Merging Feature Models

Pim van den Broek, Ismênia Galvão
Department of Computer Science
University of Twente
Enschede, The Netherlands
{pimvdb, i.galvao}@ewi.utwente.nl

Joost Noppen
Computing Department
University of Lancaster
Lancaster LA1 4WA, United Kingdom
j.noppen@comp.lancs.ac.uk

Abstract— **In this paper, we consider the problem of merging feature models which consist of trees with "requires" and "excludes" constraints. For any two such feature models which are parent-compatible, their merge is defined to be the smallest parent-compatible feature model which has all products of the constituents. We show how the specification can be implemented, present a number of properties of the merged feature models and give some examples.**

Keywords- feature models, merging

I. Introduction

Feature models are used to specify the variability of software product lines [1,2]. Recently, several authors have proposed methods to merge feature models [3,4,5,6], or stressed the importance of such methods [7,8,9,10]. The operation of merging several feature models into a single feature model can be used to build a software product line from existing software products or to extend a software product line with new products. However, a specification of the merging of feature models has up till now only been given for feature models based on directed acyclic graphs [6]. In this paper we consider feature models which are based on trees, since these are the feature models that are most widely used in practice. We will consider feature models whose feature diagrams are trees, and which have as cross-tree constraints only the usual "requires" and "excludes" constraints. This implies that we have to deal with three difficulties.

The first difficulty is the following. Suppose feature A has parent feature B in one feature model, and parent feature C in another feature model. In a merged feature model, A cannot have both B and C as parent features, as sharing is not allowed. Therefore we will only define merging for feature models which are parent-compatible; when two feature models are parent-compatible there are no equal features with different parents..

The second difficulty is that feature modeling languages come with different graphical notations [6]. Since we want our specification to be applicable to all feature languages which have no sharing and have requires and excludes constraints, we abstract from a particular modeling language, by adopting a formalism that is compatible to all of them. However, for readability we adopt in our examples a graphical notation as much as possible.

The third difficulty is that it is not possible to require that a merged feature model has exactly the products of the constituent feature models. We require that a merged feature model has at least the products of the constituent feature models, but allow that it has additional products. Consider the following example:

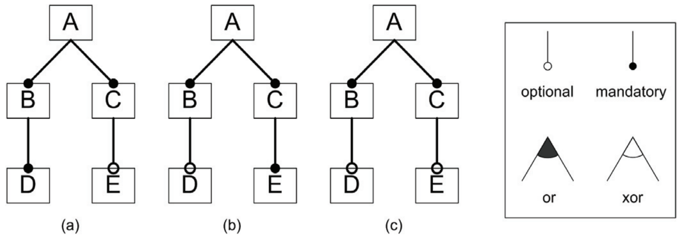

Figure 1. Example of merging of feature models

Here, we consider (c) to be the merge of (a) and (b); it has a product ABC which is not a product of either (a) or (b). A feature model without sharing, with only requires and excludes constraints, whose products are just the products of (a) and (b), and which is parent-compatible with (a) and (b), does not exist.

Therefore, our specification of merging feature models is as follows: if feature models FM1 and FM2 are parent-compatible, then their merge FM3 is the unique (up to equivalence [6]) feature model which:

1. Is parent-compatible with both FM1 and FM2.
2. Has all products of FM1 and FM2.
3. Is the "smallest" feature model which satisfies 1 and 2, i.e. if FM4 is a feature model which also satisfies 1 and 2 then the products of FM3 are also products of FM4.

In the remainder of this paper we will show how the merging of two parent-compatible feature models can be performed, and prove that the constructed merged feature model indeed satisfies the property of being the "smallest" parent-compatible feature model which has all products of both constituent feature models.

In the next section we describe the feature models we consider in this paper. In section 3 we define the merging of feature models. In section 4 we discuss the implementation of the merge operation. In section 5 we present some examples. In section 6 a number of properties of merged feature models are given. Section 7 contains the related work and section 8 concludes the paper.

