicts the current range, which is applied in the end damping operation. The value of
the feature **Set Angle Range** depicts the end damping angle range, which depend on the size of the machine. The value of the feature **CDC and CDC Supervision Installed** depicts the CDC install status, which is a Boolean value. The **Set Lever Current Range** is determined from the variation points of use cases **CDC Maintenance**, **CDC Input Request to Service Tool** and **CDC Calibration**. The **Set Angle Range** is determined from the variation points of use cases **CDC Maintenance**, **CDC Steering Angle to Service Tool** and **CDC Calibration**. The Read Out Machine Status via Service Tool feature is optional feature which is derived from the use case Read Out Machine Status via Service Tool.

The PLUS feature model depicts the commonalities and variabilities of the product line, which determines the potential of product line reuse. In the Volvo wheel loader, the features provided by each product are defined by the datasets. For a product, by setting different parameters, different combination of features are implemented on the machine, which may lead to different safety problems. For all the features implemented on a machine, some of them are safety-critical features while others are not safety-critical features. Safety analysis identifies the safety-critical features and their dependencies with other features that are under consideration.

Feature and use case have many-to-many relationship: if one feature is the group of use cases, the feature encapsulates the use cases that can be reused together; if one use case is the group of features, those features are mapped to the variation points of the use case. The wheel loader steering system feature/use case dependencies table are shown in Table 1. The table presents the feature name, the feature category indicating the roles of the features in the product line; the use cases relate to the features (as discussed in the feature model), and the use case category indicating the roles of the use cases in the product line. If more than one feature is mapped to one use case, the name of variation point is presented in the table as well.
<table>
<thead>
<tr>
<th>Feature Name</th>
<th>Feature Category</th>
<th>Use Case Name</th>
<th>Use Case Category/Variation Point</th>
<th>Variation Point Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Steering by Steering Wheel</td>
<td>common</td>
<td>Steering by Steering Wheel</td>
<td>kernel</td>
<td></td>
</tr>
<tr>
<td>Machine Movement</td>
<td>common</td>
<td>Gear Lever Operation Moving by CDC</td>
<td>kernel</td>
<td></td>
</tr>
<tr>
<td>CDC Function</td>
<td>optional</td>
<td>CDC Activate</td>
<td>optional</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>CDC Error Handling</td>
<td>optional</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>CDC Supervision Handling</td>
<td>optional</td>
<td>CDC Disabling</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CDC Supervision Request</td>
<td>optional</td>
<td>CDC Disabling</td>
</tr>
<tr>
<td>Small Machine End Damping</td>
<td>alternative</td>
<td>End Damping</td>
<td>vp</td>
<td>End Damping</td>
</tr>
<tr>
<td>Big Machine End Damping</td>
<td>alternative</td>
<td>End Damping</td>
<td>vp</td>
<td>End Damping</td>
</tr>
<tr>
<td>Set Lever Current Range</td>
<td>parameterized</td>
<td>CDC Maintenance</td>
<td>vp</td>
<td>CDC Maintenance</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CDC Input Request to Service Tool</td>
<td>vp</td>
<td>Lever Current Range</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CDC Calibration</td>
<td>vp</td>
<td>Set Lever Current</td>
</tr>
<tr>
<td>Set Angle Range</td>
<td>parameterized</td>
<td>CDC Maintenance</td>
<td>vp</td>
<td>CDC Maintenance</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CDC Steering Angle to Service Tool</td>
<td>vp</td>
<td>Angle Range</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CDC Calibration</td>
<td>vp</td>
<td>Set Angle Range</td>
</tr>
<tr>
<td>Feature Name</td>
<td>Feature Category</td>
<td>Use Case Name</td>
<td>Use Case Category/Variation Point</td>
<td>Variation Point Name</td>
</tr>
<tr>
<td>------------------------------</td>
<td>------------------</td>
<td>----------------------------------------</td>
<td>----------------------------------</td>
<td>---------------------------------------</td>
</tr>
<tr>
<td>CDC and CDC Supervision</td>
<td>parameterized</td>
<td>CDC Maintenance</td>
<td>vp</td>
<td>CDC Maintenance Install Status</td>
</tr>
<tr>
<td>Installed</td>
<td></td>
<td>CDC and CDC Supervision Installed</td>
<td>vp</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Check CDC Install Status</td>
<td>optional</td>
<td></td>
</tr>
<tr>
<td>Read out Machine Status via</td>
<td>optional</td>
<td>Read Out Machine</td>
<td>optional</td>
<td></td>
</tr>
<tr>
<td>Service Tool</td>
<td></td>
<td>Status via Service Tool</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Lever Control Left(Right)</td>
<td>alternative</td>
<td>Lever Control</td>
<td>alternative</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Left(Right)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>CDC Steering</td>
<td>vp</td>
<td></td>
</tr>
<tr>
<td>Joystick Control in Four</td>
<td>alternative</td>
<td>Joystick Control in Four Directions</td>
<td>alternative</td>
<td></td>
</tr>
<tr>
<td>Directions</td>
<td></td>
<td>CDC Steering</td>
<td>vp</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>CDC Moving</td>
<td>vp</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Forwarding(Reversing)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Gear Selection Button</td>
<td>alternative</td>
<td>Gear Selection Button</td>
<td>alternative</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>CDC Moving</td>
<td>vp</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Forwarding(Reversing)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Low(High) Speed Button</td>
<td>alternative</td>
<td>Low(High) Speed Button</td>
<td>alternative</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>CDC Moving</td>
<td>vp</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Forwarding(Reversing)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 1: CDC feature/use case dependencies of the wheel loader software product line.
4.1.2.2 Software Product Line Analysis Modeling

Static Modeling

The static model depicts the static structure of the product line system. A product line context diagram is derived from the static model. Other than modeling the static structure, the context model defines the boundary and external environment of the product line system. In the CDC case, in order to build the context model, first we have to develop a static model with all the operations specified.

In the static model, the product line is composed of various of physical I/O devices (sensors and actuators). In UML, those devices are depicted in the class form. Some of the classes are organized in an hierarchical form: the superclass is common to all the products, while the subclasses belonging to the superclass are variable for the products. The conceptual static model is shown in Figure 17.

In the wheel loader product line, the devices include: Gear Lever, Steering Wheel and CDC System. The CDC System is a superclass which contains subclasses and sub-superclass. The subclasses belongs to CDC System are: Hydraulic System, Armrest, Activate Button, Gearbox Monitor, Angle Sensor, Display Unit, Engine Monitor, Service Tool, PWM Valve. The sub-superclass belongs to CDC System are: Lever, which contains subclasses Left(Right) Control Lever and Joystick, both classes are mutually exclusive classes; Gearbox, which contains the subclasses Big Machine Gear Box and Small Machine Gearbox, both classes are mutually exclusive classes; Speed Button, which contains the subclasses Low(High) Speed Button and Change Gear Button, those two classes are mutually exclusive classes; Engine, which contains the subclasses Wit Joystick Interface and Without Joystick Interface, both classes are mutually exclusive classes as well.
As to the product line context model, we group the physical devices as context of the software product line. Each class has two stereotypes: regarding the reuse category and regarding the role those classes play in the system. The context model is shown in Figure 18.

In the machine movement system, Gear Lever, Steering Wheel are external input devices, they are implemented on every product member, thus they are kernel. Activation Button, Speed Button, Angle Sensor, Lever, Armrest are external input devices, they
Figure 18: Wheel loader software product line context model.
are implemented to realize the CDC function, which is optional, thus those are optional classes. Engine Monitor, PWM Valve, Gearbox Monitor, Service Tool, Gearbox and Engine are external input/output devices, which provides both input and output to the product line system, each product member has all such devices, thus they are kernel classes. Display Unit is external output device and kernel. Hydraulic System is external system and kernel.

The PLUS static model describes the physical devices involved in the product line system. In industry, those physical devices can be regarded as the hardware of the product line system. Since the PLUS modeling mainly considers the software of a product line, we focus on the product line software aspect as well. The product line hardware interfaces with the product line software, thus the product line hardware can be regarded as the environment where the product line software performs. The PLUS static model identifies the hardware environment as the context of the software product line system. The context model is important for safety analysis, because different failing of functions in different environments lead to different hazards. On the other hand, in the static model, the probability of the hardware failure can be identified.

