Inconsistency Resolution of Feature Trees within Multi-level Feature Modeling

“A new approach for ensuring consistency in Feature Modeling”

Erman Şen
Department of Computer Engineering
Middle East Technical University
Ankara, Turkey
erman.sen@ceng.metu.edu.tr

Seyyit Alper SERT
Department of Computer Engineering
Middle East Technical University
Ankara, Turkey
alper.sert@ceng.metu.edu.tr

Ali Hikmet Doğru
Department of Computer Engineering
Middle East Technical University
Ankara, Turkey
dogru@ceng.metu.edu.tr

Abstract—The study is focused on a new approach for ensuring the consistency of feature trees among a multi-level feature model hierarchy. Although a single global feature tree sounds well in theory, practical cases showed that multiple feature trees, each of which standing for a specific feature of any sub-product line among a product family, worked better. Especially, dealing with domains consisting of high complexity product lines makes it inevitable to work within a multi-level framework. Meanwhile, utilizing multi-level feature modeling brings another dilemma for “possible deviations” on to the stage due to changes applied to feature models over a long period.

The focus of this paper is to present a new approach for aligning feature models across these possible deviations, hence ensuring consistency among the overall modeling framework. We provide the basics of multi-level feature trees concept first. We then elaborate possible deviations on feature trees together with their conformity issue that may become cumbersome, inflexible and unmanageable over a long time. Finally, we propose a solution for dealing with the inconsistency between referring feature trees and their respective reference feature trees.

Index Terms—Multi-level feature model, reference feature tree, referring feature tree, model deviations, consistency check, alignment of feature models.

I. INTRODUCTION

The aerospace industry is one of the manufacturing environments with high-complexity. Many independent groups and departments collaborate to accomplish their dedicated jobs. However, the integration among thousands of components of an aircraft is vital. Especially in such intricate organizational contexts, feature modeling has great potential: features may serve as a link between management, marketing and development in a single company and as a link between companies to ease communication; and they can provide a central view of variability in a wide range of development artifacts - such as requirements, design models or test cases - thus becoming the core of all variability and evolution management [4]. On the other hand, involving feature models with such a huge complexity becomes an issue in terms of flexibility and management.

An effective approach is already presented to handle the management of large feature modeling framework in [3]. With this approach, the aim is to avoid a global large variability modeling by introducing multi-level feature modeling. Nevertheless, the implication of that approach also brings the concept of “deviations” on feature models as it moves on adding derivative features apart from those defined on their respective reference feature models. Hence, the feature models become very much alike from their respective reference models over a long time. Ensuring consistency among the whole modeling structure then needs the alignment of those deviations.

We will start in Section I with presenting basic definitions together with some domain information – the aircraft manufacturing. Then, we will mention possible deviations in Section II. Thereafter, we will explore our contributions by defining the “alignment process” to be applied on deviated feature trees. Finally, we will mention about some related and future work.

II. BACKGROUND

In this section we will briefly present the basis for feature models and then introduce the “reference feature model” followed by an extension called “referring feature model”. Afterwards, Multi-level feature model will be defined.
A. Feature Models

“Product line” stands for a set of products similar in nature or functionality while having some different specifications. Although they have common characteristics, much variability is also in place. Figure 1 stands for a sample product family for aircraft manufacturer. In modern product oriented approach, the individual products are no longer developed independently from one another, instead only a single, but variable product is developed. From this variable product, the actual products (also called product instances) can then be derived by configuration [3]. At this stage, feature modeling is utilized. Feature models are well suited and convenient for domain analysis of large and complex systems. It is a widely used technique in Software Product Line development. Feature models allow stakeholders to describe domain concepts in terms of commonalities and differences within a family of software systems [2]. They are used in defining the set of common characteristics among different products as well as setting up the restrictions to which extend deviations are allowed. From this view point a “feature” is an individual specification each product instance may or may not have. Further documentation about feature models can be found in [4] and [7].

![Figure 1. Sample aircraft product family](image1)

B. Reference Feature Models

Within highly-complex domains, it is helpful to establish templates and guidelines as being a reference for some others. To support the domain analysis process, traditional feature models are extended such that some feature models are regarded as a basis – or “reference feature model” – for some other feature models. The reference model then serves as a template and guideline for any referring model by defining default features together with their default properties and by defining which deviations from these defaults are allowed [3]. Any reference feature model can in turn may refer to another reference feature model in a higher level in the hierarchy, hence becomes a “referring feature model”.