II. FEATURE MODELS

Features in this paper are considered to be atomic entities. A feature model FM in this paper consists of:

1. A set of features FE(FM); if it is non-empty, then it contains a designated element called the root feature, denoted by RO(FM).
2. A tree structure on FE(FM) whose root is RO(FM); for each feature F ≠ RO(FM) its parent is denoted by PA(FM,F).
3. For each feature F of FE(FM) a set PC(FM,F) whose elements are sets of features. Elements of this set are the sets of features which are considered to be possible sets of children of F.
4. A set RE(FM) of ordered pairs of features; these are the "requires" constraints.
5. A set EX(FM) of pairs of features; these are the "excludes" constraints.

We will take the liberty to omit the first argument of FE, RO, PA, PC, RE and EX in case where it is clear from the context to which feature model they refer.

A nonempty subset P of FE(FM) is a product of FM if and only if the following four conditions are satisfied:

1. $\forall F \in P$: $F \neq RO \Rightarrow PA(F) \in P$.
2. $\forall F \in P$: $\{G \in P \mid PA(G) = F\} \in PC(F)$.
3. $\forall (A,B) \in RE$: $A \in P \Rightarrow B \in P$.
4. $\forall (A,B) \in EX$: $A \in P \Rightarrow B \notin P$.

The semantics of a feature model is the set of all its products [6]; we denote the semantics of FM by SEM(FM). The feature model whose set of features is empty, which has been introduced in [10], is denoted by NIL; it has no products. When SEM(FM1) = SEM(FM2) then FM1 and FM2 are said to be equivalent, denoted by FM1 ~ FM2.

Feature model languages normally use a graphical notation or annotations to express PC. Our specification will be independent of the notational conventions of feature modelling languages. In this paper we adopt the convention that features are denoted by capitals, and products by strings. An example of a feature model is given in Fig 2.

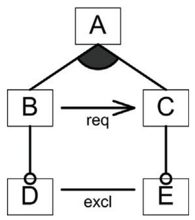

Figure 2. Example feature model

Here FE = {A,B,C,D,E}, RO = A, PA(B) = A, PA(C) = A, PA(D) = B, PA(E) = C, PC(A) = {{B},{B,C},{C}}, PC(B) = {{},{D}}, PC(C) = {{},{E}}, PC(D) = {{}}, PC(E) = {{}}, RE = {(B,C)} and EX = {(D,E)}. The semantics of this feature model is {ABC, ABCD, ABCE, AC, ACE}.

III. MERGING OF FEATURE MODELS

Given two feature models, FM1 and FM2, we want to define the feature model FM3 = merge(FM1,FM2), which is the merging of FM1 and FM2. This definition should only depend on the SEM(FM1) and SEM(FM2). This means that if FM1 and/or FM2 are replaced by equivalent feature models then SEM(FM3) does not change.

In order to be mergeable with each other, FM1 and FM2 have to be parent-compatible. Feature models FM1 and FM2 are parent-compatible if and only if

1. If neither FM1 = NIL or FM2 = NIL then RO(FM1) = RO(FM2).
2. $\forall F \in FE(FM1) \cap FE((FM2)$: If $F \neq RO(FM1)$ then PA(FM1,F) = PA(FM2,F).

The rationale of this precondition is that in FM3 the feature must have a single parent, and we have no means to choose between the parent features from FM1 and FM2 if these are different. The same precondition is assumed by Segura et al. [3]. Chen et al. [4] and Acher et al. [5] do not impose this condition, but without it their method fails to recognise all common features of FM1 and FM2, thereby producing feature trees with multiple occurrences of features. Schobbens et al. [6] do not need the precondition, as their feature models allow sharing.

We start the definition of FM3 with the definition of FE(FM3). It consists of those features from FE(FM1) and FE(FM2) which occur in products of FM1 resp. FM2. Note that we do not simply take the union of FE(FM1) and FE(FM2); this is done to guarantee that the result only depends on SEM(FM1) and SEM(FM2).

