Multiple software product lines in automotive software development

Uwe Beher¹, Günter Bönisch¹, Mike Heidrich²

¹ VDO Automotive AG / A company of Continental Corporation, Regensburg
uwe.beher@esg.de, Guenter.Boenisch@continental-corporation.com

² Fraunhofer Einrichtung Systemed er Kommunikationstechnik (ESK), München
mike.heidrich@esk.fraunhofer.de

Abstract: Since today’s well known software product lines (SPL) approaches [At00] [Ba99] [Ka90] [Ka98] [Lö04] [MND02] [Ob00] [Pu08] [WLT99] focus on one single SPL, a methodology for connection and adjustment of multiple product lines is needed. The paper will shortly survey existing processes and methods for product lines and show adaptations to automotive industry. The focus of the paper will be an adapted process for automotive functional development, which is based on multiple software product lines (MSPL) and particularly regards customer-supplier relationship. The paper proposes a SPL interface to manage the software development using MSPL. The interface consists of a SPL Interface Methodology which defines steps for the adjustment of two or more different SPL as well as a data interface (SPL Interface Data) which defines a data format for the exchange between different SPL tools. The SPL adjustment is demonstrated using a case study of an imaginary advanced driver assistance system called mobilSoft Adaptive Cruise Control (MCC), which consists of three product lines, one for the whole MCC and two for the subsystems linear tracking and traverse control.

1 Introduction

In automotive functional development aspects like software reliability and productivity draw more and more attention. As a consequence automotive manufacturers and suppliers address improved software engineering processes. Introducing software product lines (SPL) increases reuse of software elements and supports accomplishing higher quality at less effort. Implementing improved SPL adaptations for automotive industry was one of the goals in the research project mobilSoft [So07]. With the help of latest scientific methods and the experiences of the partners in industry, which were automotive manufacturers and suppliers, solutions for manifold requirements to existing development processes especially for automotive applications were found. Among these requirements were short development timelines, less development effort and high quality for each single software product. However, a more and more important challenge for development processes is the increasing variability and complexity of the final products. Optimized SPL can help automotive manufactures and suppliers to meet this challenge while fulfilling existing requirements to their development processes.
Four steps were stated as necessary for Automotive OEMs or suppliers to obtain optimized SPL and to get to a functional demonstrator, which verifies the feasibility of the planned adaptations for the SPL. The first step was an analysis at automotive manufactures and suppliers which lead to global requirements and characteristics for automotive SPL and supporting processes. The second step was the evaluation of existing methods and tools, which define the technical state of the art, with the requirements found. The third step was evolving a specification of an automotive specific SPL, because existing approaches for SPL were not particularly made for automotive application. The last step was the setup of a demonstrator, where the appropriate processes and methods were verified. Aim of this last step is not only proving the concept of an adapted process, but also testing acceptance of comprehensive tool landscapes which are often combined with the introduction of integrated approaches. Among these four steps the analysis of existing methods in the academic environment and available tools on the market turned out to be an appropriate basis for introducing SPL improvements. In order to adapt existing tools to fit into existing processes, systematic and purposive investigations and determination of efforts are necessary. Especially introducing new methods in existing complex development structures is a critical task.

Additionally, existing methods and processes were investigated for coverage of the interface between automotive OEMs and suppliers. Because of missing approaches for this important task in automotive development, additional methods were evolved which fit into existing SPL as an extension of the process framework. The paper will have a main focus on this aspect and will propose an SPL Interface Methodology for adjustment of two or more different SPL and SPL Interface Data as data interface during parallel development in different organizations. The result of applying these interfaces is a connection of single SPL to a Multiple Software Product Line (MSPL).

2 State-of-the-art software product lines

2.1 Existing SPL approaches

The arrangement of software development as a software product line is an effective method for increasing reuse in software development and well known in branches with high software share.

<table>
<thead>
<tr>
<th>Short</th>
<th>Name</th>
<th>Short description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FODA</td>
<td>Feature Oriented Domain Analysis</td>
<td>Domain analysis method based on feature trees [Ka90]</td>
</tr>
<tr>
<td>FORM</td>
<td>Feature Oriented Reuse Method</td>
<td>Extension of FODA for domain design [Ka98]</td>
</tr>
</tbody>
</table>
It was initially applied e.g. in telecommunication, medical equipment or consumer products, but recently automotive industry shows increasing interest as well. In table 1, a short overview of existing SPL approaches is given. The listed SPL approaches mainly differ in the focus on software engineering tasks. The application of one of these approaches as whole or partial as individual interpretation of the SPL methodology is, in some instances, realized at automotive manufacturers or suppliers. However, the adaptation of the given approaches to the needs of automotive product development remains being a very complex task and only few parts of the listed approaches fit in the special automotive environment. Thus the following chapter proposes characteristics needed for a SPL to be suitable as an (automotive) MSPL.

