Chapter 5. Supplier Independent Feature Modelling

This chapter is published as:

Abstract
The growing use of Software Supply Chains results in an increasing proportion of the functionality of a software product line (SPL) being determined by functionality of suppliers. In order to cover the whole product line, it is sometimes necessary to use several suppliers, offering partly the same functionality. This leads to overlapping feature models.

This paper introduces a Supplier Independent Feature Model (SIFM). Through dependency relations between the SIFM and the feature models of the individual suppliers, the variability of the combined components is modelled. In this way the complexity of merging feature models is avoided and the relations with the development artefacts are maintained.

The creation of an SIFM is elaborated through an example and a case study. These show that this is a straightforward process, and show that the SIFM facilitates the generation of product variants in an efficient manner.

5.1 Introduction

In software product line engineering an essential step identifies the commonalities and manages the variability [Clements 2001]. Feature modelling is a means to describe the variability in a product. Feature modelling originates from the FODA method [Kang 1990] and has subsequently been extended [Czarniecki 2000]. It is used for requirements engineering and for automated product configuration [Czarniecki 2000], i.e. creating an SPL variant.

The process of creating a product line and creating individual products is usually separated into two phases, i.e. domain engineering and application engineering [Czarniecki 2000, Pohl 2005A, Czarniecki 2004B]. During domain engineering, the commonality and variability of a software product line are identified and captured using a variability model, e.g. a feature model, and the artefacts that make up the product line are developed. During application engineering a SPL variant is created through a selection or configuration of features.

In many industries Software Supply Chains have emerged to feed the demand for components [Greenfield 2004]. Within a product line, the selected Commercial off the Shelf (COTS) components have to support the variability of the features in the product line that

The supplier selection may result in a situation in which the requirements and variability of the product line cannot be fulfilled by a single supplier. Several suppliers have to be selected that, together, fulfil the set of requirements for a product line. When deriving a member of the product line, a choice is made between the alternative components from the different suppliers. By alternative components we mean that only one of the (set) of components can be used in a particular implementation.

**Example 1:** A car manufacturer uses different Integrated Circuit (IC) Suppliers for alternative car-infotainment systems with different variability. Figure 23 shows a subset of their feature models in FODA notation [Kang 1990, Czarniecki2000].

![Feature Model](image)

**Figure 23 Car-infotainment suppliers**

The ICs from SupplierA and SupplierB represent alternative components since it is not possible to combine functionality from both suppliers. So, for a particular configuration, one supplier has to be chosen that satisfies the requirements best. This is usually achieved through an iterative process, in which the principal features are selected first, then the most appropriate supplier, and finally the remaining features provided by that supplier.

**The challenge** is to construct a feature model that integrates the variability of the components from the alternative suppliers into the overall feature model. The resulting feature model should describe the variability of the entire product line, i.e. including the required features and variability, and should be constrained by those provided by the suppliers. It should also contain the mapping from the features to the development artefacts. Furthermore, the feature model should provide a structure that supports the selection process of features and suppliers during application engineering.

**Overview:** In Section 5.2 we will give some background and in Section 5.3 we will discuss alternative approaches. In Section 5.4 the Supplier Independent Feature Modelling (SIFM) is introduced. Section 5.5 gives an example from an existing Software Supply Chain, followed by tool experiences. We will conclude with a discussion, a comparison with related art and our conclusions.
5.2 Background

5.2.1 Feature modelling and tool support

Weiss et al. [Weiss 1999] describe that the purpose of application engineering is to explore the space of requirements and to generate the application very quickly. Botterweck et al. [Botterweck 2008] observe that a product line can easily incorporate thousands of variation points and visualization techniques are needed to a productivity gain of SPL by making the application engineering process as efficient as possible. Czarnecki et al. [Czarnecki2000] describe that “A hierarchical decomposition for the primary structure of a feature configuration is important for reasons of understand-ability”. The tree structure of a feature diagram supports the selection process, e.g. a child feature only needs to be considered if its parent has been selected.