The tree structure of FM3 is defined as follows:

- If FM1 ≠ NIL then RO(FM3) = RO(FM1).
- If FM2 ≠ NIL then RO(FM3) = RO(FM2).
- $\forall F \in FE(FM3)$: if $F \in FE(FM1)$ and $F \neq RO(FM1)$ then PA(FM3,F) = ΠA(FM1,F).
- $\forall F \in FE(FM3)$: if $F \in FE(FM2)$ and $F \neq RO(FM2)$ then PA(FM3,F) = PA(FM2,F).

This is unambiguous, due to the requirement that FM1 and FM2 are parent-compatible.

The next property of FM3 we will define is PC, but first we need the concept of actual sets of children. To each feature model we associate the property AC, such that AC(FM,F) consists of those sets of child features of feature F which occur as actual set of child features in some product of FM. For each feature model FM and each feature $F \in FE(FM)$ we have $AC(FM,F) \subseteq PC(FM,F)$. When these two sets are not equal this means that some sets of children, although possible according to PC, do not occur as such in any product due to the external constraints. We define:

$$\forall F \in FE(FM3): PC(FM3,F) = AC(FM1,F) \cup AC(FM2,F)$$

Here we do not simply take the union of PC(FM1,F) and PC(FM2,F) since, as before, we require that the result only depends on SEM(FM1) and SEM(FM2). Using AC instead of PC assures that this is the case.

The final step is the specification of the external constraints. Informally, we want the feature A of FM3 to require(exclude) the feature B of FM3 if and only if A requires(excludes) B both in FM1 and in FM2. To formalise this, we start with the definition of some relations on FE(FM3). For any feature model FM, let REQ(FM) be the relation on FE(FM3) defined by (A,B) $\in$ REQ(FM) $\Leftrightarrow \forall$ P $\in$ FM: A $\in$ P $\Rightarrow$ B $\in$ P. Similarly, EXCL(FM) is the relation on FE(FM3) defined by (A,B) $\in$ EXCL(FM) $\Leftrightarrow \forall$ P $\in$ FM: A $\in$ P $\Rightarrow$ B $\notin$ P. We want to define RE(FM3) and EX(FM3) in such a way that the following equalities hold:

REQ(FM3) = REQ(FM1) $\cap$ REQ(FM2)
EXCL(FM3) = EXCL(FM1) $\cap$ EXCL(FM2)

To achieve this, we define RE(FM3) and EX(FM3) by simply taking all external constraints which are compatible with this requirement:

RE(FM3) = REQ(FM1) $\cap$ REQ(FM2)
EX(FM3) = EXCL(FM1) $\cap$ EXCL(FM2)

It is obvious that with this definition we have REQ(FM3) $\supseteq$ REQ(FM1) $\cap$ REQ(FM2) and EXCL(FM3) $\supseteq$ EXCL(FM1) $\cap$ EXCL(FM2. In section 6 we will prove that indeed the equal sign holds here.

This definition of RE(FM3) and EX(FM3) only depends on SEM(FM1) and SEM(FM2). Of course, the sets RE(FM3) and EX(FM3), as defined above, are in general larger than necessary (for instance, RE(FM3) contains all pairs (F,PA(FM3,F)); elements may be deleted from these sets as long as this does not change SEM(FM3).

This concludes the definition of the merging of two feature models. In the next section, we consider a number of issues which arise when one wants to compute a merged feature model.

IV. Computation of a Merged Feature Model

For each feature model FM, there exists an equivalent feature model FM' with the properties that all features occur in products and the properties PC and AC are equal. A feature model which has these properties will be called a *normal feature model*. FM' can be obtained from FM by deleting all features which do not occur in any product (dead features) and replacing PC by AC. If FM has no products, then FM' is NIL.

If both FM1 and FM2 are normal feature models, the computation of FE(FM3) and PC(FM3,F) is easy:

FE(FM3) = FE(FM1) $\cup$ FE(FM2)
PC(FM3,F) = PC(FM1,F) $\cup$ PC(FM2,F)

Therefore, we propose that the first phase of the computation is the replacement of FM1 and FM2 by the equivalent normal feature models FM1'and FM2'. This requires the computation of dead features and the actual sets of children of FM1 and FM2. This can be done in a straightforward way when SEM(FM1) and SEM(FM2) are not prohibitively large; otherwise one can, in the case of none or only few external constraints, transform these feature models into generalised feature trees and use the techniques from [11], or else use any of the techniques described in [12]. However, the worst case computational time-complexity is exponential in the number of features; where the problem to determine whether a feature model has products is NP-complete, there is no remedy for this.

