Model-Based Product Line Engineering in an Industrial Automotive Context: An Exploratory Case Study

Damir Bilic
Mälardalen University
Västerås, Sweden
damir.bilic@mdh.se

Daniel Sundmark
Mälardalen University
Västerås, Sweden
daniel.sundmark@mdh.se

Wasif Afzal
Mälardalen University
Västerås, Sweden
wasif.afzal@mdh.se

Peter Wallin
RISE SICS
Västerås, Sweden
peter.wallin@ri.se

Adnan Causevic
Mälardalen University
Västerås, Sweden
adnan.causevic@mdh.se

Christoffer Amlinger
Volvo Construction Equipment
Eskilstuna, Sweden
christoffer.amlinger@volvo.com

ABSTRACT

Product Line Engineering is an approach to reuse assets of complex systems by taking advantage of commonalities between product families. Reuse within complex systems usually means reuse of artifacts from different engineering domains such as mechanical, electronics and software engineering. Model-based systems engineering is becoming a standard for systems engineering and collaboration within different domains. This paper presents an exploratory case study on initial efforts of adopting Product Line Engineering practices within the model-based systems engineering process at Volvo Construction Equipment (Volvo CE), Sweden. We have used SysML to create overloaded models of the engine systems at Volvo CE. The variability within the engine systems was captured by using the Orthogonal Variability Modeling language. The case study has shown us that overloaded SysML models tend to become complex even on small scale systems, which in turn makes scalability of the approach a major challenge. For successful reuse and to, possibly, tackle scalability, it is necessary to have a database of reusable assets from which product variants can be derived.

CCS CONCEPTS

• **Software and its engineering** → *Software product lines*;

KEYWORDS

System product lines, Model-based systems engineering, Variability management, Orthogonal variability modeling

ACM Reference Format:
Damir Bilic, Daniel Sundmark, Wasif Afzal, Peter Wallin, Adnan Causevic, and Christoffer Amlinger. 2018. Model-Based Product Line Engineering in an Industrial Automotive Context: An Exploratory Case Study. In *22nd International Systems and Software Product Line Conference - Volume B (SPLC '18), September 10–14, 2018, Gothenburg, Sweden.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3236405.3237200

1 INTRODUCTION

Product development is about satisfying the needs and requirements of customers. In order for a product to be successful it must be well adapted to all its customers. However, different customers might need similar products with subtle differences, but also they might have completely different requirements for the same product. This is apparent in the automotive domain, where product configurations can be very specific to customers and specific to the region where the product is to be distributed. Having standardized products is sometimes challenging, especially in recent years, where software is becoming part of products in all domains. Such software-intensive systems introduce even more variations within products [20].

The first software used in the automotive industry was used to control the engine, namely, to control the engine spark timing in the GM Oldsmobile Tornado [4]. Nowadays, software is used to control the complete engine behavior. The same applies to the diesel engines considered in our case study at Volvo Construction Equipment (Volvo CE). Volvo CE produces heavy machinery such as excavators, articulated haulers, wheel loaders, etc. The engines used in heavy machines have several thousand parameters which need to be configured for each product variant individually.

In order to facilitate reuse within the engine system, while reducing time to market and cost of development, it is necessary to address variability, starting from variable user requirements, through the design process, implementation and later coping with variants during validation and verification procedures [3].

Traditionally, when a system is to be developed, it is rarely developed from scratch. Rather, it is considered as an evolution of an existing system. The same is true at Volvo CE, where new generation engines are developed by improving the functionality and performance of previous systems. Even though reuse of artifacts such as specification documents or test procedures is a fairly common practice in the industry, additional work is necessary to define reuse strategies within the systems engineering domain [24].

Model-Based Systems Engineering (MBSE) is a methodology proposed to address issues in the systems engineering process, by using models to support systems engineering activities [7]. Adopting a model-based approach leads to a paradigm shift from document-based representation of information to having models as a central source of information [19]. A visual modeling language widely used in MBSE is SysML [8]. It is indifferent to any specific systems engineering methodology [7].

This paper presents the first steps towards the adoption of model based product line engineering within the Engine Controls department at Volvo Construction Equipment. The main objective is to create design models of the the engine systems with respect to variability. The secondary objective is to identify potential challenges that arise while modeling variability within the MBSE process.

The case study was conducted on the Urea Dosing System (UDS), which is a part of the exhaust aftertreatment system. Two types of UDS systems have been modeled and combined into a single, overloaded, SysML model. The model was then annotated with Orthogonal Variability Modeling constructs. A successful derivation of individual UDS variants from the overloaded model has been demonstrated.

An observation made in the study is that the overloaded model of the UDS became complex even though the UDS is a fairly simple system. This in turn makes scalability a great challenge. A possible approach to tackle scalability is to divide a complex system into manageable subsystems and model them as a database of reusable assets from which product variants can be created.

2 BACKGROUND

Systems engineering represents a set of processes that, when combined, bridge the gap between customer requirements and the system solution which is intended to satisfy the customer requirements. From a technical perspective, it includes activities to analyze, specify, design, verify and validate the product to be developed to ensure that customer requirements are satisfied [10]. With the advancements in technology, systems are getting more complex and it is necessary to have an efficient systems engineering process that allows for communication between different domains. MBSE is an evolving approach targeting these challenges [19].