Feature modelling is supported by commercial variability management tools [Debbi 2007]. These tools are used during domain engineering to create the feature model and to relate the features to the variation points in the product line architecture. These tools are used during application engineering to generate the product variant and they offer validation and consistency checking.

5.2.2 Supply chains and COTS selection

Wallnau et al. [Wallnau 2002] describe a growing influence of software supply chains and a change in the practice of component-based software design. In the past, a software product lines was developed, based on specifications from the stakeholders’ needs. Nowadays, more and more software is being developed from commercially available components. Consequently, a trade-off has to be made between the stakeholders’ needs and the desire to leverage components from the marketplace [Albert2002].

COVAR [Pohl 2005A,Pohl2001] describes a process to select Software Suppliers and COTS components for Software Product Line Engineering. This is positioned during domain engineering. COVAR introduces provided variability to describe the variability of the COTS components, which can be evaluated against the required variability of the product line.

5.3 Integrating Feature Models

The overall feature model of a product line contains several functional areas, e.g. a car contains engine management, chassis, infotainment, etc. With the adoption of Supply Chains, many of these functional areas are implemented by components from external suppliers. Therefore, a downstream participant in the supply chain uses a set of components from (several) suppliers and integrates them into a product, possibly combined with its own components [Czarnecki2005A, Hartmann 2008, see Chapter 4].

The feature models are integrated at two levels. First, for each functional area, the feature models of the alternative suppliers are combined. Subsequently, these are integrated into the overall feature model for the complete product, as shown in Figure 24. The second step may
also add dependencies between the integrated feature models that arise from the product line’s requirements and technical dependencies.

5.3.1 Overlapping feature models

In an earlier paper [Hartmann 2008, see Chapter 4] we described that merging feature models is especially difficult in cases where there are overlapping features, i.e. features that represent alternative implementations.

Example: Before illustrating the approach with a case study, we will introduce the following example. We have 6 features: F1, F2... F6 and 3 different suppliers: Sup1, Sup2 and Sup3, which offer largely the same functionality (see Figure 25). The features F1..F6 of each supplier represent the same functionality and we will call them corresponding features. In
an actual situation the different suppliers might use different names for corresponding features.

- \( F_1 \) is an optional feature, supported by all suppliers, but it is a mandatory feature for Sup2.
- \( F_2 \) is a mandatory feature supported by all suppliers
- \( F_3 \) is an optional feature, only supported by Sup2
- \( F_4 \) is an optional feature, not supported by Sup2
- \( F_5 \) and \( F_6 \) are optional children of \( F_2 \). For Sup1 they are alternatives. For Sup2 and Sup3 they are optional children of \( F_2 \).
- For Sup3: \( F_5 \) requires \( F_1 \)

The following sub-sections describe two possible approaches for integrating overlapping feature models and the limitations of each approach. Section 5.4 will describe a new approach that overcomes the limitations.

### 5.3.2 Possible approach 1: Alternative subtrees

A basic way to integrate feature models from suppliers is to create subtrees which represent the alternative feature sets from the suppliers, see Figure 26.

To configure the feature model to generate an SPL variant, first a supplier has to be selected and only then the child features that belong to that supplier can be selected. The drawback of this approach is that it is not easy to compare the feature sets that are offered by these suppliers, especially not when the number of features becomes large.

![Alternative subtrees](image)

**Figure 26 Alternative subtrees**

However, in order to generate a product variant, a comparison is required to select the most suitable supplier for a product. This means that a part of the supplier selection, which was already done during domain engineering, has to be redone during application engineering. This is too time-consuming.
We therefore conclude that this approach does not provide a structure to effectively support the feature selection process during application engineering.

5.3.3 Possible approach 2: Merging

Another approach is to merge the feature models from the alternative suppliers by creating a set of features which is the superset of the features from the different suppliers. We will call these *Supplier Independent Features* and, in our example, there would be six of these. A variation point is added to select between the alternative suppliers. This variation point is modelled as a mandatory feature with alternative child features, as shown in Figure 27.

By using constraints between the Suppliers variation point and the merged features, the differences between the suppliers can be captured.