We now have computed the feature tree of FM3. Let the feature model FM4 be equal to FM3 without external constraints. The elements of REQ(FM4) and EXCL(FM4) need not to be included in RE(FM3) resp. EX(FM3), as these constraints are satisfied already. Therefore RE(FM3) and EX(FM3) can be computed by

RE(FM3) = (REQ(FM1) $\cap$ REQ(FM2)) \ REQ(FM4)
EX(FM3)= (EXCL(FM1) $\cap$ EXCL(FM2)) \ EXCL(FM4)

where the backslash denotes set difference. All relations can be computed in a straightforward way from the semantics of the feature models.

V. Examples

In our first example, FM1 is given in Fig. 3(a), FM2 in Fig. 3(b) and FM3 = merge(FM1,FM2) in Fig. 3(c).

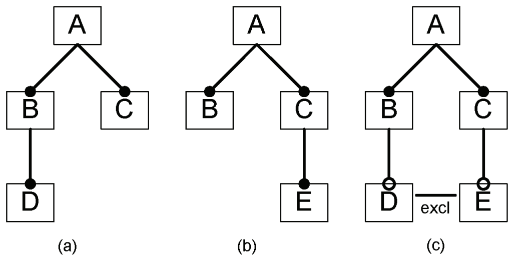

Figure 3. Example with new external constraint

Neither FM1 nor FM2 contains dead features, so FE(FM3) = {A,B,C,D,E}. The tree structure of FM3 is a consequence of the requested parent-compatibility. The possible sets of children of B in FM3 are determined as follows: PC(FM3,B) = AC(FM1,B) $\cup$ AC(FM2,B) = {{D}} $\cup$ {{}} = {{D},{}}. The possible sets of children of the other features of FM3 are determined similarly.

It remains to compute the sets of external constraints. We start with the computation of REQ(FM1). This relation contains all ordered pairs from {A,B,C,D,E} except the pairs (F,E) with F in {A,B,C,D}. Note that (E,A) belongs to REQ(FM1) since all products of FM1 which contain E, although their number is zero, do contain A. REQ(FM2) contains is all ordered pairs from {A,B,C,D,E} except the pairs (F,D) with F in {A,B,C,E}. So REQ(FM1) $\cap$ REQ(FM2) consists of the following 17 ordered pairs: {(A,A), (A,B), (A,C), (B,A), (B,B), (B,C), (C,A), (C,B), (C,C), (D,A), (D,B), (D,C), (D,D), (E,A), (E,B), (E,C), (E,E)}. All these 17 ordered pairs are require constraints; however, from FM3 constructed thus far we see that all of them are implied already, so RE(FM3) can be taken to be the empty set.

Next we obtain EXCL(FM1) = {(A,E),(B,E),(C,E),(D,E), (E,A),(E,B),(E,C),(E,D)} and EXCL(FM2) = {(A,D),(B,D), (C,D),(D,A),(D,B),(D,C),(D,E),(E,D)}; it follows that

EXCL(FM1) ∩ EXCL(FM2) = {(D,E),(E,D)}. These two pairs are (equal) exclude constraints, and since they are not yet satisfied by FM3 as constructed thus far, we find EX(FM3) = {(D,E)}. So this example shows that a merged feature model may have external constraints, even if none of its constituents do have external constraints.

For our second example, let FM1 be the feature model whose feature tree is depicted in Fig. 4:

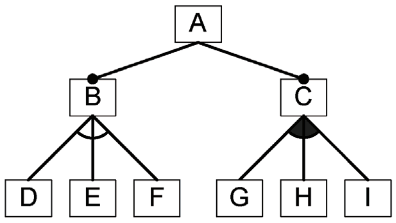

Figure 4. Feature tree of FM1

and whose external constraints are given by:

RE(FM1) = {(D,G),(D,H), (E,G), (E,I), (F,H),(F,I)}
EX(FM1) = {(D,I),(E,H),(F,G)}

There are three products: ABCDGH, ABCEGI, and ABCFHI. FM1 is not normal, since PC(FM1,C) contains 7 allowable sets of children of C, while only three of them occur in products: AC(FM1,C) = {GH,GI,HI}.