### 2.2 Characteristics of an automotive SPL

Regarding [Bö04] the SPL is divided into the process areas domain engineering and application engineering. Domain engineering leads to the setup of the software product line and the design of the assets, application engineering deploys the assets of the software product line in order to generate a final product. According to the generative approach [CE00] the automotive SPL is divided into requirements space and solution space. This results in the following alignments of the SPL:

- Domain analysis is the requirements space of the domain engineering

<table>
<thead>
<tr>
<th>COPA</th>
<th>Component Oriented Platform Architecting Method</th>
<th>Approach for setup of SPL based on architecture and component assets [Ob00]</th>
</tr>
</thead>
<tbody>
<tr>
<td>FAST</td>
<td>Family-Oriented Abstraction Specification Translation</td>
<td>Total SPL framework [WLT99]</td>
</tr>
<tr>
<td>QADA</td>
<td>Quality Driven Architecture Design</td>
<td>Quality oriented software design [MND02]</td>
</tr>
<tr>
<td>PuLSE</td>
<td>Product Line Software Engineering</td>
<td>Generic and comprehensive framework for SPL [Ba99]</td>
</tr>
<tr>
<td>KobrA</td>
<td>Komponentenbasierte Anwendungsentwicklung</td>
<td>Component based software development, application of PuLSE [At00]</td>
</tr>
<tr>
<td>EAST-ADL</td>
<td>EAST - Architecture Description Language</td>
<td>Usage of variants in automotive software design [Lö04], [Vo04]</td>
</tr>
<tr>
<td>pure::variants</td>
<td>pure::variants</td>
<td>SPL modeling tool [Pu08]</td>
</tr>
</tbody>
</table>

Table 1: Overview of SPL approaches
• Domain design is the solution space for the domain engineering
• Application analysis is the requirements space of the application engineering
• Application design is the solution space for the application engineering

A common method for organizing the requirements space is the usage of feature models for variant requirements, which include features and the relations between features. Generally a feature model e.g. FODA represents all requirements which lead to functional properties in the product line.

• A Feature describes the requested product properties from the point of view of a stakeholder, which in the context of a SPL is a customer, developer, manager, investor or supplier. A feature can be derived from single requirements or a set of aggregated requirements.

• The feature relation describes the relationship between one or more features, which is mandatory, optional or excluding.

The selections of features in the feature model during application engineering builds up single products of solution space.

The solution space contains a solution model for the domain and the resulting solutions by specific selections in the solution model. The solution selection itself is the result of decisions made along the solution path. The solution model consists of selected assets and their relationship. An asset is the smallest unit of a solution model and can’t be divided into smaller units. Assets can be categorized into three types.

• A basic asset is a solution, which can be used in many products without any change.

• A customized asset is a basic asset, which can be used in many products and which includes complete calibration possibilities.

• A specific asset is a solution for only one product, which is initially introduced into the product line and which aims at utilization for further products.

For example an innovation is a specific asset that is implemented in a product for the first time and which, in the case of a successful introduction in the market, is planned to be extended to other products as customized or basic asset. Aligning the design of an asset with an existing product line eases the implementation of the project specific solution into the SPL.
The feature-asset relation expresses the relationship between an asset and a feature. Because of a feature being a set of one or more requirements, an assignment between requirements and solution is achieved. The FA Linker (feature asset linker) includes the relationship between chosen features and related assets and thus regards only the relevant subset from the total set of all feature asset relations. For the interface between SPL, which is described in the following chapter, the FA linker plays a crucial role.

With the characteristics and main tasks proposed for SPL software engineering, an appropriate fundament is formed for the specification of an automotive SPL. This leads to connecting different SPL to a Multiple Software Product Line (MSPL), which is also an automotive requirement.

3 Connection of software product lines

3.1 Fundamentals of connecting product lines

The automotive final product “vehicle” of the OEM is not the result of only one integrated product line. It rather consists of different subsystems with their own product lines and additional single solutions e.g. innovations, which are initially implemented only in few car models. Additionally, each subsystem may be delivered from its own organization, which may be internal or an external supplier. As a result the structure of each product line and the stakeholders mapped to it may differ.