We would then get: Sup2 \textless\textless excludes\textgreater\textgreater F4, Sup1 and Sup3 \textless\textless excludes\textgreater\textgreater F3 and Sup2 \textless\textless requires\textgreater\textgreater F1. We also need to describe that, for Sup1, F5 and F6 are alternative features, and for Sup3, F5 requires F1. This is illustrated in Figure 27, which only shows two of these constraints.

This approach has the advantage of permitting the selection of features first and, because of the dependency relations, the possible suppliers are determined. For instance, the selection of feature F3 will result in only Sup2 being suitable, and consequently F1 becomes a mandatory feature. Using the notation of [Czarniecki2005B], we can represent the reasoning as:

- \textit{excludes} F3 \implies (\neg\text{Sup3} \land \neg\text{Sup1}) \land
- \textit{xor-group} (\neg\text{Sup3} \land \neg\text{Sup1}) \implies \text{Sup2} \land
- \textit{requires} \text{Sup2} \implies F1.

The approach also permits the selection of a supplier first, which then gives a specialized version of the supplier independent feature tree. However, this approach has two disadvantages.

- The possible relations inside each of the suppliers feature models have to be modelled as dependency relations between the supplier variation point and the merged features. For instance, F5 and F6 are alternative features for Sup1, but
optional child features of $F1$ for the other suppliers. This leads to complex
dependency relations between the supplier variation point and the supplier
independent feature tree.

- The mapping between the feature models and the development artefacts is not kept
intact because the feature models from the suppliers are replaced by a merged
feature model. Czarnecki et al. argue that features in a feature model are merely
symbols. The mapping of features to development artefacts, such as components,
models and user documentation, gives the feature models semantics [Czarnecki2004B, Czarnecki2005B]. Pohl et al. [Pohl 2005A] describe how a
development artefact can be related to one or more variants and that a variant must
be related to at least one development artefact. The preservation of the mappings
between the feature model and the artefacts is essential for automatic application
derivation from the configuration of the feature model. The lack of traceability
means that the selection and configuration of supplier-independent features is not
sufficient to generate an SPL variant.

5.4 Supplier Independent Feature modelling

5.4.1 Composition of models

To address the limitations described above, we retain the feature model from each supplier
and create a separate feature model that composes them. This will be called the Supplier
Independent Feature Model (SIFM). The SIFM contains the super-set of the features from
all the suppliers, together with a variation point to distinguish between the suppliers.
Through dependency relations, the SIFM is linked to the Supplier Specific Feature Models
(SSFM) and consequently, linked with their development artefacts.

This composition, which we will call Composite Supplier Feature Model (CSFM), models
the variability of the alternative components.

5.4.2 Example

The SIFM, see Figure 28, consists of the Supplier Independent Features and a variation point
to describe the alternative Suppliers, $Sup1$, $Sup2$ and $Sup3$. This is similar to that described
in Section 5.3.3.

However, we will not replace the feature models from the separate suppliers but use
dependency relations between the features of the SIFM and the SSFMs to capture the
differences between the suppliers and to maintain the relations to their artefacts. Unlike the
dependency relations required in the merging approach in Section 5.3.3, the dependency
relations between the SIFM and SSFMs are now uniform and independent of the dependency
relations within the SSFMs. For reasons of clarity we have only shown a sub-set of the
dependencies in Figure 28.
The dependency relations are as follows. First we need to capture that a Supplier Independent Feature requires a Supplier Specific Feature and vice versa.

\[ F_1 \text{\textless requires\textgreater} S_1.F_1 \text{ XOR } S_2.F_1 \text{ XOR } S_3.F_1, \quad F_2 \text{\textless requires\textgreater} S_1.F_2 \text{ XOR } S_2.F_2 \text{ XOR } S_3.F_2, \quad F_3 \text{\textless requires\textgreater} S_2.F_3 \]

(Not for S1 and S3, since F3 is not supported by Sup1 and Sup3)

\[ F_4 \text{\textless requires\textgreater} S_1.F_4 \text{ XOR } S_3.F_4, \]