So, in the first phase of the merge process, FM1 is replaced by an equivalent normal feature model FM1' whose feature tree is given in Fig. 5. Here the notation [2..2] means that the number of children is at least 2 and not more than 2. Note that some of the external constraints have become superfluous; for instance, we might replace either RE(FM1) or EX(FM1) by the empty set. Such a simplification is not necessary, but it might ease the upcoming computations.

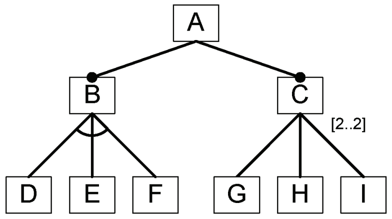

Figure 5. Feature tree of FM1'

Let FM2 be the normal feature model whose feature tree is depicted in Fig. 6:

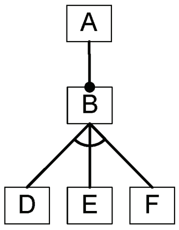

Figure 6. Feature tree of FM2

and which does not have external constraints. There are three products: ABD, ABE and ABF.

The feature tree of FM3 is computed to be the feature tree of Fig. 7. For instance, since AC(FM1',A) = {{B,C}} and AC(FM2,A) = {{B}}, we have PC(FM3,A) = {{B,C},{B}}.

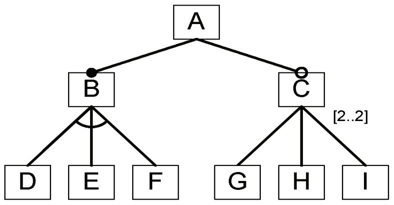

Figure 7. Feature tree of merge (FM1,FM2)

Let FM4 be the feature model whose feature tree is given by Fig. 7 and which has no external constraints. We have to compute the relations REQ(FM1), REQ(FM2), REQ(FM4), EXCL(FM1), EXCL(FM2) and EXCL(FM4) on the set FE(FM3) = {A,B,C,D,E,F,G,H,I}. The results are given in Tables I VI. In these tables a pair (X,Y) appears in row X and column Y.

TABLE I. REQ(FM1)

	A	B	C	D	E	F	G	H	I
A	x	x	x	.	.	.	.	.	.
B	x	x	x	.	.	.	.	.	.
C	x	x	x	.	.	.	.	.	.
D	x	x	x	x	.	.	x	x	.
E	x	x	x	.	x	.	x	.	x
F	x	x	x	.	.	x	.	x	x
G	x	x	x	.	.	.	x	.	.
H	x	x	x	.	.	.	.	x	.
I	x	x	x	.	.	.	.	.	x

TABLE II. REQ(FM2):

	A	B	C	D	E	F	G	H	I
A	x	x	.	.	.	.	.	.	.
B	x	x	.	.	.	.	.	.	.
C	x	x	x	x	x	x	x	x	x
D	x	x	.	x	.	.	.	.	.
E	x	x	.	.	x	.	.	.	.
F	x	x	.	.	.	x	.	.	.
G	x	x	x	x	x	x	x	x	x
H	x	x	x	x	x	x	x	x	x
I	x	x	x	x	x	x	x	x	x

TABLE III. REQ(FM4)

	A	B	C	D	E	F	G	H	I
A	x	x	.	.	.	.	.	.	.
B	x	x	.	.	.	.	.	.	.
C	x	x	x	.	.	.	.	.	.
D	x	x	.	x	.	.	.	.	.
E	x	x	.	.	x	.	.	.	.
F	x	x	.	.	.	x	.	.	.
G	x	x	x	.	.	.	x	.	.
H	x	x	x	.	.	.	.	x	.
I	x	x	x	.	.	.	.	.	x