---

![Fig. 1: Example: Structure of an OEM SPL](image1)

![Fig. 2: Example: Structure of a supplier SPL](image2)
Examples for possible product structures with assets at OEM and supplier are depicted in figure 1 and figure 2. Figure 1 shows an OEM with its three platforms “Upper”, “Middle” and “Compact”. The “Upper” platform itself consists of four assets. “U/M/C” is used in “Upper”, “Middle” and “Compact” platform, “U/M” in “Upper” and “Middle” platform, “U/C” in “Upper” and “Compact” platform and finally “U” only in “Upper”, which is a single solution. The structure of the supplier in figure 2 is similar in methodology but different for the solution. Its platform “OEM1” consists of “A/B”, a common asset for its platform “OEM1” and “OEM2”, and a single solution “A”. It is used in the “Upper” platform of the OEM in figure 1, but not restricted to it. The supplier has actually no influence on the OEM decision, to put it into “U/M/C”, “U/M”, “U/C” or “U”. And finally, supplier may sell a similar product to another OEM, which will not be regarded in the SPL of the first OEM. As a consequence, even if products from the supplier are used in the platform of the OEM, the configuration of assets which build up the solution space is independent.

Furthermore a simple one-to-one match in the solution space where a product “A/B”+”A” may fit into “upper class” vehicles is hardly applicable for automotive industry. In general, separate SPL exist at different levels of functional hierarchy and depend on their own field of activity. In this case, all individual SPL need to be synchronized in order to deliver the correct product at the right time.

![Fig. 3: Connected software product lines](image)

After synchronization, the setup being composed of connected SPL is stated as a Multiple Software Product Line (MSPL). Figure 3 shows an example for the structure of connected SPL. The interfaces in this example occur between SPL internally in one company or externally in many companies. For a functional connection of SPL it is crucial, to align each single SPL in a correct hierarchical order.

A layered architecture e.g. EAST-ADL [Lö04] is able to structure SPL hierarchically. In the given example, “SPL A” at supplier may be at the highest level “user” as interface to the customer. But also internal product lines “SPL C” at “cluster” level or “SPL D” at “platform” level need correct alignment to facilitate “SPL A” providing the requested product.
3.2 Interfaces of multiple software product lines

As stated in the last chapter an important prerequisite for a MSPL is the synchronization of each SPL to each other in order to align development of the SPL assets. This paper proposes two main tasks for synchronization:

- Exchange of requirements and design data via a standardized data model “SPL Interface Data”
- Adjustment of SPL specific tasks via a general methodology “SPL Interface Methodology”

The SPL Interface Data generally contains design data like:

- Product requirements
- Features of the product as set of requirements
- Architecture patterns as reference solutions

The data exchange for synchronization mainly takes part at an early stage of the development process in a SPL. An appropriate phase is after domain analysis, where each single SPL defines its own reuse concept and asset structure for the product development. Before reaching domain design phase, inputs from all other SPL complete own basic product requirements by aligning own product application to the requirements of the total product. The SPL interface based on the data models and synchronization tasks needs to cope with a contradiction. On the one hand it shall be as flexible as needed to react on changed constraints after domain design, which at this point typically come from functional implementation. On the other hand the SPL interface shall serve as a consistent backbone for development data. Figure 4 illustrates the role of SPL Interface Data and SPL Interface Methodology in the context of single SPL implemented into the total product development.

Fig. 4: Concept of the SPL interface
The part SPL development represents the individual SPL process of an organization like a company or a profit centre in a company. The product development covers the development of the final product as the result of the combination of the individual SPL processes. The SPL interface in between consist of two main elements, the central data container SPL Interface Data and the SPL Interface Methodology in order to synchronize the individual SPL processes. The SPL Interface Data serve as exchange mechanism for design data. It consists of:

- an architecture mapping
- an SPL IF Allocation List
- the data container itself

The reason for an architecture mapping is founded in the different SPL approaches, where the connection into the total product architecture is not provided. The architecture mapping verifies the architecture patterns of the single SPL and assigns each component to its correct place in the product, thus a common understanding of the product is established throughout all SPL.

The allocation list synchronizes the product features and assets and supports the compliance of the requirements to a consistent product design. It deals as configuration support and is a main document for the total product development.

The data container is the physical container for design data for combined SPL development which are mainly requirements and assets. It is recommended to structure the container in terms of connected SPL, this eases each SPL updating their own content during change management.

The SPL Interface Methodology describes the process for connecting, adapting and aligning two or more SPL. It provides initialization and synchronization, which realize a strong connection of the individual development processes to the final product. The SPL Interface Data and the SPL Interface Methodology will be explained in more detail in the following chapters.