(Not for S2, because F4 is not supported by Sup2)

\[ F_5 \text{\textless requires\textgreater} S_1.F_5 \text{ XOR } S_2.F_5 \text{ XOR } S_3.F_5, \]

\[ F_6 \text{\textless requires\textgreater} S_1.F_6 \text{ XOR } S_2.F_6 \text{ XOR } S_3.F_6 \]

S1.F1\textless requires\textgreater F1, S2.F1\textless requires\textgreater F1, S3.F1\textless requires\textgreater F1

S1.F2\textless requires\textgreater F2, S2.F2\textless requires\textgreater F2, S3.F2\textless requires\textgreater F2

S2.F3\textless requires\textgreater F3 S1.F4\textless requires\textgreater F4, S3.F4\textless requires\textgreater F4

S1.F5\textless requires\textgreater F5, S2.F5\textless requires\textgreater F5, S3.F5\textless requires\textgreater F5

S1.F6\textless requires\textgreater F6, S2.F6\textless requires\textgreater F6, S3.F6\textless requires\textgreater F6

We also need to express that the alternative suppliers require the corresponding supplier’s features, and vice versa so:

\[ \text{Sup1\textless requires\textgreater S1, Sup2\textless requires\textgreater S2, Sup3\textless requires\textgreater S3,} \]

\[ \text{S1\textless requires\textgreater Sup1, S2\textless requires\textgreater Sup2, S3\textless requires\textgreater Sup3} \]

Note 1: As described, the dependency relations between the SIFM and the SSFM are independent from the internal relations within the SSFMs, e.g. \( S1.F5 \) and \( S1.F6 \) are
alternative features for Sup1, but this does not need to be expressed in the dependency relations.

Note 2: rather than merging the SSFMs into the SIFM’s supplier sub-tree, the SSFMs have been modelled as separate feature models, with dependencies with the SIFM. The split clarifies which features are explicitly selected by the application engineer, i.e. those of the SIFM. It also has practical benefits when maintaining the feature models, as described in Section 5.6.1.

5.4.3 Feature and supplier selection during application engineering

The SIFM gives the application engineer the possibility of starting by either selecting features, selecting the supplier, or a combination of both. We will show two scenarios using the example.

Scenario 1: Primary selection of features.
Selection of F3 results in Sup2 becoming selected, since only Sup2 supports F3. Consequently F2 becomes a mandatory feature and F4 is not available anymore.

\[
\begin{align*}
\text{requires} & \quad F3 \Rightarrow S2.F3 \\
\text{child-parent} & \quad S2.F3 \Rightarrow S2 \\
\text{requires} & \quad S2 \Rightarrow Sup2 \\
\text{xor-group} & \quad Sup2 \Rightarrow (\neg Sup1 \land \neg Sup3) \\
\text{requires} & \quad \neg Sup1 \Rightarrow \neg S1 \\
\text{parent-child} & \quad \neg S1 \Rightarrow \neg S1.F4 \\
\text{requires} & \quad \neg Sup3 \Rightarrow \neg S3 \\
\text{parent-child} & \quad \neg S3 \Rightarrow \neg S3.F4 \\
\text{requires} & \quad \neg S1.F4 \land \neg S3.F4 \Rightarrow \neg F4
\end{align*}
\]

Scenario 2: Primary selection of supplier.
Selection of Sup1 results in F3 not being available any more.

\[
\begin{align*}
\text{xor-group} & \quad Sup1 \Rightarrow (\neg Sup2 \land \neg Sup3) \\
\text{requires} & \quad \neg Sup2 \Rightarrow \neg S2 \\
\text{parent-child} & \quad \neg S2 \Rightarrow \neg S2.F3 \\
\text{requires} & \quad \neg S2.F3 \Rightarrow \neg F3
\end{align*}
\]

Selection of F5 results in F6 not being available anymore, because F5 and F6 are alternative features for Sup1.