2.1 Variability Modeling

To address product variability in any domain it is first necessary to identify commonalities and differences between product variants. Product variants appear as a consequence of decisions made by system designers in order to address different user requirements. They can appear throughout all design artifacts, from requirements, to design specifications, test cases etc. Variability modeling is the process of documenting and defining variants and constraints between variants within product artifacts in order to increase understanding and provide an overview of the system variability [22].

Variability can be viewed from two perspectives. The first one being the variability in the problem domain where a specific product is decomposed as a set of features that can appear within that product. The features in the problem domain represent what functionality must be implemented in the system, e.g., the base engine system needs to have a Turbocharger, an Engine Brake etc. On the other hand, solution domain variability represents the variability within design artifacts which are used to extend, customize or change the system in order to implement product features. In other words, solution domain represents how the features are implemented in the given system [15]. For example, there are several types the Turbocharger, or the Engine Brake can be implemented in several different ways.

Variability information, from the problem domain perspective, is usually captured in separate models [5]. Feature models and decision models are the most common way of representing variability in the problem domain. Several feature-based modeling methods can be identified in literature and comprehensive studies on the research can be found in [1, 20]. Most feature-based modeling approaches are derived from the feature-oriented domain analysis [13]. Feature models represent user-visible aspects, or features, of a product line, with relationships between them. From feature models it is possible to derive all possible variants of the product line. However, as said, to create a feature model, it is necessary to identify commonalities and differences within a product line. Guidelines on how to extract and structure features can be found in [14].

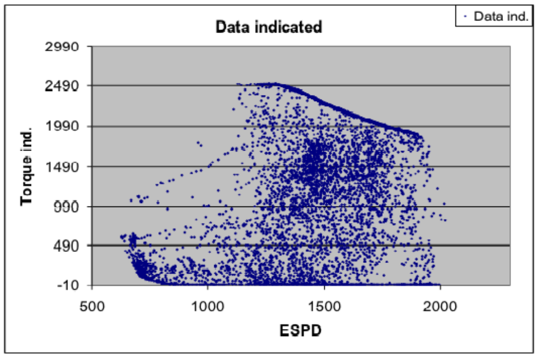
After identifying features of a system, it is necessary to present these in a model. In [6] it is proposed that notations for feature models should be graphically represented in a readable and clear form. They should distinguish between types of variability (options, alternatives etc.). It must be possible to specify properties of variation points, such as cardinalities. Additionally, it is necessary to represent dependencies between features. Feature models should support evolution over time and provide a way to manage traceability between versions. They need to be adaptable to specific needs, scalable and unified for the whole development cycle. A report on the experience from three industrial settings that use feature modeling for variability management can be found in [2]. They conclude that the primary benefit of variability modeling lies in the ability to organize, visualize and scope features.

Other than feature models, there are decision models which essentially represent a fixed set of questions and answers that are used to derive a product. A systematic literature review on decision-oriented variability modeling techniques can be found in [21].

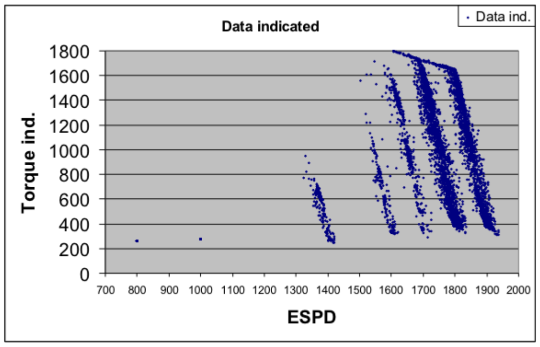
In order to address the solution domain it is necessary to address the variability in the systems engineering process. That means variability has to be considered throughout the development of requirements, system analysis, system design, testing procedures etc. [15]. Using SysML in MBSE allows the representation of complex systems throughout the systems engineering process using one modeling standard [7]. However, SysML does not provide language constructs to represent variability. Nevertheless, it is possible to express variability in SysML by, for example, extending the standard notations with custom profiles [23].

Feature and decision models in the problem domain are presented separately from, but linked to, development artifacts in the solution domain [5]. Derivation is performed such that features are selected in the separate models and a set of development artifacts is derived according to the selected features. According to [2], managing traceability in an efficient manner between models in the problem and solution domain is key for variability management.

An approach that allows the combination of variability models from the problem domain with models from the solution domain is proposed in [18]. Orthogonal Variability Modeling (OVM) is an annotative approach which enables the annotation of SysML models with variability constructs. In this case, both design and variability are presented in the same model with explicit links between variants and parts of the design models.



(a) Articulated Hauler



(b) Excavator

Figure 1: Machine Duty Cycle

3 CASE STUDY DESIGN

The Engine Controls department at Volvo CE is in the initial phase of migrating from the traditional document-based to the MBSE approach with the aim to increase reuse, reduce costs and reduce time-to-market of new products. For Volvo CE, variability management is a key factor for a successful reuse. An example of variability can be seen in Figure 1. The two machines: excavator (Fig. 1b) and articulated hauler (Fig.1a) are using the same base engine, however the required engine behavior is completely different between the machine types. The x-axis represents the engine speed in rotations per minute, whereas the y-axis represents the torque produced by the engine. It can be seen that an excavator engine (Fig. 1b) needs to produce a wide range of torque while maintaining a fixed engine speed. On the other hand, the engine of an articulated hauler needs to be able to produce a wide range of torque while operating on varying engine speeds.