![Figure 2. Example of reference and referring feature models](image2)

C. Multi-Level Feature Modeling

Constructing reference model schema leads to a hierarchically structured tree of reference feature models for a specific domain. Each node in the product line tree of the domain would then be represented by a feature model. Such a tree of reference/referring feature models is called a multilevel feature tree.
D. Extension to Reference Feature Modeling

Reference feature models studied in a manner that each referring feature model can only have one reference feature model at a time [3]. However, an extension to this approach is also possible proposing the usage of more than one reference feature model for a specific referring feature model. The aim is to bring flexibility in managing the variability in referring feature models. Consider the following feature model below. Feature “wiper” has two reference feature “windscreen” and “rearwindow” meaning that it inherits features from both. Modeling frameworks that allow multiple reference models is beyond the scope of this article and left as future work. Our contribution is based on the single-reference model.

![Figure 4. Sample feature tree with shared features](image)

![Figure 5. Sample multi-reference feature model](image)

III. DEVIATIONS

Let $R$ be the reference feature model and $F$ be the referring feature model. Furthermore, let $f_R$ be the feature in $F$ that refers to feature $f_R$ in $R$. It is assumed that both $R$ and $F$ are identical initially.

As additions and deletions are made on both $R$ and $F$ in terms of $f_R$ over time, both models will become “modified” and be different from each other. Any “differences” between $R$ and $F$ then is called “deviation”. Taking the specifications into consideration like feature constraints, attributes, ranges, cardinalities, etc., any $R$ and $F$ might be in one of the following states:

- $F$ is identical to $R$, hence they conform to each other
- $F$ deviates from $R$ but still conforms to it.
- $F$ does not conform to $R$ at all.

![Figure 6. Possible states](image)

Although the last state seems that it is out of the boundary of “reference vs. referring” philosophy, it is worthwhile to be included in practice for much flexibility and ease of management of the whole feature framework. In fact, our proposed approach is mainly focused on this type of deviations.

At a point of time, there might be different number of features in both $R$ and $F$. As a general rule of thumb, we can classify these deviations into two distinct groups. Below is a brief description of this classification.

**Case I:** either there is a feature in $R$ for which no referring feature is defined in $F$ because the feature in $R$ was added; or the corresponding feature in $F$ was removed.

**Case II:** either there is a feature in $F$ for which no reference feature is defined in $R$ because the feature in $F$ was added; or the corresponding feature in $R$ was removed.
The next section introduces the approach for aligning these deviations for ensuring the consistency between $R$ and $F$.

IV. ALIGNMENT OF DEVIATED FEATURE MODELS

Feature models having the same reference feature models but with possible deviations may stand for different product instances. The deviations among these referring feature models may make these product instances so different from each other in a manner that, even their feature trees may become so different from their common reference feature model. In an effort for keeping the modeling framework consistent, it is needed to align reference models by eliminating the deviations as much as possible.

One possible way of resolving this is to find the intersection of all referring feature models and then use that intersection as a new reference feature model.

Another approach would be creating new reference model(s) other than the existing one and redesign the relations between referring and reference models such that, there might be more than one reference model rather than having a single one for any referring feature model.

In this paper, we will present the first solution and leave the second as a future work.

V. INTERSECTION OF FEATURE TREES

A. Alternative Methods

In literature, there are few different algorithms for constructing intersection of feature trees. In [8], for instance, the disadvantage is that it does support feature models having common features as well as those that are not parent-compatible (i.e. equal feature has equal parents). The main drawback in this approach is the duplication of features. In [10] however, it does not take into account the case where the intersection of sets of children can be empty. Furthermore, this approach does not consider cross-tree constraints. Apart from these two, in [6], it is proposed translating feature models into propositional formulas, and the conjunction of two such formulas represent the intersection of the corresponding feature models. Their method to retrieve this intersection from its propositional formula however requires user interactivity.

In this paper, we used the approach in [5] in which “requires” and “excludes” constraints are taken into account and does not suffer from the drawbacks and disadvantages mentioned above. Following sub-sections describes this algorithm.

B. Terminology

Prior to describing the algorithm, some definitions and conventions are listed below:

- $FE(FM)$ is the set of features.
- $RO(FM)$ is the root feature.
- $\forall f \in FE(FM), PA(FM, f)$ is the parent feature of “$f$” in the tree structure of $FE(FM)$ whose root is $RO(FM)$.
- $\forall f \in FE(FM), PC(FM, f)$ is a set whose elements are sets of features. Each element of $PC(FM, f)$ is a possible set of children of $f$.
- A set $RE(FM)$ of ordered pairs of features; these are the "requires" constraints.
- A set $EX(FM)$ of pairs of features; these are the "excludes" constraints.
- The semantics of a feature model is the set of all its products [8]; and the semantics of FM is denoted by $SEM(FM)$. The feature model whose set of features is empty is denoted by NIL; it has no products. When $SEM(FM_1) = SEM(FM_2)$ then $FM_1$ and $FM_2$ are said to be equivalent, denoted by $FM_1 \sim FM_2$.