\[
\begin{align*}
\text{requires} & \quad F5 \Rightarrow S1.F5 \\
\text{xor-group} & \quad S1.F5 \Rightarrow \neg S1.F6 \\
\text{parent-child} & \quad \neg S2 \Rightarrow \neg S2.F6 \\
\text{requires} & \quad \neg Sup3 \Rightarrow \neg S3
\end{align*}
\]
Chapter 5

parent-child

\[ \neg S3 \Rightarrow \neg S3.F6 \land \]

requires

\[ (\neg S1.F6 \land \neg S2.F6 \land \neg S3.F6) \Rightarrow \neg F6 \]

The selection of features of the SIFM can be used to fully configure a product. Since it contains a complete set of the features, the features in the SSFMs do not also need to be considered. Hence, this structure provides an efficient means to support feature selection and configuration during application engineering.

The selection of features might easily lead to conflicting situations, which are difficult to resolve and requires studying the SSFMs. The solution proposed in [White 2008] provides a powerful approach for automatically identifying the source of such conflicts.

5.4.4 Resolving ambiguous supplier selection

The selection of features may result in situations in which more than one supplier can still be used. This means that the Supplier variation point is still undetermined. We propose to resolve this situation by using the preferred supplier for the remaining available suppliers. This can be achieved by introducing prioritized alternative features in a feature model. The prioritized preferred suppliers could be determined at COTS component selection, during domain engineering, by considering the lowest cost, highest quality, or any combination of non-functional requirements.

5.4.5 Structure of the SIFM and dependency relations with SSFMs

In the example, we have seen that F5 and F6 are child features of F2 in the SIFM. This is only possible because this was also the case for all the suppliers. If, for instance, F6 had been a child feature of F1 for Sup3, then F6 had to be modelled as an optional feature, directly below the root feature. In general, the more commonality there is in the structure of the SSFMs, the more it can be “copied” to the SIFM. Since a tree-like structure enhances the usability of a feature model during application engineering [Czarniecki2000], it is worthwhile to apply this structure to the SIFM.

To identify common structural elements, both common parent-child relations and grouping of features might be used. If one feature requires another, it may be treated as a parent-child relation and features that are mutually exclusive might be treated as alternatives [Czarniecki2007]. However this also depends on other relations of the involved features.

The dependency relations between the SIFM and SSFMs do not change when the structure of the SIFM is changed. These relations are independent of the internal structure of the SIFM and the SSFMs. These relations only depend on the availability of a feature. In our example, this was the availability of F3 and F4 for different suppliers.

5.4.6 Generic description and work process

The generic description of the CSFM is a generalization of the example.

In order to create a CSFM and subsequently the overall feature model, the following work process is applicable.

**Step 1:** Identify the correspondence between features from different suppliers.

**Step 2:** Create a sub-tree to describe the supplier variability.
Step 3: Create a complete list of the corresponding features, which will become the leaves of the Supplier Independent Feature Model.

Step 4: To create a tree-like structure, use the structure which is common in the referenced SSFMs, e.g. parent child relations and grouped features.

Step 5: Add the dependency relations between the corresponding features in the SIFM and the SSFMs and the relations between the suppliers and their SSFMs. This combination forms the CSFM.

Step 6: Integrate the CSFM, covering a particular functional area, into the overall feature model.

Step 7: Add the constraints that arise from the product lines requirements and technical constraints to the SIFM.

5.4.7 Corresponding features

There are situations where it is difficult to define corresponding features. Two features might look similar, but do not represent identical functionality. For instance, two suppliers might offer JPEG enhancement (JE). However, these might give different image results and therefore cannot be called corresponding features. We will define them as JPEG1, which belongs to Supplier1 (Sup1), and JPEG2, which belongs to Supplier2 (Sup2). In this case a Supplier Independent feature can be defined “JE” with two alternative child features JPEG1 and JPEG2. Now the dependency relations are JE<<requires>>Sup1.JPEG1 XOR Sup2.JPEG2, just as if they were corresponding features. We also have to add: JPEG1<<requires>>Sup1.JPEG1, JPEG2<<requires>>Sup2.JPEG, Sup1.JPEG1<<requires>>JPEG1 and Sup2.JPEG2<<requires>>JPEG2. This gives the possibility to select JE alone, in cases where the type of enhancement is not important. The alternative choice of JPEG1 and JPEG2 might then be forced by the selection of other features that determine the supplier. Alternatively, an additional selection of one of the child features can be done to choose the type of enhancement explicitly.