TABLE IV. EXCL(FM1)

	A	B	C	D	E	F	G	H	I
A	.	.	.	.	.	.	.	.	.
B	.	.	.	.	.	.	.	.	.
C	.	.	.	.	.	.	.	.	.
D	.	.	.	.	x	x	.	.	x
E	.	.	.	x	.	x	.	x	.
F	.	.	.	x	x	.	x	.	.
G	.	.	.	.	.	x	.	.	.
H	.	.	.	.	x	.	.	.	.
I	.	.	.	x	.	.	.	.	.

TABLE V. EXCL(FM2)

	A	B	C	D	E	F	G	H	I
A	.	.	x	.	.	.	x	x	x
B	.	.	x	.	.	.	x	x	x
C	x	x	x	x	x	x	x	x	x
D	.	.	x	.	x	x	x	x	x
E	.	.	x	x	.	x	x	x	x
F	.	.	x	x	x	.	x	x	x
G	x	x	x	x	x	x	x	x	x
H	x	x	x	x	x	x	x	x	x
I	x	x	x	x	x	x	x	x	x

TABLE VI. EXCL(FM4)

	A	B	C	D	E	F	G	H	I
A	.	.	.	.	.	.	.	.	.
B	.	.	.	.	.	.	.	.	.
C	.	.	.	.	.	.	.	.	.
D	.	.	.	.	x	x	.	.	.
E	.	.	.	x	.	x	.	.	.
F	.	.	.	x	x	.	.	.	.
G	.	.	.	.	.	.	.	.	.
H	.	.	.	.	.	.	.	.	.
I	.	.	.	.	.	.	.	.	.

We compute (REQ(FM1) ∩ REQ(FM2)) \ REQ(FM4) and find that it is empty. This means that we may take RE(FM3) to be the empty set.

We compute (EXCL(FM1) ∩ EXCL(FM2)) \ EXCL(FM4) and obtain the result which is shown in Table VII, from which it follows that we may take EX(FM3) = {(D,I),(E,H),(F,G)}, which completes the computation of the merge(FM1,FM2).

TABLE VII. (EXCL(FM1) ∩ EXCL(FM2)) \ EXCL(FM4)

	A	B	C	D	E	F	G	H	I
A	.	.	.	.	.	.	.	.	.
B	.	.	.	.	.	.	.	.	.
C	.	.	.	.	.	.	.	.	.
D	.	.	.	.	.	.	.	.	x
E	.	.	.	.	.	.	.	x	.
F	.	.	.	.	.	.	x	.	.
G	.	.	.	.	.	x	.	.	.
H	.	.	.	.	x	.	.	.	.
I	.	.	.	x	.	.	.	.	.

Note that this example shows that the "normalization" phase of the merge process cannot be abandoned. If we would have abandoned the normalization phase, then the merged tree of Fig. 7 would have an "or" annotation instead of the "[2..2]" annotation at feature C, while the external constraints would be the same. Then the merged result would have the product ABCDG, while FM3 does not have this product.

Also note that in this example the set of products of FM3 is equal to the union of the sets of products of FM1 and FM2. This is not generally the case. In the introduction we already gave an example where a merge has a product which is not a product of one of its constituents.

VI. PROPERTIES OF MERGED FEATURE MODELS

In this section we give some properties of merged feature models; the properties for which we provide no proof are straightforward consequences of the construction.

Property 1: For all parent-compatible feature models FM1, FM1', FM2, and FM2': If FM1 ~ FM1' and FM2 ~ FM2' then merge(FM1,FM2) ~ merge(FM1',FM2')

This property formally specifies that the specification of the merge operation of feature models is compliant with the semantics of feature models.

Property 2: For all feature models FM:
merge(FM,NIL) ~ FM and merge(NIL,FM) ~ FM

Property 3: For all feature models FM:
merge(FM,FM) ~ FM

So, a feature model does not change when it is merged with a feature model without products or with itself.

Property 4: Let FM1 and FM2 be parent-compatible feature models. Then merge(FM1,FM2) is parent-compatible with both FM1 and FM2.