There are two objectives with this study. Firstly, we want to create models of the engine systems at Volvo CE with regards to variability. The second objective is to identify potential challenges which arise due to the introduction of variability modeling in MBSE. Identifying challenges in early stages is necessary to plan ahead and adjust the MBSE practices to fit the specific context at Volvo CE.

3.1 Case and Study Object

The **studied case** is the Engine Controls department at Volvo CE in Eskilstuna, Sweden. Volvo CE develops, manufactures and markets equipment for the construction and non-road applications. The product range of Volvo CE includes different types of wheel loaders, excavators, articulated haulers and construction machinery.

The Engine Controls department at Volvo CE is the provider of engine systems for their complete range of products. In general, all engine systems have two main subsystems: (i) the diesel engine with the role to produce torque and (ii) the aftertreatment system with the role to reduce the amount of emission gases. In order to have a comprehensible scope for this exploratory study, the Urea Dosing System (UDS), part of the aftertreatment system, was selected as the study object of analysis. The UDS consists of physical components and software calibration parameters (e.g. linearizion parameters for the urea tank unit sensor, urea dosing pump injection calibration parameters etc.). These calibration parameters affect the behavior that is shown in Figure 1.

Due to regulations and norms imposed on diesel engines, it is important that the amount of pollutants in the exhaust stream are reduced below certain limits depending on the region where the machines are to be used [12]. The role of the UDS is to inject the diesel exhaust fluid (urea) into the exhaust in order to reduce nitrogen oxides (NOx) in the Selective Catalytic Reduction Catalysts.

The UDS was selected since it is a fairly isolated system and it is possible to analyze it in this exploratory study as an individual system. It includes both shared and variable components between the two different UDS types.

3.2 Case Study Procedure

The Orthogonal Variability Modeling (OVM) [18] language was selected to represent variability in the problem domain due to already existing tool support at Volvo CE. SysML, as a standard for MBSE, was selected to create system models in the solution domain.

An overloaded SysML model was created that captures both types of UDS systems in a single model. OVM was used to create a variability model of the UDS, with constraints between variations. OVM also allowed us to explicitly link the variations from the variability model to design models in SysML. The analysis and modeling was performed with support from expert system architects from the Engine Controls department at Volvo CE.

The PTC Integrity Modeler[1] tool was used for modeling activities. The derivation procedure is demonstrated in Section 4.

4 RESULTS

Most of the research on the topic of product lines and variability management concerns *software* product lines [20]. Although software is an essential component of Volvo CE products, it is necessary to address variability within the complete systems engineering process. That includes not just software but also other disciplines such as mechanical and electrical engineering, industrial engineering, project management and organizational studies [10]. Systems engineering is essentially the whole process of developing a complete system, hardware and software, based on customer requirements.

[1]https://www.ptc.com/en/products/plm/plm-products/integrity-modeler

4.1 Variability within the engine system

The engine system consists of two main subsystems, the base engine (responsible for producing torque) and the aftertreatment system (responsible for the reduction of emissions of the base engine) .

The base engines can be classified into three categories according to their size: heavy duty, medium duty and light duty engines. Engines of different sizes will be treated as separate product lines due to major differences in both hardware and software.

In addition, there are three types of aftertreatment systems, tightly coupled to the three categories of base engine platforms. The three aftertreatment systems consist of a set of mechanisms (such as the UDS, diesel particulate filters etc.) to reduce the emissions of the engine systems. However, some features of the aftertreatment system depend as well on the regulations and norms of the location where the machine is intended to be used. Additionally, software calibration parameters might vary depending on the environmental legislation as well. The regulations in the three largest markets of heavy machinery: US, China and Europe differ significantly in aspects such as allowed NOx emissions [12].

In some cases product features, or parts of features, are shared between different product lines. It is important to address these commonalities in order to reuse the design and development artifacts between product lines. One example is the urea dosing subsystem. It is a part of the aftertreatment system and its purpose is to support the reduction of NOx emissions within the engine system.

The following subsections will introduce the mechanisms used at Volvo CE in order to model variability within the problem and solution domains.

4.2 Modeling techniques and tools

The current reuse approach at Volvo CE Engine Controls is based on the fact that engines and engine control software are developed to be able to answer a wide range of different needs. However, a mechanism to manage variations in features (including calibration) of an engine is not clearly defined. When a new product is to be developed, a gap analysis is performed and the closest already existing product is replicated and modified according to the new set of requirements.

Variations and commonalities at the engine controls can range from user visible characteristics (e.g. type of the UDS system) to internal characteristics such as type of the NOx level sensor. By combining all system features into one model, an overloaded product line is created. The product line includes all possible internal and external variations that can appear within a given engine system.