**Dynamic Modeling**

**Dynamic Interaction Modeling**

Static model depicts the structuring criteria of the objects, dynamic diagram addresses the interaction among those objects. The relationship can be described by either communication diagram or message sequence diagram. Gomaa [22] proposes the communication diagram, we try it out in this report.

The communication diagram describes the way objects communicate to satisfy the requirement described by each of the use case. In this report we are presenting an excerpt of the complete set of communication diagrams.

To draw the communication diagram, we have to define the objects involved in it by the object structuring criteria. In order to find out the objects, we have to determine the classes first, where we follow both the kernel first approach and evolution approach. In UML, the communication diagram is depicted by the object communication diagram. For the CDC case, the communication diagram is shown from Figure 19 to Figure 24.

For the wheel loader software product line system specified in the CDC function, there are types of classes which are based on the static model, we determine those classes by using the class structuring criteria:

- **Input device interface classes**: Joystick Interface, Armrest Interface, Activate Button Interface, Left(Right) Control Lever Interface, Gear Selection Button Interface, Low(High) Speed Button Interface,
Angle Sensor Interface. The input device interface object interfaces to an input hardware I/O device [22].

- **Output device Interface class**: Display Unit Interface. The output device interface object interfaces to an output hardware I/O device [22].

- **Input/Output device Interface classes**: PWM Valve Interface, Gearbox Monitor Interface, Engine Monitor Interface. The input/output device interface object interfaces to an input/output device [22].

- **Control classes**: CDC Supervision Control and CDC Function Control. The control class encapsulates a state machine.

- **Entity class**: CDC Data. The entity class stores information, i.e. data.

Objects are derived from those classes as their instances, the communication diagrams are built based on the objects and the way they communicate to address the functionality of each use case:

- **The CDC Activate use case**. The CDC Activate communication diagram is shown in Figure 19. The content of the messages are:
  
  B1: CDC Installed

  Armrest Down:
  B2: CDC Activate
  B2.1: Armrest Down

  Activate:
  B3: Activate Button Input
  B3.1: Pushed CDC Activate Button
  B3.2: Activate Request
  B3.3: Check CDC Status
  B3.4: CDC Status
  B3.5: CDC Ready

  The CDC activate process is based on the premise that the CDC function is installed, as shown in message B1:CDC Installed, which is sent out by the object CDC Data to the object CDC Function Control. This process includes two steps: first the user puts down the armrest, then the user presses the activate button. When the user puts down the armrest, the Armrest object sends message B2:CDC Activate to the object Armrest Interface, which sends the message B2.1:Armrest Down to the object CDC Function Control. In the next step, the user presses the activate button, the Activate Button sends message to the Activate Button Interface, which sends the B3.1:Pushed CDC Activate Button message to the CDC Function Control object. After getting the activate request, the CDC Function Control object sends the message B3.2:Activate Request to the object Gearbox Monitor Interface, which passes the B3.3:Check
Figure 19: Communication diagram for the use case: CDC Activate.
CDC Status request to the Gearbox Monitor. In the activation process, the Gearbox Monitor is mainly used for checking the gearbox status, i.e., whether the gearbox is ready for activating CDC function. In the last step, the Gearbox Monitor sends the gearbox status back to the Gearbox Monitor Interface, which sends the status to the CDC Function Control object.

- The **Joystick Control in Four Directions** use case. The user has the possibility to choose a joystick steering for moving the machine. If the user controls the machine in CDC mode with joystick. The joystick can either steer the machine or drive the machine forwarding(reversing), we discuss those two situations separately.

The Joystick Control in Four Directions communication diagram is shown in Figure 20. The content of the messages are:

**Steering:**
- A1: Joystick Left(Right) Input
- A1.1: Joystick Move to the Left(Right)
- A1.1a: Joystick Move to the Left(Right)
- A1.2: Steering Signal
- A1.2a: Steering Signal
- A1.3: Steering Signal Output
- A1.4: Steering Return Status Input
- A1.5: Steering Return Status

**Forwarding(Reversing):**
- A2: Joystick Forward(Reverse) Input
- A2.1: Joystick Forward(Reverse)
- A2.1a: Joystick Forward(Reverse)
- A2.2: Forwarding(Reversing) Signal
- A2.2a: Forwarding(Reversing) Signal
- A2.3: Forwarding(Reversing) Output
- A2.3a: Forwarding(Reversing) Output
- A2.4: Forwarding(Reversing) Return Status Input
- A2.4a: Forwarding(Reversing) Return Status Input
- A2.5: Return Status
- A2.5a: Return Status

If the user uses joystick to steer the machine, the Joystick send the steering request to the Joystick Interface, whereas the Joystick Interface sends the message A1.1:Joystick Move to the Left(Right) to the object CDC Function Control. As we discussed before, the whole CDC operating should be under supervision, which is realized by the CDC Supervision Control object. Therefore, the Joystick Interface object sends concurrent message A1.1a:Joystick Move to the Left(Right) to the object CDC Supervision Control. If the user uses the joystick to drive the machine forwarding or reversing. The Joystick
Figure 20: Communication diagram for the use case: Joystick Control in Four Directions.
object passes the information to the objects CDC Function Control and CDC Supervision Control in the same path as steering. However, the CDC Function Control object generates different messages. The CDC Function Control generates the message A2.2:Forwarding(Reversing) Signal and A2.2a:Forwarding(Reversing) Signal, passes them to the object Gearbox Monitor Interface and Engine Monitor Interface separately. Those two objects passes the output control signal to the objects Gearbox Monitor and Engine Monitor. Those monitors will response the A2.5, A2.5a:Return Status Signal to the Gearbox Monitor Interface and Engine Monitor Interface, which passes the return signal to the CDC Function Control object.

- The **Gear Selection Button Control** use case. A gear selection button is an alternative way to control the machine speed other than joystick. The gear selection button can be used only when the machine is implemented with the left(right) control lever. The Gear Selection Button Control communication diagram is shown in Figure 21. The content of the messages are:

  **Steering:**
  C1: Lever Left(Right) Input  
  C1.1: Lever Move to the Left(Right)  
  C1.1a: Lever Move to the Left(Right)  
  C1.2: Steering Signal  
  C1.2a: Steering Signal  
  C1.3: Steering Signal Output  
  C1.4: Steering Return Status Input  
  C1.5: Steering Return Status

  **Forwarding(Reversing):**
  C2: Gear Selection Button Input  
  C2.1: Increase (Decrease) Gear  
  C2.1a: Increase (Decrease) Gear  
  C2.2: Increase (Decrease) Gear signal  
  C2.2a: Increase (Decrease) Gear signal  
  C2.3: Increase (Decrease) Gear Output  
  C2.4: Increase (Decrease) Gear Return Status Input  
  C2.5: Return Status

When the user uses the left(right) control lever to steer the machine, the Left(Right) Control Lever object sends the steering message (message C1:Lever Left(Right) Input) to the object Left(Right) Control Lever Interface, which sends the lever message (message C1.1:Lever Move to the Left(Right)) and message C1.1a:Lever Move to the Left(Right) to the objects CDC Function Control and CDC Supervision Control. The CDC Function Control object generates the message C1.2, C1.2a:Steering Signal and sends them to the object PWM Valve Interface and CDC Supervision Control object separately. The PWM
Figure 21: Communication diagram for the use case: Gear Selection Button Control.
Valve Interface object sends the C1.3:Steering Signal Output to the object PWM Valve. The same principle as we described in the use case Joystick Control in Four Directions, the PWM Valve sends the return status signal to the object CDC Function Control via the PWM Valve Interface.

The user can also adjust the gear of the machine by using the gear selection button. The user pushes the button, the Gear Selection Button object sends the message C2:Gear Selection Button Input to the Gear Selection Button Interface object. The Gear Selection Button Interface object then sends the message C2.1,C2.1a:Increase(Decrease) Gear to the objects CDC Supervision Control and CDC Function Control. The CDC Function Control object generates the C2.2, C2.2a:Increase(Decrease) Gear Signal and sends them to the Gearbox Monitor Interface object and CDC Supervision Control object separately. The Gearbox Monitor Interface sends the message C2.3:Increase(Decrease) Gear Output to the object Gearbox Monitor object. The monitor detects the status of the gearbox and sends the return status to CDC Function Control object via the Gearbox Monitor Interface.