Property 5: Let FM1 and FM2 be parent-compatible feature models. Then:
SEM(FM1) ∪ SEM(FM2) ⊆ SEM(merge(FM1,FM2)).

Property 6: Let FM1 and FM2 be parent-compatible feature models and let FM3 = merge (FM1,FM2). With the notational conventions of section3:
REQ(FM3) = REQ (FM1) ∩ REQ(FM2) and
EXCL(FM3) = EXCL(FM1) ∩ EXCL(FM2).

Proof. In section 3 we argued that REQ(FM3) ⊇ REQ (FM1) ∩ REQ(FM2) and EXCL(FM3) ⊇ EXCL(FM1) ∩ EXCL(FM2). It is therefore sufficient to prove that REQ(FM3) ⊆ REQ(FM1) and EXCL(FM3) ⊆ EXCL(FM1).

Suppose (A,B) ∈ REQ(FM3). Then ∀P ∈ SEM(FM3) : A∈P ⇒ B∈P. Since SEM(FM1) ⊆ SEM(FM3) (property 5) we have ∀P ∈ SEM(FM1) : A∈P ⇒ B∈P, which means that (A,B) ∈ REQ(FM1). Thus REQ(FM3) ⊆ REQ(FM1).

Suppose (A,B) ∈ EXCL(FM3). Then ∀P ∈ SEM(FM3) : A∈P ⇒ B∉P. Since SEM(FM1) ⊆ SEM(FM3) (property 5) we have ∀P ∈ SEM(FM1) : A∈P ⇒ B∉P, which means that (A,B) ∈ EXCL(FM1). Thus EXCL(FM3) ⊆ EXCL(FM1).

The next property implies that when the result of a merge operation is merged again, the normalization phase may be omitted.

Property 7: Let FM1 and FM2 be parent-compatible feature models and let FM3 = merge (FM1,FM2). Then FM3 is a normal feature model.

Proof. We have to prove that FM3 has no dead features and AC(FM3,F) = PC(FM3,F) for all F in FE(FM3). FE(FM3) only contains features which occur in products of FM1 or FM2. Property 5 gives that these products all belong to SEM(FM3). Thus all features of FM3 occur in products of FM3, so FM3 has no dead features. For any feature F $\in$ FE(FM3), and any set Fs $\in$ PC(FM3,F), either Fs $\in$ AC(FM1,F) or Fs $\in$ AC(FM2,F). So there is a product P with P$\in$ SEM(FM1) or P $\in$ SEM(FM2) where Fs is the set of children of F in P. Since P $\in$ SEM(FM3) as well (property 5), Fs $\in$ to AC(FM3,F). So AC(FM3,F) = PC(FM3,F).

The next property states that our merged feature models are indeed minimal, as announced in the introduction.

Property 8: Let FM1 and FM2 be parent-compatible feature models and let FM3 = merge (FM1,FM2). Let FM4 be a feature model which is parent-compatible with FM1 and FM2 and satisfies SEM(FM1) $\cup$ SEM(FM2) $\subseteq$ SEM(FM4). Then SEM(FM3) $\subseteq$ SEM(FM4).

Proof. Let P $\in$ SEM(FM3). We will prove that P $\in$ SEM(FM4). Let F be a feature of P and let Fs be the set of children of F in P. We have to prove:

1. F $\in$ FE(FM4).
2. F $\neq$ RO(FM4) $\Rightarrow$ PA(FM4,F) $\in$ P.
3. Fs $\in$ PC(FM4,F).
4. P satisfies the constraints of RE(FM4).
5. P satisfies the constraints of EX(FM4).

Proof of 1. F $\in$ FE(FM3). By definition, this means that F occurs in a product P' with P' $\in$ SEM(FM1) and/or P' $\in$ SEM(FM2). Since SEM(FM1) $\cup$ SEM(FM2) $\subseteq$ SEM(FM4), P' $\in$ SEM(FM4). Thus F $\in$ FE(FM4).