The SysML standard does not support variability modeling explicitly. Nevertheless it is possible to express variability by creating SysML profiles to extend the language with variability concepts, as it can be seen in [25]. However this approach increases the complexity of models significantly. Variability management is an important aspects of product lines and in order to be successful in our approach of using SysML, an easy and holistic approach to formalize variability in all SysML models is required. The language we used for variability modeling is the Orthogonal Variability Modeling language [18]. The notation for OVM is shown in Figure 2 [18]. The connection between SysML models and the OVM models is represented with explicit links that are drawn between the variants

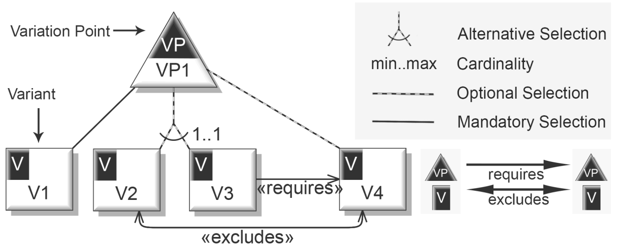


Figure 2: Orthogonal Variability Modeling language notation

and design blocks in SysML. The links between OVM and SysML are part of the OVM language. In case a variant is not selected, the linked SysML blocks need to be removed from the diagram.

OVM is used to capture the problem domain variability in diagrams that are separate from SysML diagrams. It is possible to represent variation points, variants, dependencies and constraints between variants, as illustrated in Figure 2. Variability of the solution domain is presented in the form of overloaded SysML models, where all possible variations of the solution are represented.

In this case, when a new product needs to be developed, again a gap analysis is performed. However, now instead of replicating a complete product, one would select features that satisfy the requirements of the new product. It is important to consider constraint rules between artifacts that must be satisfied. Constraints between artifacts are represented within the OVM models (Figure 2).

The next subsection provides a demonstration of the overloaded SysML models created for the urea dosing systems (UDS) including the procedure for derivation of a specific UDS type.

4.3 Demonstration on the Urea Dosing System

The system logical view of the UDS is shown in Figure 3, represented as a SysML activity diagram. The main three tasks of the UDS are:

- Storing the urea, represented by the Store UREA activity.
- Dosing the urea into the exhaust (Pressurize UREA activity).
- Defrosting the urea (Defrost Urea System activity).

The UDS receives dosing demands to the urea system via the DosingDemand interface from the Engine System. The dosing demand is processed and the Pressurize Urea activity is invoked. The pump (representing the Pressurize Urea activity) is then creating a suction effect to move the urea from the urea tank, pressurize it and inject into the exhaust stream through the exhaust port. Temperature sensor readings from the tank and pump unit are then sent back to the Process UDS demands activity where a temperature evaluation is performed. In case of low temperatures, the defrosting activity is invoked. Defrosting is necessary since the Volvo CE machines are supposed to work in environments with temperatures below -11°C, which is the freezing temperature of urea.

Depending on the engine system requirements, imposed from the machine level, there are two types of dosing systems. Air supported injection is when urea is initially pressurized by a pump and then mixed with high pressure air in order to disperse it into the exhaust stream. However, if pressurized air is not provided by the machine,

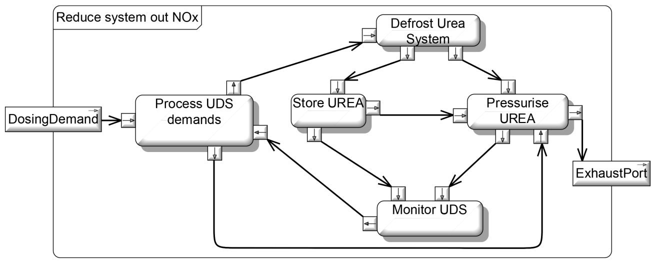


Figure 3: Urea Dosing System Logical Architecture

it is necessary to use a dosing system that can create high enough pressures to disperse the urea without having air support.

An overloaded SysML model that combines both types of the UDS is presented in Figure 5. It includes the urea storage tank unit, suction hose and pressure hose as shared components. It can be seen in Figure 5 that individual components from both systems are included in a single diagram. The blocks: Air Supported Pump, Air UDS ACM (Air UDS After treatment Control Module) and Air Dosing Nozzle are part of the air supported dosing system. The blocks: Dosing Pump, UDS ACM, Cooling Water Valve, Return hose, Dosing Nozzle are part of the UDS without air support.

The Urea Tank Unit from Figure 5 can be mapped to the Store Urea activity in Figure 3. The UDS control unit is mapped to the Process demands and Monitor UDS activities. The software, which defines behavior of the system resides within this block, it is parametrized with calibration parameters of individual blocks in the system.

The defrosting activity is such that the UDS is defrosted by circulating engine cooling water through the urea pump unit and the urea storage tank. Transport hoses are heated electrically from the respective UDS Controller.

The transport hoses, urea pump and injection nozzle from Figure 5 can be mapped to the Pressurize Urea activity in Figure 3.

As a component that is shared between the different dosing systems, the urea storage tank (as shown in Figure 5) is connected to both types of UDS in the overloaded model. There exist several variations within the storage itself: storage tank size, sensor type, direction of flow of cooling water and type of return hose port on the tank armature (Figure 7).

The direction of flow of engine cooling water through it is opposite for the two types of UDS, therefore the tank unit calibration parameters that define defrosting time are also different.

The UDS without air support has an additional hose between the pump unit and urea tank since some of the urea fluid is flowing back from the pump to the tank. The return hose is not explicitly stated as a variation and is included in the UDS System Type - Without Air Support variant. The return hose port is, however, represented as a variation point for the urea tank, since one could derive a variant of the tank unit without considering the complete UDS.