- The Low(High) Speed Button Control use case. As a simplified version of the gear selection button, the low(high) speed button decreases the normal gear selection choices to two gears: low speed gear and high speed gear. The low(high) speed button is used based on the premise that the left(right) control lever is implemented. The Low(High) Speed Button Control use case has the similar information flow as the Gear Selection Button Control use case, for simplicity, we do not describe the information path in detail.

The Low(High) Speed Button Control communication diagram is shown in Figure 22. The content of the messages are:

Steering:
C1: Lever Left(Right) Input
C1.1: Lever Move to the Left(Right)
C1.1a: Lever Move to the Left(Right)
C1.2: Steering Signal
C1.2a: Steering Signal
C1.3: Steering Signal Output
C1.4: Steering Return Status Input
C1.5: Steering Return Status

Forwarding(Reversing):
D1: Low(High) Speed Button Input
D1.1: Increase (Decrease) Gear
D1.1a: Increase (Decrease) Gear
D1.2: Increase (Decrease) Gear signal
D1.2a: Increase (Decrease) Gear signal
D1.3: Increase (Decrease) Gear Output
D1.4: Increase (Decrease) Gear Return Status Input  
D1.5: Return Status

Figure 22: Communication diagram for the use case: Low(High) Speed Button Control.

- The **End Damping** use case. In CDC steering mode, the machine should reduce its speed when reaching the left(right) maximum angle, which is called as end damping. The end damping is crucial for the machine to operate smoothly. Both joystick control and left(right) lever control should have end damping. The joystick control end damping and left(right) lever control end damping have similar principles, thus we discuss the joystick control end damping in detail.

The joystick control end damping communication diagram is shown in Figure 23. The content of the messages are:

**End Damping:**

F1: Joystick Left(Right) Input  
F1.1: Joystick Move to the Left(Right)
E2: Current Angle Input
E2.1: Current Angle
E2.1a: End Damping Range, Maximum Angle Value
E2.2: Reduced Steering Signal
E2.3: Reduced Steering Output
E2.4: Steering Return Status Input
E2.5: Steering Return Status

Figure 23: Communication diagram for the use case: End Damping of joystick control.

As shown in Figure 23. When the user uses the joystick to steer the machine, the Joystick object sends the F1: Joystick Left(Right) Input message to the object Joystick Interface. Whereas the Joystick Interface sends the F1.1: Joystick Move to the Left(Right) message to the object CDC Function Control. On the other hand, the Angle Sensor detects the current angle and send it to Angle Sensor Interface, which passes the angle information to the object CDC Function Control. Meanwhile, the CDC Function Control receives the
The CDC Function Control object compares the current angle with the angle information in message E2.1a:End Damping Range, Maximum Angle Value, once the machine reaches the end position, the CDC Function Control object sends the E2.2:Reduced Steering Signal to the object PWM Valve Interface, which pass the message to the object PWM Valve. The PWM Valve responds the steering status to the CDC Function Control object via the PWM Valve Interface object.

The left (right) lever control end damping communication diagram has similar information flow as the joystick end damping communication diagram, however, the Joystick object and Joystick Interface object are substituted with the Lever object and (Left)Right Control Lever Interface. For simplicity reason, we do not present the left(right) lever control end damping communication diagram.

- The **CDC Error Handling** use case. If the output control signal does not match with the input request signal, or the input request signal cannot be detected, indicating error happens while CDC operation. The communication diagram for the CDC Error Handling use case is shown in the Figure 24. The content of the messages are:

  Steering Error Handling:
  G1: Error Happen while Steering
  G1a: Error Happen while Steering
  G1.1: Steering Disable
  G1a.1: Error Information

  Forwarding(Reversing) Error Handling:
  G2: Error Happen while Forwarding(Reversing)
  G2a: Error Happen while Forwarding(Reversing)
  G2b: Error Happen while Forwarding(Reversing)
  G2.1: Forwarding(Reversing) Disable
  G2a.1: Forwarding(Reversing) Disable
  G2b.1: Error Information

When the error happens during CDC steering, the object CDC Supervision Control sends the message G1,G1a:Error Happen while Steering to the objects PWM Valve Interface and Display Unit Interface. Those two interfaces send the message G1.1:Steering Disable and G1a.1:Error Information to the objects PWM Valve and Display Unit separately. As the result, the CDC function is disabled and the error information is presented to the user.

When the error happens during CDC forwarding (reversing), the object CDC Supervision Control sends the messages G2,G2a,G2b:Error Happen while Forwarding(Reversing) to the objects Engine Monitor Interface, Gearbox Monitor Interface and Display Unit Interface. Those Interfaces send the
Figure 24: Communication diagram for the use case: CDC Error Handling.
message to the corresponding external devices. The similar result as the CDC steering error handling, the CDC function is disabled and the user is informed.

In the case that error happens during CDC steering operation and CDC forwarding (reversing) operation, the message path is the combination of the former two cases, for simplicity reason, we do not discuss the case in detail.

**Finite State Machine**

A statechart is useful for describing the historical state of the system and its input. The system control and sequencing view is depicted by the statechart. In UML, the statechart is depicted by the state machine diagram.

In the CDC case, the objects *CDC Function Control* and *CDC Supervision Control* are state dependent control objects, indicating that the objects contain statecharts. Normally, one state dependent object contains one statechart. Since the CDC function control and the CDC supervision control perform together, we choose to use one statechart depicting the two objects.

The statechart is shown in Figure 25. The content of the events are:

**Activate:**
- B2.1: Armrest Down
- B3.1: Pushed CDC Activate Button
- B3.2: Activate Request
- B3.5: CDC Ready

**Steering(Lever):**
- C1.1: Lever Move to the Left(Right)
- C1.1a: Lever Move to the Left(Right)
- C1.2: Steering Signal
- C1.2a: Steering Signal
- C1.5: Steering Return Status

**Forwarding(Reversing)(Lever and Gear Selection Button):**
- C2.1: Increase (Decrease) Gear
- C2.1a: Increase (Decrease) Gear
- C2.2: Increase (Decrease) Gear signal
- C2.2a: Increase (Decrease) Gear signal
- C2.5: Return Status

**Forwarding(Reversing)(Lever and Low(High) Speed Button):**
- D1.1: Increase (Decrease) Gear
- D1.1a: Increase (Decrease) Gear
D1.2: Increase (Decrease) Gear signal  
D1.2a: Increase (Decrease) Gear signal  
D1.5: Return Status

Steering (Joystick):  
A1.1: Joystick Move to the Left (Right)  
A1.1a: Joystick Move to the Left (Right)  
A1.2: Steering Signal  
A1.2a: Steering Signal  
A1.5: Steering Return Status

Forwarding (Reversing):  
A2.1: Joystick Forward (Reverse)  
A2.1a: Joystick Forward (Reverse)  
A2.2: Forwarding (Reversing) Signal  
A2.2a: Forwarding (Reversing) Signal  
A2.5: Return Status  
A2.5a: Return Status

End Damping:  
E1.1: Lever Move to the Left (Right)  
F1.1: Joystick Move to the Left (Right)  
E2.1: Current Angle  
E2.1a: End Damping Range, Maximum Angle Value  
E2.2: Reduced Steering Signal  
E2.5: Steering Return Status

Error Handling:  
G1: Error Happen while Steering  
G1a: Error Happen while Steering  
G2: Error Happen while Forwarding (Reversing)  
G2a: Error Happen while Forwarding (Reversing)  
G2b: Error Happen while Forwarding (Reversing)

When the CDC and CDC supervision are installed, the statechart is in the CDC Available and CDC Monitor Available state. When the user puts down the armrest and pushes the activate button, the object Armrest Interface and Activate Button Interface send messages Armrest Down (message B2.1 in Figure 19) and Pushed CDC Activate Button (message B3.1 in Figure 19) to the object CDC Function Control. The arrival of the message triggers the events Armrest Down and Pushed CDC Activate Button. If the lever control left (right) feature is selected, then the state transits from CDC Available to {feature=Lever Control Left (Right)} Lever; if the joystick control in four directions feature is selected, the CDC Available state transits to
Figure 25: Statechart for CDC Supervision Control and CDC Function Control objects.
state \{feature=Joystick Control in Four Directions\} Joystick. Concurrently, the CDC Monitor Available state transits to the CDC Monitor Activated state.

After activating the CDC function, the user controls the machine by using either the left (right) control lever or the joystick. Since the joystick control has a similar state transition path as the left (right) lever control, but more states are involved in the joystick control, we only present the joystick control state transition in detail.