Another situation is where one feature defines a subset of the functionality of another feature or is specified more precisely. For instance, Audio compression versus MP3 and WMA. In this case, the finest level of granularity of the features has to be used as the Supplier Independent features, to allow a precise representation and explicit selection. Features with a coarser level of granularity have to be analyzed to determine their functionality. A similar approach to that of JPEG enhancement might be used to allow both explicit and implicit selection. A combination of both granularity and quality situations is also possible.

Comparing features in general can be difficult, as this is a problem that is pervasive in COTS selection.

5.4.8 Separate components

Until now, we have described a situation in which the components from different suppliers cannot be combined, i.e. they are alternatives. One can imagine that a subset of the development artefacts of a supplier can be separated. For instance, in our example, we could imagine that, for feature F4, supplier Sup3 has a separate component that implements the feature and fits within the architecture. Now, this component can be treated as independent
from Sup3. Only a constraint with feature $S1.F4$ is required to prevent these implementations from being combined. This separation is possible at every level of granularity. In the description of the ZigBee case, in Section 5.5, we will show an example where different suppliers are used for different layers of the software architecture.

There may also be situations in which several components that relate to similar features may be part of the product configuration but may not be available together at run-time. In this case these components may be treated as independent as well but an extra restriction is needed to prevent them from being instantiated together at run-time.

5.5 ZigBee Case Study

5.5.1 Functionality and products

ZigBee is an industry standard for wireless communication over short distances. In contrast to Bluetooth and Wifi, ZigBee focuses on low-power applications, such as the monitoring and control of devices for industrial control, medical data collection and home automation, see [Zigbee 2009].

5.5.2 Architecture and variation points

A ZigBee device consists of hardware and software. The hardware contains small microprocessors, memory, a RF transceiver and optional sensors. For the software, there is an OSI reference architecture that consists of 6 layers, as shown in Figure 29.

Figure 29 ZigBee reference architecture.

The specifications for the Physical (PHY) and MAC layers are defined by IEEE 802.15.4-2006. This specification contains a set of conformance tables that describe the mandatory and optional features in relation to different device types and describes dependencies between features. The Network (NWK) and Application (APL) layer are defined by the ZigBee alliance [Zigbee 2009]. This alliance has defined a conformance table for the NWK layer. This table defines different device types with their role in the network and also includes mandatory and optional features with dependencies.

The following gives some examples of optional features. For the PHY and MAC layers there is an optional Beaconing feature and there are differences in device types. For the
NWK and APL layers there are optional features that involve the creation and separation of networks and the role of a device in a network, i.e. co-ordinator, router or end-device.

Because the software relates to an industry standard that contains variation points, there is naturally a common interpretation of most of the features. However, suppliers provide other features that are outside the scope of the standard. Some examples of such features are the possibility to use external non-volatile memory and dynamic power saving.

5.5.3 Supply chain

The supply chain consists of IC vendors, software vendors and OEMs that create the final product. In the supply chain we found multiple parties for each link, partly offering the same functionality. In the example of this paper we take the position of an IC vendor that considers using software from specialized suppliers. In this chain, an IC vendor may offer hardware only, or may deliver hardware with software that for the PHY and MAC layers, or which may also include the NWK layers.

For the software, we investigated three different suppliers. We found that some of the suppliers only contained the PHY and MAC layers and others also included the NWK and APL layers. We analyzed the available features using a similar approach to that described in [Pohl 2005A].

Table 11 below gives a subset of the features of the suppliers, which we will identify as A, B and C. In the table Man means Mandatory, Opt means Optional and No means not supported.