C. Algorithm

The pre-requisite for the algorithm is that both feature trees must be parent-compatible. That is, each equal feature has equal parent or both are the root. Let $FM_1$ and $FM_2$ be the two feature models to be intersected. Let $T_1$ be the set of products whose features belong to $FE(FM_1)$ and satisfy “the tree constraints” of $FM_1$. Let $C_1$ be the set of products whose features belong to $FE(FM_1)$ and satisfy “the non-tree or cross-tree constraints” of $FM_1$. Likewise we define the sets $T_2$ and $C_2$. The algorithm is, in fact, simply the solution of Equation 1.

$$(T_1 \cap C_1) \cap (T_2 \cap C_2) = (T_1 \cap T_2) \cap (C_1 \cap C_2) \quad (1)$$

By applying the associative and commutative rules of the intersection operator, it yields;

$$(T_1 \cap C_1) \cap (T_2 \cap C_2) = (T_1 \cap T_2) \cap (C_1 \cap C_2) = (T_1 \cap T_2) \cap (C_1 \cap C_2) \quad (2)$$

where $(T_1 \cap C_1)$ is the set of products which satisfy all constraints of $FM_1$ i.e. $SEM(FM_1)$. Hence;

$$(T_1 \cap C_1) = SEM(FM_1) \quad (3)$$

$$(T_2 \cap C_2) = SEM(FM_2) \quad (4)$$

$$(SEM(FM_1) \cap SEM(FM_2)) = (T_1 \cap T_2) \cap (C_1 \cap C_2) \quad (5)$$

The left-hand side of the Equation 5 is thus the set of products of the “intersection” – i.e. the output of the
algorithm. The set \((T_1 \cap T_2)\) is the set of products which are common to FM\(_1\) and FM\(_2\) when cross-tree constraints are not considered. Furthermore, the set \((C_1 \cap C_2)\) is the set of products which are common to FM\(_1\) and FM\(_2\) when cross-tree constraints are considered. The algorithm therefore consists of two phases:

- Phase I: constructing the intersection of FM\(_1\) and FM\(_2\) without cross-tree constraints, leading to a feature model with semantics \((T_1 \cap T_2)\), and
- Phase II: incorporating the cross-tree constraints, such that the semantics \((T_1 \cap T_2)\) gets intersected with \(C_1\) and \(C_2\).

**Phase I**

Assume that FM\(_3\) is a feature model with semantics \((T_1 \cap T_2)\). Then:

\[
FE(FM_3) = FE(FM_1) \cap FE(FM_2) 
\]

(6)

That means FM\(_3\) consists of the features which are common to FM\(_1\) and FM\(_2\). The tree structure of FM\(_3\) is inherited from FM\(_1\) and FM\(_2\), \(\forall f \in FE(FM_3)\) which is not the root of FM\(_1\):

\[
PA(FM_3, f) = PA(FM_1, f) = PA(FM_2, f) 
\]

(7)

\[
PC(FM_3, f) = PC(FM_1, f) \cap PC(FM_2, f) 
\]

(8)

The products of FM\(_3\) are those products which satisfy the tree constraints of both FM\(_1\) and FM\(_2\), so SEM(FM\(_3\)) = \((T_1 \cap T_2)\), as intended.

If \(PC(FM_3, f) = \{\{\}\}\) then \(f\) becomes a leaf. However, if \(PC(FM_3, f)\) is empty, there are no legal possibilities for the children of \(f\). This means that \(f\) is a dead feature, i.e. there are no products containing \(f\). We may therefore apply a semantics preserving refactoring of FM\(_3\) by:

- eliminating \(f\) from FM\(_3\)
- discarding all \(f\)'s successors from FM\(_3\)
- removing all sets containing \(f\) from PC(FM\(_3\), PA(\(f\))).

Note that if \(f\) is a mandatory child of PA(\(f\)), then PC(FM\(_3\), PA(\(f\))) becomes empty, and PA(\(f\)) must be eliminated as well.

**Phase II**

In this phase, FM\(_3\) is to be modified within the constraints defined in the scope of both FM\(_1\) and FM\(_2\). In other words all products that disagree with semantics \((C_1 \cap C_2)\) have to be removed from FM\(_3\). Therefore, it is needed to apply all constraints from FM\(_1\) and FM\(_2\) to FM\(_3\). Here is the pseudo code:

```plaintext
for (each constraint (A,B) in [RE or EX])
{
    /* if all features still in FE(FM_3)
    if A ∈ FE(FM_3) & B ∈ FE(FM_3)
    { /* then just keep them
    keep (A,B) in RE(FM_3) or EX(FM_3)
    } else
    {
    /* if the required feature missing, then remove
    the caller feature and its successors
    if ( (A,B) ∈ RE or EX(FM_3)) &
    (A ∈ FE(FM_3) & B ∉ FE(FM_3))
    { eliminate A and its successors
    } }
}
```

(9)

**Example**

Suppose that we have initially a reference feature model as indicated below. In addition, suppose that we have two referring feature models FM\(_1\) and FM\(_2\) which show some variants that occurred during the development cycle.