In the overloaded SysML model, the UDS components from the two systems connect to the same interfaces on shared components. E.g., both control units (Air UDS ACM and UDS ACM) are connected to the same tank port for sensor readings. However, only a single connection will remain when a specific UDS variant is derived.

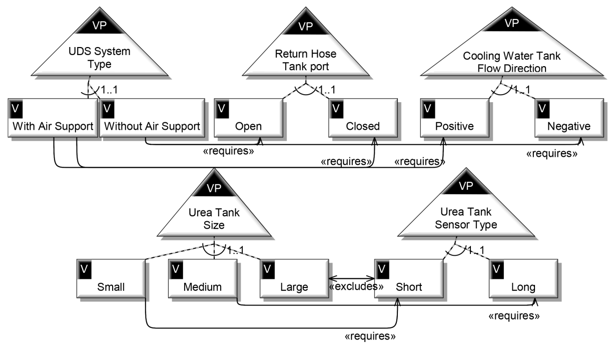


Figure 4: Variant Diagram of the UDS

An overview of all variation points with variants and dependencies between them can be seen in Figure 4. For example, if one selects the UDS with air support variant, then it is required to select the Closed variant of the Return Hose Tank port. Moreover, if one selects a Large variant of the Urea Tank Size, then the Short Variant of Sensor Type is excluded due to the dependency between them.

The overloaded SysML diagram from Figure 5 is annotated with OVM variability constructs in order to connect components of different UDS types (solution domain) to the variability model from Figure 4 (problem domain). After the variability model is connected to the SysML design model it is possible to start the derivation process. The PTC Integrity Modeler tool used for this demonstration allows the configuration of individual products through the Variant Selector tool capability. The Variant Selector provides a configuration interface based on the variability model in Figure 4. The derived system variant must satisfy the constraints from the variability model.

Figure 6 represents the derived air supported UDS variant. It can be seen that it was derived by removing all the blocks which are not part of the selected variant. The overloaded tank unit model is shown in Figure 7. The tank unit variation points are: tank size, tank sensor type, engine cooling water flow direction and a varying return hose armature port. During the derivation process of the air supported UDS from Figure 6, the variant of the Urea Tank Unit has been selected as shown in Figure 8. It includes a medium tank size, long tank sensor, armature without a return hose port (closed) and a positive flow of engine cooling water through the tank.

5 LESSONS LEARNED

Although internal block diagrams were used to represent the physical structure of an engine subsystem, it is possible to annotate any type of SysML diagram with variability constructs.

We have avoided using different diagram viewpoints to model the system as that would contribute to the complexity of overloaded models as each of the diagrams would need to be consistently annotated with variability constructs. Furthermore, since variants in physical components and their calibration parameters were considered in the current study, internal block diagrams were sufficient to express the required information.

The study allowed us to identify potential benefits and challenges which arise from variability modeling in the MBSE approach

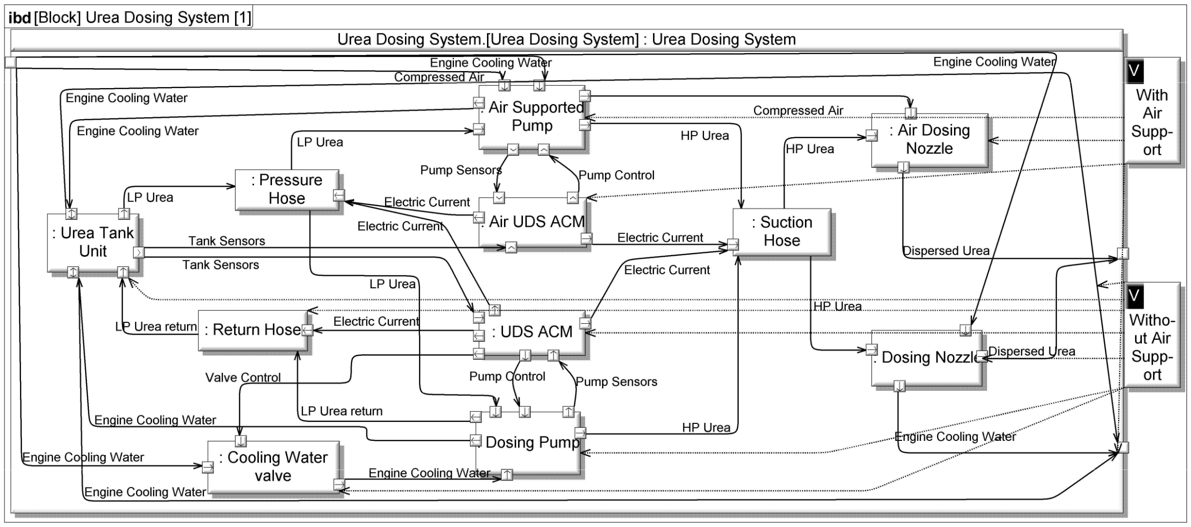


Figure 5: Internal block diagram of the overloaded model of the urea dosing system