If the user moves the joystick to the left (right), the object Joystick Interface sends the message Joystick Move to the Left(Right) (message A1.1, A1.1a in Figure 20) to the object CDC Function Control and CDC Supervision Control. The arrival of the message triggers the event Joystick Move to the Left(Right) (event A1.1 in Figure 25) which cause the state transition from \{feature=Joystick Control in Four Directions\} Joystick to state Steering. The state transition resulting in the transition action Steering Signal (action A1.2, A1.2a in Figure 25). The transition action causes the object CDC Function Control sending the message Steering Signal (message A1.2, A1.2a in Figure 20) to the object PWM Valve Interface.

If the user moves the joystick to the front (end), the state transition path is similar as the joystick steering. But the input/output device interface involved in the communication diagrams are Engine Monitor Interface and Gearbox Monitor Interface instead of PWM Valve Interface.

Besides the Steering state and Forwarding(Reversing) state, there can be the other state Forwarding(Reversing) Steering state. In this state, the machine moves forwarding (reversing) while steering.

If the machine reaches the maximum angle while steering, it performs the end damping. When the user uses the joystick to control the machine, the Joystick Interface sends message Joystick Move to the Left(Right) (message F1.1 in Figure 23) to the object CDC Function Control. The object Angle Sensor Interface sends message Current Angle (message E2.1 in Figure 23) to the object CDC Function Control, and the object CDC Data sends the message End Damping Range, Maximum Angle Value (message E2.1a in Figure 23) to the object CDC Function Control. The message triggers the corresponding event in the statechart, which cause the self-transition of the state Steering. The self-transition results the object CDC Function Control sends the message Reduced Steering Signal (message E2.2 in Figure 23) to the object PWM Valve Interface.

If an error happens during joystick steering operating, the event CDC Error is triggered, which causes the state transition from Steering to CDC Disabled, meanwhile, the state CDC Monitor Activated transits to CDC Disabled. The later state transition results in the transition action Error Happen while Steering (action G1, G1a in the Figure 25). The transition action cause the object CDC Supervision Control sends the message Error Happen while Steering (message G1, G1a in the Figure 24) to the corresponding external devices.
If an error happens during joystick forwarding (reversing) operating, the event **CDC Error** is triggered, which cause the state transition from **Forwarding(Reversing)** to the state **CDC Disabled**, and the state transition from **CDC Monitor Activated** state to the state **CDC Disabled**. The later state transition causes the transition action **G2, G2a, G2b: Error Happen while Forwarding(Reversing)**, which in turn cause the object **CDC Supervision Control** sends messages **Error Happen while Forwarding(Reversing)** (message G2, G2a and G2b in the Figure 24) to the objects **Engine Monitor Interface**, **Gearbox Monitor Interface** and **Display Unit Interface**.

If the error happens during joystick steering and forwarding (reversing) operation, the statechart transits from **Forwarding(Reversing) Steering** state to the **CDC Disabled** state, at the same time, the statechart transits from the **CDC Monitor Activated** state to the **CDC Disabled** state. For simplicity reason, we do not describe the state transition process in detail.

After the CDC function is disabled because of error. The user can try to restart the CDC by putting the armrest up and done, which trigger the event **Restart CDC**. The event causes the statechart transits from **CDC Disabled** state to the **CDC Monitor Available** and **CDC Available** states. After the restart operation, if the machine can operate properly, the statechart returns to the initial states.

The PLUS dynamic interaction model depicts the communication among objects to realize the requirement of each use case. The information path, i.e., scenario, is analyzed in specific context where the scenario performs. Different risk levels are determined from the different scenarios. To be specific, the scenario presents a specific message path. The message is passed among the objects, which are derived from the different class configuration decisions. The different class configuration may lead to different risk levels. The statechart depicts the behavior of a specific system, and provides dynamic information for the functional safety analysis. According to [38], the statechart can be tested against the scenario depicted in the dynamic interaction model. The test helps defining the functional safety requirement and improving the design.

### 4.1.2.3 Software Product Line Feature/Class Dependency Modeling

The feature model describes the characteristics of the software product line. Static model determines the kernel, optional and variant classes of the software product line. The relationship between features and classes is presented by the feature/class dependency modeling.

The wheel loader software product line CDC function feature/class dependencies are summarized in Table 2.

In the CDC case, we start modeling the feature/class dependency by analyzing the classes involved to realize each of the feature. Since we mainly consider the CDC function in our
modeling, we only concern the features related to the CDC function in the feature/class dependency analysis. For each of the feature related to the CDC function:

- **The CDC Function** feature. The feature is an optional feature. Some of the superclasses related to the feature contain variant classes. The superclasses are: Lever Interface, which contains the variant classes Joystick Interface and Left(Right) Control Lever Interface, the two classes are mutually exclusive. The Speed Button Interface superclass, which contains the variant classes Gear Selection Button Interface and Low(High) Speed Button Interface, those two classes are mutually exclusive as well; the CDC Supervision Control, CDC Function Control and CDC Data superclasses, which are parameterized classes, the parameter those classes contain differ from each of the product.

- **The CDC and CDC Supervision Installed** feature. The feature is optional for the software product line, since the CDC function is optional. The class relates to the feature is CDC Data, which contains the Install Status: Boolean parameter.

- **The Lever Control Left(Right)** feature. In the case that the CDC function is installed, the user can choose either the left (right) control lever or the joystick. The class that is related to the feature is Left(Right) Control Lever Interface. Since the left (right) control lever and the joystick are alternative for each of the product implemented with the CDC function, the class related to this feature is variant.

- **The Joystick Control in Four Directions** feature. The feature is alternative to the software product line. The class relates to this feature is Joystick Interface. As we stated before, the feature related to the left (right) control lever and the joystick are mutually exclusive, thus the class related to this feature is variant.

  Because the Lever Control Left(Right) feature and the Joystick Control in Four Directions feature belong to the feature group CDC Steering, which has the stereotype zero-or-one-of feature group (as shown in the feature model, see Figure 16), those two feature cannot coexist, the classes relate to the features cannot coexist either.

- **The Big Machine End Damping** feature. Different product has the different parameterized end damping range, the classes related to the feature is parameterized variant class. The classes are CDC Function Control and CDC Data, the parameters related to the classes are End Damping Interval: float.

- **The Small Machine End Damping** feature. The same as the Big Machine End Damping feature, the variant classes related to the feature are CDC Function Control and CDC Data, their parameter is End Damping Interval: float.

  The Big Machine End Damping and Small Machine End Damping feature belong to the feature group End Damping, which is a mutually exclusive feature
Methods for Modeling of Product Lines for Safety-critical Systems

MSC Group (as shown in the feature model, see Figure 16). The variant classes support those features cannot coexist.

- The **Gear Selection Button** feature. When the left (right) control lever is selected, the speed button must be implemented. There are two kinds of speed buttons: the gear selection button and the low (high) speed button. The Gear Selection Button feature describes the characteristic of the gear selection button. The class relates to this feature is Gear Selection Button Interface.

- The **Low(High) Speed Button** feature. As an alternative feature to the feature Gear Selection Button, the variant class relates to the feature is Low(High) Speed Button Interface.

The Joystick Control in Four Directions feature, Gear Selection Button feature and the Low(High) Speed Button feature belong to the feature group CDC Forwarding(Reversing), which is a mutually exclusive feature group (as shown in Figure 16). The variant classes relate to those features cannot coexist.

- The **Set Lever Current Range** feature and **Set Angle Range** feature. Those two features belong to the feature group CDC Maintenance (as shown in the Figure 16). The features are essential to the feature group, but the parameters of the feature vary for each product, thus the classes relate to the features have stereotype optional-param-vp. The class is CDC Data, the class parameter is Angle Range: float.

- The **Read Out Machine Status via Service Tool** feature. The feature is common for the software product line, since using the service tool for maintenance is a crucial step for each of the product. The class relate to the feature is Service Tool Interface, which is kernel class. On the other hand, the Read Out Machine Status via Service Tool is essential for the CDC Maintenance feature group.