![Initial reference feature model](image)

Let FM\(_4\) be the feature model with the following tree and cross-tree constraints “\(D\) requires \(H\)” and “\(F\) requires \(E\)”.

![Initial reference feature model](image)
Let \( \text{FM}_2 \) be the feature model with the following tree and cross-tree constraints “\( I \) requires \( G \)”, “\( J \) excludes \( P \)” and “\( E \) excludes \( F \)”.

Both \( \text{FM}_1 \) and \( \text{FM}_2 \) are parent-compatible, hence the algorithm is applicable for their intersection \( \text{FM}_3 \).

**Phase I:**

\[
\begin{align*}
\text{FE}(\text{FM}_1) &= \{\text{Root, A,B,C,D,E,F,G,H,I,J}\} \\
\text{FE}(\text{FM}_2) &= \{\text{Root, A,B,C,D,E,P,K,G,H,I,J}\} \\
\therefore \text{FE}(\text{FM}_3) &= \text{FE}(\text{FM}_1) \cap \text{FE}(\text{FM}_2) \\
&= \{\text{Root,A,B,C,D,E,G,H,I,J}\}
\end{align*}
\]

So \( \forall f \in \text{FE}(\text{FM}_3) \) i.e. \( \forall f \in \{\text{Root, A,B,C,D,E,G,H,I,J}\} \) we should compute \( \text{PC}(\text{FM}_3, f) \) where;

\[
\text{PC}(\text{FM}_3, f) = \text{PC}(\text{FM}_1, f) \cap \text{PC}(\text{FM}_2, f)
\]

Since \( \text{PC}(\text{FM}_3, \text{B}) = \{\}\), the feature “\( B \)” must be removed from the set \( \text{FE}(\text{FM}_3) \), together with its children features \( \{G,H\} \). Hence, the updated \( \text{FE}(\text{FM}_3) = \{\text{Root, A,C,D,E,I,J}\} \). Furthermore, sets containing “\( B \)” are removed from \( \text{PC}(\text{FM}_3, \text{Root}) \), which becomes \( \{\},\{A\},\{C\},\{AC\} \}. In addition \( \forall f \in \{D,E,I,J\} \) “\( f \)” is a leaf since \( \text{PC}(\text{FM}_3, f) = \{\}\).

The resulting feature model has a graphical notation, which can be determined with a simple algorithm, given in [9]. The result is:

**Phase II:**

Regarding all constraints within \( \text{FM}_1 \) and \( \text{FM}_2 \); constraints “\( D \) requires \( H \)” and “\( I \) requires \( G \)” imply that features \( D \) and \( I \) have to be removed from the intersection model, since features \( G \) and \( H \) are not present anymore. Hence, \( \text{FE}(\text{FM}_3) = \{\text{Root, A,C,E,I,J}\} \). Sets containing \( D \) and \( I \) should be removed from \( \text{PC}(\text{FM}_3, A) \) and \( \text{PC}(\text{FM}_3, C) \) respectively, which yields \( \text{PC}(\text{FM}_3,A)=\{E\} \) and \( \text{PC}(\text{FM}_3,C)=\{J\} \). The final configuration is:

Constraints “\( J \) excludes \( P \)” and “\( F \) requires \( E \)” can be neglected since \( P \) and \( F \) are already eliminated. Finally the constraint “\( E \) excludes \( J \)” must be added to \( \text{EX}(\text{FM}_3) \). The
final intersected feature model – hence the new reference feature model is therefore:

![Figure 12. The new reference feature model](image)

Thereafter, the new reference feature model is now to be replaced with the existing one, as it is now equi-distant to both referring feature models in terms of their intersection trees which in turn ensures consistency between the reference model and its referring models.

VI. CONCLUSION AND FUTURE WORK

Utilizing multi-level feature modeling framework in high-complexity manufacturing environments brings a major problem in terms of “deviations between the reference feature model and it’s referring feature models” that needs to be aligned for ensuring the inconsistency within them that might occur over a long time.

In this paper, we proposed a new approach for eliminating the inconsistency between reference feature models and their respective referring feature models. The approach simply tries to capture the commonality of referring feature models by applying the “intersection of feature models resolution technique” and then replacing the existing reference feature model with the resulting feature model which will stand for the “new reference feature model”.

The approach we used in this paper uses the intersection algorithm detailed in [5]. However, there are other alternatives as studied in [6], [8] and [10]. For future work, it is worthwhile to work with these alternative algorithms and compare yielded different reference feature models.

We believe that the proposed approach contributes to the management of highly-complex product lines within the multi-level modeling framework, as it revises the reference model as needed hence increases the flexibility as well as ensures the consistency.

REFERENCES