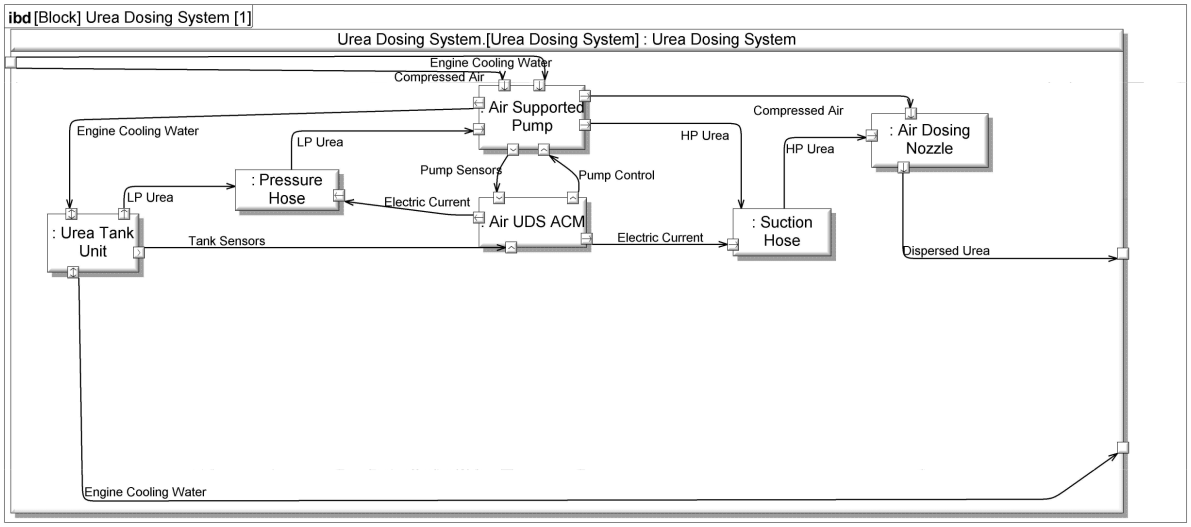


Figure 6: Internal block diagram of the Air supported UDS system

at Volvo CE. Overloaded models present a potentially good way of managing commonalities in a product line and allow for reduction of efforts required to maintain the commonalities between systems. For the engine system, an efficient approach to document and manage variability would have the benefit of firstly having a single source of information (the model).

Additionally, the configuration of an engine system (e.g. the UDS) would now allow for a simplified derivation procedure, from a single model, compared to the manual synchronization of several document-based artifacts. When the models are created, it is possible to derive a specific variant of a system or subsystem with little effort.

However, during the later evaluation of the models, we noted the following challenges:

- **Complexity**
  The creation of overloaded SysML models was shown to be a complex and error prone task. A single UDS contains a small number of components and there are only two different UDS types. Yet, when one combines them into a single overloaded model, the design becomes rather difficult to comprehend. That is seen in Figure 5. The variations in interfaces on the shared components need to be clearly defined. In the case of UDS, the tank unit had an additional interface for the return hose when the UDS without air support was used. Additionally, the flow of engine cooling water is reversed between the UDS types, meaning that an input for one system, in the overloaded model, can be an output for the other system. On the other hand, calibration parameters of each

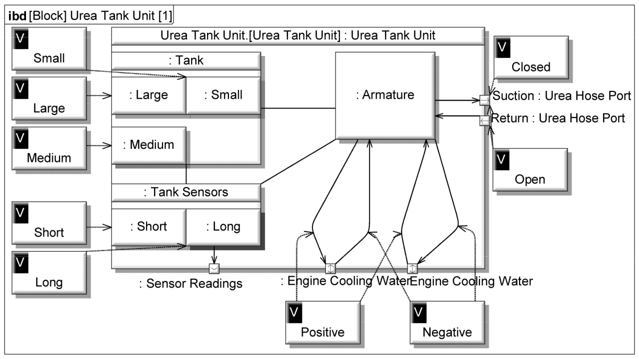


Figure 7: Internal block diagram of the overloaded model of the UDS tank unit

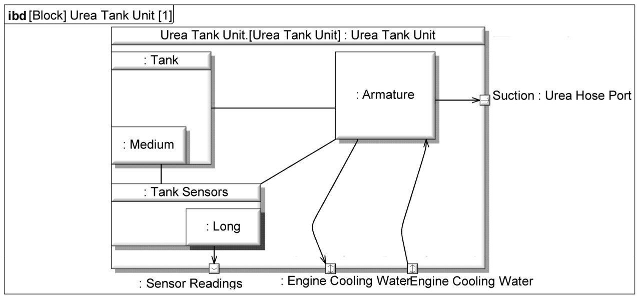


Figure 8: Internal block diagram of the tank unit for the Air supported UDS

of the physical components are stored within the respective blocks, it allows us an easier parametrization of the software by taking calibration parameters directly from a derived variant model. Due to the complexity of overloaded models, scalability represents a major challenge.