The different PLUS models support different aspects of the functional safety analysis. The main functional safety analysis technique we consider in our work is the FTA. In the later discussion, we will present the relationship between the PLUS model and the fault tree in detail.
<table>
<thead>
<tr>
<th>Feature Name</th>
<th>Feature Category</th>
<th>Class Name</th>
<th>Class Reuse Category</th>
<th>Class Parameter</th>
</tr>
</thead>
<tbody>
<tr>
<td>CDC Function</td>
<td>optional</td>
<td>Lever Interface</td>
<td>optional-abstract-vp</td>
<td>kernel</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Display Interface</td>
<td></td>
<td>kernel</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Angle Sensor Interface</td>
<td></td>
<td>optional</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Activate Button Interface</td>
<td></td>
<td>optional</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Armrest Interface</td>
<td></td>
<td>optional</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CDC Supervision Control</td>
<td></td>
<td>optional-param-vp</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CDC Function Control</td>
<td></td>
<td>kernel</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Gearbox Monitor Interface</td>
<td></td>
<td>optional-param-vp</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Engine Monitor Interface</td>
<td></td>
<td>kernel</td>
</tr>
<tr>
<td></td>
<td></td>
<td>PWM Valve Interface</td>
<td></td>
<td>optional-param-vp</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CDC Data</td>
<td></td>
<td>kernel</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Gearbox Interface</td>
<td></td>
<td>optional-abstract-vp</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Engine Interface</td>
<td></td>
<td>kernel</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Hydraulic System</td>
<td></td>
<td>optional-param-vp</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Speed Button Interface</td>
<td></td>
<td>optional-abstract-vp</td>
</tr>
<tr>
<td>CDC and CDC Supervision</td>
<td>parameterized</td>
<td>CDC Data</td>
<td>optional-param-vp</td>
<td>Install Status: boolean</td>
</tr>
<tr>
<td>Installed</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Lever Control</td>
<td>alternative</td>
<td>Left(Right) Control</td>
<td>variant</td>
<td></td>
</tr>
<tr>
<td>Left(Right)</td>
<td></td>
<td>Lever Interface</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Joystick Control in Four</td>
<td>alternative</td>
<td>Joystick Interface</td>
<td>variant</td>
<td></td>
</tr>
<tr>
<td>Directions</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Big Machine End Damping</td>
<td>alternative</td>
<td>CDC Function Control</td>
<td>optional-param-vp</td>
<td>End Damping Interval: float</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CDC Data</td>
<td>optional-param-vp</td>
<td></td>
</tr>
<tr>
<td>Small Machine End Damping</td>
<td>alternative</td>
<td>CDC Function Control</td>
<td>optional-param-vp</td>
<td>End Damping Interval: float</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CDC Data</td>
<td>optional-param-vp</td>
<td></td>
</tr>
</tbody>
</table>
4.1.3 PLUS-based Embedded Real-Time Software Product Line Design

In ISO 26262, in order to enable the controlled changes of the software behavior for different applications, the configurable software is provided that uses the configuration and calibration data. In order to detect, prevent or mitigate the fault that can cause interference between software elements, the effects of faults from the achievement of freedom from interface, timing and execution, memory and exchange of information should be considered.

As a model-based development method, the PLUS process can be used for designing embedded real-time software product lines [45]. The PLUS model addresses different views of the software product line, i.e., requirements models, static models, dynamic models. After the PLUS analysis phase, the design phase is defined for designing the software architecture. In the design modeling, a system is structured into subsystems.

The structure of the subsystems is decided by the software architecture structure patterns. The software architecture pattern concerns the architecture static structure, it is a set of design decisions for the connectors, software components and behavior. The

<table>
<thead>
<tr>
<th>Feature Name</th>
<th>Feature Category</th>
<th>Class Name</th>
<th>Class Reuse Category</th>
<th>Class Parameter</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gear Selection Button</td>
<td>alternative</td>
<td>Gear Selection Button Interface</td>
<td>variant</td>
<td></td>
</tr>
<tr>
<td>Low(High) Speed Button</td>
<td>alternative</td>
<td>Low(High) Speed Button Interface</td>
<td>variant</td>
<td></td>
</tr>
<tr>
<td>Set Lever Current Range</td>
<td>parameterized</td>
<td>CDC Data</td>
<td>optional-param-vp</td>
<td>Lever Current Range: float</td>
</tr>
<tr>
<td>Set Angle Range</td>
<td>parameterized</td>
<td>CDC Data</td>
<td>optional-param-vp</td>
<td>Angle Range: float</td>
</tr>
<tr>
<td>Read Out Machine Status</td>
<td>common</td>
<td>Service Tool Interface</td>
<td>kernel</td>
<td></td>
</tr>
</tbody>
</table>

Table 2: CDC feature/class dependencies of the wheel loader software product line.
PLUS process is a component-based approach, in which each subsystem is designed as a distributed software component. The concurrent component is designed as active and passive tasks. The tasks communication and synchronization interfaces are designed as well.

The interactions between components are addressed by the software architecture communication patterns. The communication pattern concerns the nature of the connectors and their organization. For the distributed software components, all the communication between components is restricted to message communication.

According to [45], the software product line design performance can be judged by the approach based on rate monotonic analysis, which is “a collection of quantitative methods, provides a basis for designing, understanding, and analyzing the timing behavior of real-time industrial computing systems” [46]. For hard real-time systems, systems miss deadline could lead to catastrophic results. Such potential performance problem should be detected in the quantitative analysis of real-time system design [45]. The design performance should be judged from the safety perspective as well. Considering the compliance with ISO 26262, the PLUS model can provide the information for the achievement of freedom from interfaces, exchange of information. With further improvement to the PLUS model, it can be used for analyzing the timing and execution. But the PLUS model cannot provide information with respect to memory.

4.1.4 Functional Safety Analysis

As we described in Section 2.3, there are many kinds of functional safety analysis techniques, and we choose the FTA to study further. In this section, we extend the system FTA for analyzing the wheel loader product line functional safety.

Since the fault tree is used for investigating the cause of a hazard, for product lines it is necessary to take all possible causes in different products into account. For a single system we have presented the approach of [2] above in Figure 8. In order to meet the requirements for product lines, we adapt the system fault tree layers to meet the specifics for the product line approach. In detail, the ‘top undesired event’ is also used in the product line approach that is derived from the PHA. The ‘system functions’ layer needs to be adapted to cover all possible functions of all product line members and is denoted as ‘product line function’. The ‘system phases’ layer is adapted to cover the system hardware and software aspects, which is defined as ‘product line system’. The ‘fault flow’ is specified as ‘single system fault flow’ which reveals the dynamic information. Later on, the dynamic information is depicted by the product line model. For the product line fault tree, we do not go down to the system component level, thus the fault tree does not contain the ‘basic failure events’ layer.

The information needed by product line fault tree is derived from the product line models mentioned above. We will show the process of building the fault tree in detail, moreover,
we will try to map the PLUS model and feature diagram with the product line fault tree.

In order to build the fault tree, we have to define the undesired top event first. The undesired top event for the FTA is derived from the hazards identified in the PHA. We use a worksheet to present the PHA, as presented in Table 3, which is showing an simplified excerpt from the PHA for the Steer by Wire hazard analysis.

Table 3: PHA worksheet for detailed and hazard analysis. The table is specified for the hazard analysis of function Steer by Wire.

<table>
<thead>
<tr>
<th>No.</th>
<th>Rev.</th>
<th>Date</th>
<th>Hazard</th>
<th>Situation and frequency (F)</th>
<th>Analysis</th>
<th>ISO 15982</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td></td>
<td>Steer by Wire</td>
<td>Machine Operator inspecting (operator not active)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td></td>
<td>Steer by Wire</td>
<td>Machine Operator inspecting (operator not active)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td></td>
<td>Steer by Wire</td>
<td>Machine Operator inspecting (operator not active)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td></td>
<td>Steer by Wire</td>
<td>Machine Operator inspecting (operator not active)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

We describe the general information contained in the table in top-down and left-right sequence.

- **Rev.** indicates the date the table is created.
- **Updated acc. To ISO/WD 15998-2 risk graph** identifies the risk level and the Safety Integrity Level (SIL) decided by the standard ISO 15998-2.
- **Function/application** identifies the function under analysis.
- **Phase** indicates the condition that the machine operates. M represents machine operator make daily service and walk around inspection(engine not running); I represents engine running at low idle, machine parked; OP represents normal operation, not specified in detail (bucket loaded); PR represents travelling on public road.
- **Hazard** identifies the specific hazard that is evaluated.
- **Situation and frequency (F)** describes the frequency and/or exposure to hazard.
• **Maximum injury (S)** identifies the severity of injury.