<table>
<thead>
<tr>
<th>Feature \ Suppliers</th>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>NWK layer</td>
<td>No</td>
<td>Opt</td>
<td>Opt</td>
</tr>
<tr>
<td>Beaconing</td>
<td>Opt</td>
<td>No</td>
<td>Opt</td>
</tr>
<tr>
<td>GTS</td>
<td>Opt</td>
<td>No</td>
<td>Opt</td>
</tr>
<tr>
<td>Security</td>
<td>No</td>
<td>Opt</td>
<td>Opt</td>
</tr>
<tr>
<td>Mesh/Cluster Config</td>
<td>No</td>
<td>Man</td>
<td>No</td>
</tr>
<tr>
<td>Outside Conformance</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Power saving</td>
<td>No</td>
<td>No</td>
<td>Opt</td>
</tr>
<tr>
<td>Message fragmentation</td>
<td>No</td>
<td>No</td>
<td>Man</td>
</tr>
<tr>
<td>…..</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Giving these differences, an IC-vendor can offer a wider product portfolio by using multiple suppliers. For instance, the power saving feature together with the beaconing feature gives the possibility to obtain battery powered nodes with a lifetime of 10 years. This is required in advanced e- metering infrastructure [Kistler 2008A] and, consequently, Supplier C is the only suitable supplier for this. Supplier B offers more capabilities in Security and Network configuration, which makes it more suitable for commercial buildings with strict
Chapter 5

authentication and authorization rules [Kistler 2008B]. Supplier A is an open source implementation, with a very small set of features and without a NWK layer. This supplier is most suited to form the basis for an in-house implementation that can be used for implementing ZigBee End Devices, which require the minimum set of functionality and the least amount of memory and, consequently, the lowest cost.

5.5.4 Creation of the SIFM

The creation of a SIFM was started by first modelling the features listed in the conformance tables. These features could be identified easily in the implementations from the different suppliers because very similar names were used. This made it easy to define the corresponding features and to find the variability in the implementations.

Subsequently, we added the features outside the conformance tables. For these features, a careful comparison of their specifications was required to ascertain the full set of features and their correspondence across suppliers.

5.5.5 Different suppliers for different layers

It is feasible to combine implementations from different suppliers. For instance, a different supplier could be chosen for the MAC and PHY layers from that selected for the NWK and APL layers. These different supplier choices are represented by different variation points and, therefore, in this case, there are two variation points, selecting the MAC/PHY supplier and the NWK/APL supplier respectively.

5.6 Tool support

5.6.1 Experiences

The CSFM approach was tested using a commercial variability management tool [Purevariants 2009]. We were able to generate the CSFM, as described in this paper. Some observations from this are listed below:

1. The SSFMs where modelled as separate projects within the Eclipse environment. The CSFM was created using a separate SIFM feature model, with the SSFMs as “referenced” projects. This allowed us to maintain the SSFMs and related family models as separate entities. When changes were made to the development artefacts or internal relations of the SSFMs, no changes need to be made to the CSFM. Furthermore, the SSFMs could be referenced by multiple projects for different customers.

2. Instead of using the “requires” relation from the SSFMs to the SIFM, e.g. S1.F1<<requires>>F1, we used F1<<requiredFor>>S1.F1 etc. In this way we only needed to add relations to the SIFM and not to the SSFMs, which might have been obtained from suppliers.

3. During selection of features, the auto-resolve functionality of the variability management tool made it possible to immediately see the remaining choices after
a feature selection, thus preventing an invalid selection being made. For instance, after a selection of \textit{Sup1} and \textit{F5}, \textit{F6} was not selectable anymore.

4. The \textit{auto-resolve} functionality of the variability management tool did not show changes in the relations between features. For instance, when \textit{Sup1} was selected, \textit{F5} and \textit{F6} could have been visualised in the SIFM as alternative children of \textit{F2}, because of the relations with the SSFM of \textit{Sup1}. However, this was not supported. Additional visualisation [Botterweck 2008] might further increase the efficiency of the selection of features, caused by complex dependency relations. We will leave this for further research.

5.6.2 Automation

The creation of a CSFM could be further automated. When the corresponding features are defined and the SIFM has been constructed, the creation of the dependencies between the SIFM and the SSFMs could be done automatically.