- **Scalability**
  In order to manage scalability and create comprehensible models, one possibility would be to have several layers of abstraction in the design process. In the case of the UDS (Figure 5), one could group together variable components based on the type of UDS. These component groups could be placed into separate internal block diagrams in order to simplify the overloaded model in Figure 5. Although the separation of concerns (observing variable and shared components individually) is fairly simple on the UDS, separating concerns within the complete engine system is no easy task. The engine subsystems are highly interconnected, interdependent and difficult to observe each on its own. Due to the complexity of the presented overloaded models, we conclude that this approach would be unfeasible for a larger scale. To address scalability, one possibility would be to create a library of assets, without interconnecting them into overloaded model as in Figure 6.
- **Asset Management**
  With an model-based asset library, we want to avoid the use of large overloaded models. It is necessary to divide the engine system into self contained subsystems as far as possible in order to create an extensive database of assets. An example of an asset would be the Tank Unit, it does not necessarily have to be modeled as a part of a larger subsystem as shown in Figure 5, rather it would be contained in the asset database. Another asset of the UDS example would be the injection system (ACM, pump unit and dosing nozzle). As long as the individual assets are consistently annotated with variability constructs, a derived UDS variant would still be a correct, although not interconnected in a diagram anymore (as in Fig. 6). With an asset database, it would be possible to derive a single product by selecting variants from the database according to system requirements. Constraints between assets must be clearly defined and maintained. Due to varying requirements, some assets might need modifications for a new product variant and it is necessary to keep track of different versions.
- **Evolution of assets**
  Asset changes are related to adjustments of subsystems for new machine variants (e.g. addition of a new urea tank size, change of the cooling water valve etc.). In the proposed asset library, incremental changes would be added to the currently existing assets in the form of additional variants of its overloaded model. For example, if a new tank sensor type is required, the asset shown in Figure 7 would be updated with a new type of the tank sensor. Furthermore, the variant diagram (Fig.4) would need to be updated with the new variant and its constraints similar to the approach presented for feature models in [16], except that in our case, the version number of the variability model would not change. In case changes to assets can not be made without affecting its current components, a new type of the same asset would have to be created. For example, if the modular Tank Unit would be replaced with a compound unit that integrates all elements into itself, it would be added as new tank asset to the library. Additionally, the product line variant diagram would have to be updated with the new assets and their constraints to other assets. A product version, at any time, would be defined as a certain selection of features from the same variability model since changes are introduced as new features (new assets or new features to existing assets).

It should be noted that no specific guidelines for structuring solution domain variability within MBSE have been followed for this research, as none are present in the current state of the art. These limitations will be addressed as a part of the continuation of research on this topic.

6 RELATED WORK

Variability management with OVM and overloaded SysML models is presented in [11]. An example on how to structure a library of reusable assets is demonstrated. The potential benefits of their approach are the reduction of development costs and reduction in time to market.

The use of SysML profiles to represent variants in different diagram views has been presented in [23]. They used the IBM Rational Rhapsody/Gears bridge tools to model different variants for a wind turbine cooling system. They demonstrated successful use of SysML with stereotypes to manage product line variability. However, the paper does not describe how the SysML variants are mapped to feature diagrams and it is not shown how a specific variant can be derived by utilizing the feature models.

A study on the use of MBSE within product lines from the rail transportation industry is presented in [9]. They report on first steps towards introducing MBSE into PLE by creating a product line of a metro train. The product line includes development artifacts from which all possible alternatives can be derived. The Orthogonal Variability Modeling language [18] was used to define problem space variability. They estimate cost savings on fixed engineering costs in the specification phase for up to 50%.

Structuring product lines and mapping variability models is not an easy task, an industrial report on the topic is presented in [17]. They found that most of the challenges when adopting product lines and variability modeling in an automotive context arise from lacking modeling guidelines and limited scalability of current approaches.

7 CONCLUSIONS AND FUTURE WORK

In this paper, we presented an exploratory study on the introduction of model based product line engineering at the Engine Controls department at Volvo Construction Equipment, Sweden.

We have demonstrated an model-based approach for artifact reuse and product derivation on the UDS. It was shown that it is possible to generate system variants from a overloaded model with little effort.

However, overloaded models become complex and difficult to manage even on smaller subsystems. In our example, the UDS system has only two system types and a small number of components, yet, the overloaded model became rather complex. Due to high complexity of the models, scalability represents an issue as well. In order to scale this approach to the complete engine system it is necessary to separate it into manageable subsystems, however the separation of concerns has proven to be a great challenge.

Further work will be done on the identification and documentation of artifacts that are reused between engine product lines. After that it is necessary to create a asset database from which it will be possible to derive and configure a specific product. Additionally, it is necessary to define guidelines on structuring the design models in order to create a library of reusable design assets where version control is an important aspect to consider. Another interesting aspect is to evaluate modeling tools with respect to the support for variability modeling. The end goal is to exploit commonality between products developed at Volvo CE in order to facilitate reuse, reduce the time-to-market and reduce costs of developing new products.

ACKNOWLEDGMENTS

The research is funded by the Electronic Component Systems for European Leadership Joint Undertaking under grant agreement No. 737494, The Swedish Innovation Agency, Vinnova through the MegaM@Rt2 project, and The Swedish Knowledge foundation through project 20130258 (Volvo Chair). We also thank employees at Volvo CE for their valuable inputs.

REFERENCES

[1] D. Benavides, S. Segura, and A. Ruiz-Cortés. Automated analysis of feature models 20 years later: A literature review. *Information Systems*, 35(6):615 – 636, 2010.