• **Possibility to avoid or reduce risk (P)** identifies the possible of avoiding hazard or limiting harm.

The F, S, P are judged by degree: F1-F2, S1-S3, P1-P2. The risk degree increases with the number increases.

• **ISO 15998-2 PLr** integrates F, S, P by using a risk matrix (for commercial sensitivity, the matrix is not presented in this report).

After deciding the undesired top event **Unintended Movement when Working at the Work site** from Table 3 hazard 5, we build the product line fault tree as shown in Figure 26 and Figure 27. When developing the FTA for the product line, we identify which PLUS model provides the information required by each fault tree layer.

The description for different layers of the fault tree, their relationships with the PLUS model and feature diagram are as following:

• In the wheel loader case, one of the top undesired events can be **Unintended Movement when Working at the Work Site**. Considering the supporting from PLUS model, the use case diagram generally describes the use cases relate to the top undesired event. As to the feature diagram, the first layer contributes the information needed by the fault tree on this layer.

• For the second layer of the fault tree, which models the product line functions, indicating the **immediate, necessary and sufficient** [2] cause to the top event. As shown in the Figure 26, the cause events are generally categorized as **Human Error**, **CDC Function Failure** and **Steering Wheel Control Failure**. Since we mainly model the CDC function in the former work, we investigate the events relate to CDC function in detail. The **CDC Function Failure** happen only when both the **CDC Supervision Failure** and the **CDC Control Failure** happen. All the events cause those two failures are shown in Figure 26. For simplicity reason, in the later layer analysis, we only consider the events **Joystick Steering Control** and **Joystick Forwarding(Reversing) Control**.

The feature diagram and the context diagram of the PLUS model support this layer by providing the functions and context of the product line; the feature diagram layer 2 to layer 4 contribute to this layer by providing the functions of the software product line.

• The third layer of the fault tree models the product line system failures. For the product line system, the failure can be caused by either the **Hardware Failure** or the **Software Failure**. On this level, we should consider the interferes among different functions. To be specific, in the wheel loader case, as we mentioned in Section 3, the CDC function shares the same functional control block with the boom function, and the CDC steering signal is the same type of signal as the boom lifting (lowering) signal, it is possible that the CDC steering failure is effected by
the boom function. Therefore, in analyzing the Hardware Failure, except all the hardware failures related to CDC, we have to consider the pinning failure, for example the one caused by the Wrong Wiring from Boom Function. Similarly, the Software Failure should consider the Boom Software Failure caused by the External System Software Failure, as shown in Figure 26.

For the Joystick Forwarding(Reversing) Control Failure, we still need to control the sub tree on lower layers. Different from the Joystick Steering Control Failure, the Joystick Forwarding(Reversing Control Failure) does not affected by the boom function, we only need to consider the Hardware Failure related to the CDC forwarding (reversing) function.

Considering the relationship with the PLUS model, this layer get the information from the PLUS context diagram. Since the feature diagram models only the features of the software product line, it has no contribution to the fault tree on this layer.

- On the fourth layer of the fault tree, the single system fault flow is investigated. The event on this level is derived from the PLUS model communication diagram and statechart diagram.

As shown in Figure 26, the objects that can cause the CDC Software Failure are: “:Joystick Interface” Failure, “:CDC Function Control” Failure and “:PWM Valve Interface” Failure. Those failures are determined from the communication diagram for Joystick Control in Four Directions use case (see Figure 20). Since the CDC Function Control object is state dependent control object, meaning that the object contains a statechart. Thus the “:CDC Function Control” Failure happen only when the CDC Control Status Failure and “CDC Disabled” State Failure happen together. The CDC Control States Failure is caused by either of the states failure: “Joystick” State Failure, “Steering” State Failure, “Forwarding(Reversing) Steering” State Failure. Those states are determined from the statechart, as shown in Figure 25.

In Figure 27, the objects that cause the Software Failure are: “:Joystick Interface” Failure, “:CDC Function Control” Failure, “:Engine Monitor Interface” Failure and “:Gearbox Monitor Interface” Failure. Those failures are determined from the communication diagram for Joystick Control in Four Directions use case (see Figure 20). As we stated above, the “:CDC Function Control” Failure happen only when the CDC Control Status Failure and “CDC Disabled” State Failure happen together. The CDC Control States Failure is caused by either of the states failure: “Joystick” State Failure, “Forwarding(Reversing)” State Failure, “Forwarding(Reversing) Steering” State Failure. Those states are determined from the statechart, as shown in Figure 25.

The fourth layer of the fault tree depicts the dynamic aspect of the fault, i.e., the fault flow. The dynamic information is provided by the PLUS model communi-
cation diagram and statechart diagram. The dynamic analyze is important for modeling the behaviors of the system, which contribute to creating the system’s functional safety requirements [7].
Figure 26: Wheel loader product line fault tree.
Figure 27: Wheel loader product line fault tree (cont’d).
4.2 Modeling Techniques Evaluation

4.2.1 Evaluation Matrix for Safety Analysis

Section 4.1.4 presents the wheel loader product line fault tree for a specific hazard. When building the fault tree, we mapped the PLUS model and feature diagram mentioned in Section 4.1.1 to different layers of the fault tree. In this section, we evaluate the modeling techniques mentioned in Section 2.2 by mapping them to different layers of the wheel loader product line fault tree. The evaluation matrix is shown in Table 4.

<table>
<thead>
<tr>
<th>Fault Tree</th>
<th>Modelling Technique</th>
<th>PLUS</th>
<th>Feature-oriented Modelling</th>
<th>Decision-oriented Modelling</th>
<th>SysML-based Modelling</th>
</tr>
</thead>
<tbody>
<tr>
<td>Layer 1: Top Undesired Event</td>
<td>Use Case Diagram</td>
<td>Layer 1</td>
<td>N/A</td>
<td>Use Case Diagram</td>
<td></td>
</tr>
<tr>
<td>Layer 2: Product Line</td>
<td>Feature Diagram</td>
<td>Layer 2-4</td>
<td>Variability Analysis</td>
<td>Internal Block Definition Diagram</td>
<td></td>
</tr>
<tr>
<td>Layer 3: Product Line System</td>
<td>Context Diagram</td>
<td>N/A</td>
<td>Variability Analysis</td>
<td>Block Definition Diagram</td>
<td></td>
</tr>
<tr>
<td>Layer 4: Single System</td>
<td>Communication Diagram</td>
<td>N/A</td>
<td>N/A</td>
<td>Activity Diagram</td>
<td></td>
</tr>
</tbody>
</table>

Table 4: Modeling technique evaluation matrix. The models built from the corresponding modeling techniques are mapped to each layer of the wheel loader product line fault tree.

We have presented the relationships between the wheel loader product line fault tree and the PLUS model, the feature diagram of the feature-oriented modeling in Section 4.1.4. We only discuss the relationships between the fault tree and the decision oriented-modeling, SysML-based modeling in this section.

The decision model proposed by [29] depicts the product line variability life cycle stages. The variability may exist in the product line features or the product line system. Specifically, the variants may exist in the hardware, software or characteristics of the product line system. Based on the knowledge we have now, the decision-oriented model is mainly used for depicting the product line variability, thus it is mapped to the layer 2 and layer 3 of the product line fault tree.

Habli et al. [7] mention that the SysML-based modeling can be used for modeling product line by adding complex relationship to the current model. In that case, the system
requirements model is described by the use case diagram, which is similar to the PLUS use case diagram, thus similar information can be modeled and being mapped to the first layer of the product line fault tree. The context model is described in a block definition diagram, which models the environment of the system. The context model plays similar role as the PLUS context model, thus it is mapped to the layer 2 and layer 3 of the wheel loader product line fault tree. The feature model is described by the internal block definition diagram, which models the system operating status and functions. The feature model is similar to the PLUS feature model, thus it is mapped to the layer 2 of the wheel loader product line fault tree. The bottom layer of the fault tree contains the single system dynamic aspect, which is addressed by the SysML-based activity diagram, depicting the system behaviors.

After mapping the models to the product line fault tree, we evaluate those modeling techniques from the safety analysis perspective. We derive evaluating criteria for the modeling techniques by modifying the principles proposed by Pumfrey [23]. The evaluation criteria are presented in Table 5.