The creation of the SIFM may be automated by merging the SSFMs as described in [Segura 2007]. However the approach described in that paper assumes that features have the same name and the same parental relations. To be able to use more general structure of the SSFMs, as described in Section 5.4.5, a form of refactoring, as described in [Alves2006], is needed. Further research on these topics is needed.

5.7 Discussion and Alternatives

In this section we will describe how our approach can be used with the Orthogonal Variability Model, how it can be used during COTS selection and how it fits into staged configuration.

5.7.1 Application with the Orthogonal Variability Model (OVM)

The approach described in this paper can be applied to the OVM [Pohl 2005A] model in a similar fashion. A Supplier Independent OVM model can be constructed that consists of a variation point describing the alternative suppliers and a set of Supplier Independent variation points with dependency relations to the related variation points of the suppliers. Since the OVM model lacks the tree like structure commonly used in feature modelling, step 4 of the work process (see Section 5.4.6) can be omitted.

5.7.2 Using the CSFM during COTS selection

The selection process with more than one alternative component to serve the product line variability differs from the selection of a single component with variability [Pohl 2005A, Pohl2001]. Now the required functionality and variability has to be compared with the combined functionality and variability of the alternative suppliers. The CSFM can facilitate the selection of suppliers. By creating a CSFM, the variability of a combination of suppliers is described. This can be used to compare the required variability with the provided
variability. With the CSFM, the same approach can be used as with the selection of one supplier.

Because the CSFM describes the superset of available features, not all these features are available together. This makes the selection non-trivial. However, the same problem also occurs when one has to deal with only one supplier in which one feature excludes another.

5.7.3 Integration with context variability and staged configurations

In an earlier paper [Hartmann 2008, See Chapter 4] we introduced Context Variability to model multiple product lines. This approach can be combined with the CSFM. The Context Variability Model then specializes the SIFM in a similar fashion as the Context Variability model specializes a conventional feature model to create a feature model for multiple product lines. We have seen in some cases that the context determines the choice for the supplier. For instance, that a geographic region dictates standards that are only fulfilled by one of the suppliers. An example is an Electronic Program Guide for TVs and hard-disk-recorders, which are usually developed for a particular broadcasting standard.

The CSFM can be combined with Staged Configuration. A specialized CSFM can be created for the next participant in the supply chain. This could either be a selection of the supplier, a selection of features, or both. When a feature model that contains multiple suppliers is transferred to a customer, a number of problems might occur. E.g. it might not be desirable, for commercial reasons, that the customer is aware of the different suppliers.

5.8 Related Work

The integration of variability from alternative components has not been discussed earlier in literature. Sugera et al. [Segura 2007] described a method for merging feature models. They assume that the merged features relate to the same artefacts. So their solution is not applicable for alternative components.

Reiser and Weber introduce Multi-level Feature Trees [Reiser 2006]. A reference feature model is used as a template and guideline for referring models. Their approach aims to deal with separate organizational units that are responsible for a part of the overall feature model. Since their solution does not incorporate the relation to different artefacts, it does not provide a solution for alternative components.

5.9 Conclusions

Building a product line using alternative suppliers for similar functionality makes it possible to serve a wider group of customers and avoids a dependency on a single supplier.

In this paper we demonstrated that merging feature models is not the most suitable approach when dealing with alternative components since it would sever the link to the development artefacts. We have described an approach in which an additional feature model
is created, the Supplier Independent Feature Model. With this model we have avoided the disadvantages of merging feature models.

A particular advantage of our approach is that the original feature models have been kept “intact”, meaning that no changes have to be made to SSFMs. This means that the relation between the feature models and the artefacts haven’t changed. Consequently, any changes made to the development artefacts do not influence the created Supplier Independent Feature Model.

The ZigBee case demonstrated the need for using multiple suppliers and showed that creating the SIFM is easier in cases where there is a standard to which the suppliers adhere and the reference architecture facilitates the use of different suppliers for different layers. The experiences with a commercially available variability management tool showed that the method is straightforward to apply.