[2] T. Berger, D. Nair, R. Rublack, J. M. Atlee, K. Czarnecki, and A. Wąsowski. Three cases of feature-based variability modeling in industry. In J. Dingel, W. Schulte, I. Ramos, S. Abrahão, and E. Insfran, editors, *Model-Driven Engineering Languages and Systems*, pages 302–319, Cham, 2014. Springer International Publishing.

[3] R. Capilla, J. Bosch, K.-C. Kang, et al. Systems and software variability management. *Concepts Tools and Experiences*, 2013.

[4] R. N. Charette. This car runs on code. *IEEE spectrum*, 46(3):3, 2009.

[5] K. Czarnecki, P. Grünbacher, R. Rabiser, K. Schmid, and A. Wąsowski. Cool features and tough decisions: A comparison of variability modeling approaches. In *Proceedings of the Sixth International Workshop on Variability Modeling of Software-Intensive Systems*, VaMoS '12, pages 173–182, New York, NY, USA, 2012. ACM.

[6] O. Djebbi and C. Salinesi. Criteria for comparing requirements variability modeling notations for product lines. In *Fourth International Workshop on Comparative Evaluation in Requirements Engineering (CERE'06 - RE'06 Workshop)*, pages 20–35, Sept 2006.

[7] J. A. Estefan et al. Survey of model-based systems engineering (mbse) methodologies. *Incose MBSE Focus Group*, 25(8):1–12, 2007.

[8] S. Friedenthal, A. Moore, and R. Steiner. *A practical guide to SysML: the systems modeling language.* Morgan Kaufmann, 2014.

[9] H. G. C. Góngora, M. Ferrogalini, and C. Moreau. How to boost product line engineering with mbse - a case study of a rolling stock product line. In F. Boulanger, D. Krob, G. Morel, and J.-C. Roussel, editors, *Complex Systems Design & Management*, pages 239–256, Cham, 2015. Springer International Publishing.

[10] C. Haskins, K. Forsberg, M. Krueger, D. Walden, and D. Hamelin. Systems engineering handbook. In *INCOSE*,, 2006.

[11] J. Hummell and M. Hause. Model-based product line engineering - enabling product families with variants. In *2015 IEEE Aerospace Conference*, pages 1–8, March 2015.

[12] T. V. Johnson and A. Joshi. Directions in vehicle efficiency and emissions. *Combustion Engines*, 55, 2016.

[13] K. C. Kang, S. G. Cohen, J. A. Hess, W. E. Novak, and A. S. Peterson. Feature-oriented domain analysis (foda) feasibility study. Technical report, Carnegie-Mellon Univ Pittsburgh Pa Software Engineering Inst, 1990.

[14] K. Lee, K. C. Kang, and J. Lee. Concepts and guidelines of feature modeling for product line software engineering. In C. Gacek, editor, *Software Reuse: Methods, Techniques, and Tools*, pages 62–77, Berlin, Heidelberg, 2002. Springer Berlin Heidelberg.

[15] A. Metzger, K. Pohl, P. Heymans, P. Y. Schobbens, and G. Saval. Disambiguating the documentation of variability in software product lines: A separation of concerns, formalization and automated analysis. In *15th IEEE International Requirements Engineering Conference (RE 2007)*, pages 243–253, Oct 2007.

[16] R. Mitschke and M. Eichberg. Supporting the evolution of software product lines. In *ECMDA Traceability Workshop (ECMDA-TW)*, pages 87–96. Citeseer, 2008.

[17] O. Oliinyk, K. Petersen, M. Schoelzke, M. Becker, and S. Schneickert. Structuring automotive product lines and feature models: an exploratory study at opel. *Requirements Engineering*, 22(1):105–135, Mar 2017.

[18] K. Pohl, G. Böckle, and F. J. v. d. Linden. *Software Product Line Engineering: Foundations, Principles and Techniques.* Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2005.

[19] A. L. Ramos, J. V. Ferreira, and J. Barceló. Model-based systems engineering: An emerging approach for modern systems. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 42(1):101–111, Jan 2012.

[20] I. Schaefer, R. Rabiser, D. Clarke, L. Bettini, D. Benavides, G. Botterweck, A. Pathak, S. Trujillo, and K. Villela. Software diversity: state of the art and perspectives. *International Journal on Software Tools for Technology Transfer*, 14(5):477–495, Oct 2012.

[21] K. Schmid, R. Rabiser, and P. Grünbacher. A comparison of decision modeling approaches in product lines. In *Proceedings of the 5th Workshop on Variability Modeling of Software-Intensive Systems*, VaMoS '11, pages 119–126, New York, NY, USA, 2011. ACM.

[22] M. Sinnema and S. Deelstra. Classifying variability modeling techniques. *Information and Software Technology*, 49(7):717 – 739, 2007.

[23] S. Trujillo, J. M. Garate, R. E. Lopez-Herrejon, X. Mendialdua, A. Rosado, A. Egyed, C. W. Krueger, and J. de Sosa. Coping with variability in model-based systems engineering: An experience in green energy. In T. Kühne, B. Selic, M.-P. Gervais, and F. Terrier, editors, *Modelling Foundations and Applications*, pages 293–304, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[24] G. Wang, R. Valerdi, and J. Fortune. Reuse in systems engineering. *IEEE Systems Journal*, 4(3):376–384, Sept 2010.

[25] T. Weilkiens. *Variant modeling with SysML.* Lulu. com, 2012.