1. Being applied in the safety analysis, the model provided by the modeling techniques should have a clear definition for the suitable use of the model, the required source material of the model and the expected products from the model [23]. The defined criteria is Clear Role in Safety Analysis.

According to [47], considering the safety aspect of the model-based development at the software level, the model employs “editable, hierarchical block diagrams and extended state transition diagrams”. The graphical editor provides the intuitive development and depiction of the system. The hierarchical diagrams is applied for control the complexity. The state transition diagram models the system dynamic behavior.

The PLUS model contains all the three types of models required by the safety standard. Each model has suitable definition for the usage, the required material of the model is clearly defined, the expected result derived from the model is clear as well. The other three modeling techniques do not produce models contain enough information as the PLUS model.

2. Other than judging the role each model plays, the information provided by the model should be valuable for safety analysis, as presented in the criteria Valuable Result for Safety Analysis. In other words, the model derived from the modeling techniques should have determined information that required by the safety analysis. Among the modeling techniques we investigated, identifying valuable information for the safety analysis from the decision model is not straight forward and more investigations are needed.

3. By using the modeling technique in an industrial context, the product line model should be delivered quickly to meet the project time scales. This criteria is defined as Modeling Process is Time-benefit.
4. Since the modeling techniques are used for modeling product line, it is required that the model contain product line information, which is defined by the criteria Model Starts on Product Line Level. It is desirable that the model contains single system information.

Among all the modeling techniques we discussed in the report, the PLUS model, feature diagram and the decision model start with modeling the product line. Moreover, the PLUS method models the single system of the product line. Currently, the SysML-based Modeling is mainly used for depicting the single system. However, the SysML model derived from the SysML-based Modeling describes similar information as the PLUS model, we believe that the SysML-based Modeling can be developed to model the product line system.

5. The Traceability is defined as the degree to which a relationship can be established between two or more products of the development process, especially products having a predecessor-successor or master-subordinate relationship to one another; for example, the degree to which the requirements and design of a given software component match [48]. In industry, the traceability is used for describing the possibility to trace the identified requirement to the specific analysis phases, i.e., software development, testing.

For all the modeling techniques described in this work, none of them are traceable. Because the models do not have identifiers, version control, traces to requirements etc.

6. Hazards relating to vehicle functionality is partly realized by the E/E system. For analysis and design, it is important to distinguish the environment that the system interfaces with. With defined environment, the possible hazards can be traced to specific embedded system components. The criteria relate to the environment modeling is Model Contains Context Information.

For analysis and design it is important to have the environment and its interfaces clearly defined. Possible hazards can be traced to components of the embedded system.

In the PLUS model, the context diagram depicts the context information explicitly. Similar model is defined in the SysML model derived from the SysML-based Modeling. The context information is mixed with the system information in the feature diagram and the decision model.

7. It is desirable that the Model Contains Dynamic Information. The dynamic information is crucial for guiding the design and reducing the testing cost. Normally, the dynamic information is not included in the fault tree. Therefore, the possible failure caused by the dynamic interaction can only be detected in the late software development phase, like implementation. Identifying missed dynamic interaction failures during implementation phase might lead to re-design, which is costly in
industry. On the other hand, the dynamic information improves the quality of the testing.

PLUS model and SysML model provide dynamic information, which depict the system behavior. Feature diagram and decision model do not contain dynamic information.

8. The safety analysis must consider both hardware and software, because the safety is a system property, meaning that it can be effected from either the undesired execution of software or unintended behavior of hardware.

The PLUS model covers both the software and hardware information, but with more concern in the software information. The hardware information revealed in the PLUS model is limited to the static model, with no detailed hardware description. Similar as the PLUS model, the SysML model contains both hardware and software information. While the feature diagram and decision model contain only the software information.

The criteria mentioned above is used for evaluating the modeling techniques from the safety analysis perspective. Besides supporting the safety analysis, the product line models are used for domain analysis as well. The evaluation criteria relate to domain analysis are described in Section 4.2.2.
<table>
<thead>
<tr>
<th>Evaluation Criteria</th>
<th>PLUS</th>
<th>Feature-oriented Modelling</th>
<th>Decision-oriented Modelling</th>
<th>SysML-based Modelling</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clear Role in Safety Analysis</td>
<td>Yes</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>Valuable Result for Safety Analysis</td>
<td>Yes</td>
<td>Yes</td>
<td>N/A</td>
<td>Yes</td>
</tr>
<tr>
<td>Modelling Process is Time-benefit</td>
<td>More time consumed than the feature diagram</td>
<td>Less time consumed than the PLUS modelling</td>
<td>Notrealized in this work</td>
<td>Notrealized in this work</td>
</tr>
<tr>
<td>Modelling Starts on Product Line Level</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Normally on single system level</td>
</tr>
<tr>
<td>Traceability</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>Model Contains Context Information</td>
<td>Yes</td>
<td>N/A</td>
<td>N/A</td>
<td>Yes</td>
</tr>
<tr>
<td>Model Contains Dynamic Information</td>
<td>Yes</td>
<td>N/A</td>
<td>N/A</td>
<td>Yes</td>
</tr>
<tr>
<td>Model Contains Hardware Information</td>
<td>Yes</td>
<td>N/A</td>
<td>N/A</td>
<td>Yes</td>
</tr>
<tr>
<td>Model Contains Software Information</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Table 5: Product line modeling technique evaluation matrix for safety analysis. The criteria are proposed based on the computer system safety analysis criteria, with concern in the product line safety analysis.
4.2.2 Evaluation Matrix for Domain Analysis

The evaluation in this section is mainly based on the criteria for domain analysis methods classification and evaluation presented by Kang et al. [18] and Firth et al. [49]. In [18], a set of criteria is presented for guiding the development of the domain analysis methods by evaluating existed domain analysis products and processes. We modify those evaluation criteria and propose the criteria for evaluating the product line modeling techniques, the criteria and the modeling techniques evaluation result are indicated in Table 6. The description of those criteria and their application are as following:

1. How the producer gets and uses the model information. The criteria is defined as Extraction of Information.

   The easier to get the model information, the easier the modeling technique can be realized. It is desirable that the model can make good use of the information. PLUS and SysML-based modeling need product functions and knowledge of the system. It is easy to get the machine function, before modeling the system, it is necessary to have the system knowledge. The Feature-oriented Modeling needs system characteristic information, normally, such kind of information is not presented explicitly, while it can be derived from the functional requirements. To this extent, the Feature-oriented Modeling does the similar analysis process as the PLUS model. The decision model is normally used for depicting the product line variability, thus the knowledge of system characteristic and variants are necessary for the decision model.

2. The Ease to Build Model, the Tool Support to the model, the Flexibility of the model. In other words, the model should be as simple as possible, the model does not prevent appropriate novel idea being developed based on it. The model can be built by a commonly used modeling tool. The model can be used for addressing the specifics of different domains and development contexts [50].

   It is desirable that the modeling process is simple and easy to follow. But there is the trade-off between the modeling workload and the information contained in the model. Considering the two modeling techniques we implemented in this report: PLUS model for the PLUS process and the feature diagram for the Feature-oriented Modeling. For the PLUS process more information is collected compare to the Feature-oriented Modeling. The effort for understanding, building and maintaining the PLUS model is much higher than the feature diagram. The PLUS models cover different software engineering phases: requirement modeling, analysis modeling, design modeling and detailed design modeling. While the feature diagram only covers the analysis modeling only. As to the rest two modeling techniques, SysML-based Modeling contains similar information as the PLUS modeling, we can expect that the workload is similar as the PLUS modeling. Decision-oriented Modeling considers the analysis modeling as the Feature-oriented Modeling, however, since
there is no consensus notation for the decision modeling, we expect that it is more complicated than the feature diagram.

The tools support each modeling technique should be easy to get, easy to learn without special request. UML is a widely used and well-developed modeling language, which supports the PLUS model, feature diagram and decision model. The PLUS model extends the UML by introducing new stereotypes, the modification is acceptable and easy to understand. According to the discussion in Section 2.2, we find that UML does not fit for feature Feature-oriented Modeling and Decision-oriented Modeling very well. The authors propose to use other implementation methods, which are not as common as UML, thus not easy to be accepted as UML. Other than UML, decision model uses textual modeling and other implementation method, the textual modeling cannot provide straight forward information as the UML does. The SysML-based Modeling is supported by SysML, which is the extension of UML, with more concern in the system modeling.