### 3.3 Architecture mapping and initialization

A key element of a SPL is a reference architecture, where single products are derived from. In order to connect software product lines, the elements of the reference architectures have to match together to generate a functional total architecture. Reference architectures may be EAST-ADL [Lö04] or Car-DL [Wi06]. The common base of most reference architectures is a layered system model, where abstract system elements at a higher level are divided into smaller elements at a lower level which provide more details of the internal structure.
The architecture mapping coordinates the different reference architectures at initialization of SPL alignment. It contains methods for the combination of requirements and solution spaces in the correct level of abstraction. Provided, that each individual SPL has a reference architecture, which fits into the general architecture frame for the architecture mapping, the architecture mapping merges all SPL into one central architecture of the MSPL. As a consequence of all reference architectures being abstract and generic, no SPL has to open its internal detailed architecture to the central architecture of the MSPL for the final product. That is e.g. strategic product plans and customer structures, which are often modeled in internal architectures, stay closed to other competing SPL.

![Alignment of different reference architectures at initialization](image)

Figure 5: Alignment of different reference architectures at initialization

Figure 5 shows the result of architecture mapping of individual SPL. In this figure, three possible SPL are drawn as bordered shapes. The main objective of the architecture mapping in the SPL Interface Data is to assign each element of the single SPL to the correct layer of the reference architecture and to requirements or solution space. By overlaying all SPL valid intersections of all SPL can be determined. The dark grey area in the middle of all shapes is the common part of all SPL, which needs a common problem and solution descriptions. Otherwise areas without any interaction have no significance for the SPL Interface Methodology.

3.4 Synchronization and allocation

In a MSPL individual SPL exchange data mainly during synchronization phase. The data base for the data exchanged is the SPL IF container. Content of the container are requirements, features as sets of requirements, assets, relations of container elements and additional documents, which support version control and asset history. Synchronization is separated into two steps, filling and data adjustment. During filling phase, all interface data of the other product lines are collected and structured. Basis of the structure is the architecture mapping described previously, which is elaborated during initialization.
As a result, a comprehensive configuration for all features, assets and relations is found, which is valid for realizing the total product. This configuration is a global SPL IF allocation list which contains the relationship between all features and assets involved in the product development.

Figure 6 shows the structure of the allocation list, if two SPL “SPL A” and “SPL B” are connected. Only if the allocation list as a combination of both SPL contains only viable product solutions, data adjustment phase is finished. Product application design and domain design follow after the adjustment phase. In case of changes in single SPL are required due to the result of a review, the overall SPL adjustment is triggered again.

4 Demonstrator for Multiple Software Product Lines

4.1 Main task of the demonstrator

The demonstrator supports the verification of an adapted SPL regarding implementation into a MSPL. A second effect, which development organizations should not neglect, is the acceptance test for introducing integrated software tool landscapes which comes along with the SPL adaptation. An option for a MSPL demonstration is building up the individual SPL as hardware independent function, which eases realization of the demonstration. The software platform to be used may be a standardized platform like AUTOSAR, but AUTOSAR is not a prerequisite for the demonstrator. The concept of the stakeholder need for performing the interface for the SPL does not reflect existing personnel but describes abstract roles and tasks that need to be performed. The mapping of the requested roles to an organizational chart has to be found for each organization individually.
4.2 Results of connection of software product lines

Figure 7 shows the setup of the function mobilSoft Cruise Control (MCC), which is composed of three product lines: the overall MCC function, linear tracking for MCC function and the traverse control for MCC function. In this case study tracking and traverse control are SPL of two automotive suppliers and the total MCC function a SPL at one OEM. The modeling of all feature models for the problem and solution space was realized with pure::variants [Pu08]. Architecture mapping was accomplished within the research project mobilSoft [So07] by the use of Car-DL [Wi06] as a reference architecture. However the shown approach can be used with other architectures, e.g. EAST-ADL [Lö04] as well. The appropriate files for the SPL IF container and the SPL IF allocation list were realized as *.xml-documents according to XMS scheme definitions (XMD) as output of UML based data models. Finally an iterative refinement of the MSPL approach was accomplished.

![Fig. 7: mobilSoft Cruise Control as a MSPL](image)

5 Conclusion

This paper has provided the structure of SPL Interface Methodology and SPL Interface Data in order to adjust many individual SPL into one Multiple Software Product Line. The result is a global adjustment list for requirements and solution space of all involved SPL. The adjustment list is the base of product application in each SPL and is an additional specification at the early stage of a product development process. In automotive applications, after implementing additional requirements emerging from the adjustment list, existing SPL process for product application shall be able to perform SPL product development. The extension of the MSPL has no necessity for special SPL tools and shall be ready to be integrated in most tool supported SPL.
References