It is desirable that the modeling techniques can be used in different domains and contexts with less changes to the model. The PLUS model is flexible since the models address the context information separately. Moreover, the PLUS model has a clear definition of the system boundary. The feature model defines the context of the system as individual features, thus it is flexible to modify those features. The decision model concerns the variants of a system, which depends on the system context as well, thus the model is flexible in modeling those context variants. The SysML-based Modeling is depicted by the SysML model, which separates the different aspect of a system modeling, i.e., context, system characteristic, thus it is flexible.

3. Whether the model is easy to maintain - Maintenance: the model can be adapted for new market requirements, different customers, or technological needs [51].

As we described for the flexibility criteria, the PLUS model separates different concerns of the product line system. It is easy to maintain those individual models. The SysML-based model has similar concept as the PLUS model, i.e., use separate model to describe the different concerns. Thus the SysML-based model is easy to maintain. For the feature diagram and the decision model, when a system only contains simple characteristic information, it is easy to maintain the two models. However, if the system contains complicated characteristic information and dependency relationship, it is hard to maintain the feature diagram and the decision model.

4. How do the modeling techniques support the software development phases:
   Requirements Modeling phase, Analysis Modeling phase, Design Modeling phase, Implementation phase and Testing phase.

The PLUS model covers the first three phases of the software development process, i.e., the requirements modeling phase, the analysis modeling phase and the design
modeling phase. The SysML-based model is mainly used for single system modeling, which covers the first three phases of the software development process as well. The feature diagram and the decision model only cover the analysis modeling phase.

<table>
<thead>
<tr>
<th>Technique Evaluation Criteria</th>
<th>PLUS</th>
<th>Feature-oriented Modelling</th>
<th>Decision-oriented Modelling</th>
<th>SysML-based Modelling</th>
</tr>
</thead>
<tbody>
<tr>
<td>Extraction of Information</td>
<td>Product function Knowledge of system working principle</td>
<td>Features Knowledge of system characteristics</td>
<td>Variants Knowledge of system characteristics</td>
<td>Product function Knowledge of system working principle</td>
</tr>
<tr>
<td>Ease to Build Model</td>
<td>Different types of model</td>
<td>1 type of model</td>
<td>Different types of model</td>
<td>Different types of model</td>
</tr>
<tr>
<td>Tool Support</td>
<td>UML</td>
<td>UML etc.</td>
<td>Textual Modelling, UML etc.</td>
<td>SysML</td>
</tr>
<tr>
<td>Flexibility</td>
<td>Flexible</td>
<td>Flexible</td>
<td>Flexible</td>
<td>Flexible</td>
</tr>
<tr>
<td>Maintenance</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Software Development: Requirements Modelling</td>
<td>Yes</td>
<td>N/A</td>
<td>N/A</td>
<td>Yes</td>
</tr>
<tr>
<td>Software Development: Analysis Modelling</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Software Development: Design Modelling</td>
<td>Yes</td>
<td>N/A</td>
<td>N/A</td>
<td>Unknown</td>
</tr>
<tr>
<td>Software Development: Implementation</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>Software Development: Testing</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
</tr>
</tbody>
</table>

Table 6: Product line modeling technique evaluation matrix for domain analysis. The criteria are modified from the domain analysis methods evaluation criteria, which mainly consider the evaluation from product line modeling perspective.

The modeling techniques mentioned in this report fit for different scope of product line modeling. In order to select the appropriate modeling technique for product line functional safety analysis, both the criteria presented in Table 5 and Table 6 should be
taken into account.
5 Conclusions

Nowadays, product line engineering is widely used in industry. How to manage functional safety in product lines is one of the challenges for the product line approach. To address this question, it is important to investigate what information is necessary to perform the product line safety analysis and how the information is derived. As the main contribution of this work, we modeled the product line and found the possibility to derive the information required by product line functional safety analysis from the product line model.

5.1 Safety-critical System Product Line Modeling

The product line model is the outcome of product line modeling process, there is a wide range of modeling techniques that can be used for modeling product lines. Before getting started with modeling, the different modeling techniques need to be evaluated. The purpose of evaluation is to identify whether the techniques meet the requirements for modeling a product line. The results of our work are summarized as following:

As the first step of product line modeling, we investigated the product line modeling techniques. Then we chose some of the modeling techniques to study in detail. Those techniques are Product Line UML-based Software engineering (PLUS) process, Feature-oriented Modeling, Decision-oriented Modeling and SysML-based Modeling. We chose them for further discussion because we deem that those modeling techniques can be implemented. The chosen techniques cover different general concepts for modeling product lines. We studied the wheel loader product line industrial case as the third step, in order to gather sufficient information for modeling, we focused on the PLUS model because it contains more information than the feature diagram derived from the Feature-oriented Modeling. Moreover, PLUS model has the potential to be widely applied in the industrial domain. Because the tool support of PLUS model, i.e., the UML is easy to get and cheap. The PLUS model extend the UML stereotype, thus with the UML knowledge, no special training is expected for the PLUS modeling.

Considering the product line functional safety, we first investigated the different safety analysis techniques. As the result, we chose the FTA as the main technique for the product line functional safety analysis. We investigated how the fault tree analysis can be used for covering the possible failures of all product family members and investigated which information is required by the product line fault tree. In the following step, we investigated how the four modeling techniques described above contribute to the product line hazard analysis. As a result, we proposed an evaluation matrix which presents the relationship of the product line models and the product line fault tree.

As the extension, the product line variabilities can be depicted in the product line fault tree. The variants can exist on different layers of the fault tree. Specifically, on the fault
tree product function layer, the features can be variants, on the fault tree product line system layer, either the software or hardware can be variants. The product line fault tree has same structure but different content for different top undesired event. In the application engineering, the product line fault tree can be reused for the new product line members, which improve the product line safety analysis [40].

Finally, we derived the evaluation criteria and created two other evaluation matrices in which the performance of the modeling techniques in domain analysis and safety analysis is evaluated.

5.2 Future Work

Though PLUS provides comprehensive information for product line fault tree, the information is not sufficient for safety analysis. The PLUS model can be further improved so that it can provide more useful information for the safety analysis.

As we mentioned in Section 2.3, an ASIL/SIL is introduced for determining the risk classes. In the FTA, the ASIL/SIL is used for fault tree quantitative analysis, from which we can determine the safety-critical element, meaning that the failure of the element could lead to safety problem. As the fault tree presented in Figure 26 and Figure 27, some of the basic events are classes/objects or states of the PLUS model. By defining the ASIL/SIL level of those basic elements, we can derive the safety-critical classes/objects, states in the PLUS model, which need to be considered in safety analysis. To assign the safety property to the PLUS model classes/objects or states will result in introducing new stereotype to the PLUS model, i.e., the stereotype define from safety perspective.

As we mentioned in the PLUS static model in Section 4.1.2, for those hardware depicted in the static model, their failure probability can be identified. The hardware quantitatively functional safety analysis is applied on the hardware device with identified failure probability. The safety-critical device can be determined from the failure probability, which provides the objectives for safety analysis.

Since the product line fault tree can be used in the early design phase [2], meaning that the product line fault tree can be used for determining the system requirement and system design. For each of the function derived from the product line fault tree, it is desirable to trace it to the PLUS model, so that the function can be better analysed.

The product line modeling techniques: SysML-based modeling and decision-oriented modeling can be further improved. Currently, the SysML-based modeling is mainly used for depicting the single system, according to [7], the SysML-based modeling can be extended to model the product line by adding extra complex relationship between the product line's models. We expect that the decision-oriented modeling can reach a consensus notation in the further, so that the modeling technique is easy to follow and more applicable.
After the safety analysis, a clear and convincing argument is desirable, which proves that the system is acceptably safe. The main method being applied for safety argumentation is the goal structuring notation (GSN). The information required by the goal structure can be derived from the product line fault tree, which reduce the cost.

Considering the embedded real-time software product line design, as we described in Section 4.1.3, the main concern is to design a highly configurable software product line architecture. In the deployment time, the component-based subsystem could be freely mapped to different computer nodes (ECUs). In that case, in the product line application, the component-based system can be allocated to its own computer node or be allocated to a specific computer node together with its subsystems. The design decision increases the deployment flexibility.
References