Proof of 2. Suppose F $\neq$ RO(FM4). Then F $\neq$ RO(FM3) and PA(FM3,F) = PA4(FM4,F), due to the parent-compatibility of FM1 and FM2 with FM3 and FM4. PA(FM3,F) $\in$ P, since P $\in$ SEM(FM3), so PA(FM4,F) $\in$ P.

Proof of 3. Fs $\in$ PC(FM3,F), so Fs $\in$ AC(FM1,F) or Fs $\in$ AC(FM2,F). This means that there exists a product P' $\in$ SEM(FM1)$\cup$SEM(FM2) which contains F, and the set of children of F is Fs. Since P' $\in$ SEM(FM4), Fs $\in$ PC(FM4,F).

Proof of 4. Suppose (A,B) $\in$ RE(FM4) and P violates it. Then A $\in$ P and B $\notin$ P. Suppose first B $\notin$ FE(FM3). Then there is no product in SEM(FM1) $\cup$ SEM(FM2) which contains B. Since A $\in$ FE(FM3) there is a product P' $\in$ SEM(FM1) $\cup$ SEM(FM2) with A $\in$ P'. Then P' $\in$ FE(FM4) and P' violates constraint (A,B). This is a contradiction, so B $\in$ FE(FM3). It follows that (A,B) $\notin$ REQ(FM3). Therefore (A,B) $\notin$ REQ(FM1) $\cap$ REQ(FM2). Suppose, without loss of generality, that (A,B) $\notin$ REQ(FM1). Then there exists a product P'' $\in$ SEM(FM1) such that A $\in$ P'' and B $\notin$ P''. Thus P'' $\in$ SEM(FM4), which violates the constraint (A,B) of RE(FM4). This is a contradiction; so it follows that P satisfies all constraints of RE(FM4).

Proof of 5. Suppose (A,B) $\in$ EX(FM4) and P violates (A,B). Then A $\in$ P and B$\in$ P. Since A $\in$ FE(FM3) and B $\in$ FE(FM3), (A,B) $\notin$ EXCL(FM3). Therefore (A,B) $\notin$ EXCL(FM1) $\cap$ EXCL(FM2). Suppose, without loss of generality, that (A,B) $\notin$ EXCL(FM1). Then there exists a product P'' $\in$ SEM(FM1) such that A $\in$ P'' and B $\in$ P''. Then P'' $\in$ SEM(FM4), which violates the constraint (A,B) of EXCL(FM4). This is a contradiction; so it follows that P satisfies all constraints of EX(FM4).

Property 9: The merge operation is both commutative and associative.

Property 10: Each feature model is the merging of all its products.

VII. Related work

Segura et al. [3] provide a catalogue of graph transformation rules for the merging of feature models. However, properties of their rewrite system have not been derived; examples show that equivalent feature models may lead to nonequivalent results, meaning that their merge operation is not compliant with the semantics of the feature models.

Chen et al.[4] describe an approach where a number of products are merged; the resulting feature model has only mandatory and optional nodes. So, their approach has a very restricted applicability. Moreover, where they did not impose the condition of parent-compatibility, their resulting feature models contain multiple occurrences of features; this due to their algorithm which recursively merges trees, and misses equal features which do not have equal parents.

Acher et al. [5] define rules to determine the variability operator of a merged feature from the variability operators of the feature in the component feature models (e.g. merging a or-feature and an xor-feature gives an xor-feature). They do not consider cross-tree constraints. As in [4], their recursive tree merging algorithm misses equal features which do not have equal parents, due to the absence of the condition of parent-compatibility

Schobbens et al. [6] define merging of feature models with sharing. Using non-primitive features, they transform the diagrams of the constituents such that the primitive features become leaves, put the constituents alongside each other, add a new common xor-root, and introduce sharing of common primitive features. The merged feature model has precisely the products of both constituent feature models. This approach can not be adopted if sharing is not allowed.

The Arborcraft approach of Weston et al. [7] semi-automatically derives feature models from textual requirement specifications. This approach requires merging whenever the derivation results in multiple feature models, which occurs when multiple requirements documents need to be processed.

Currently, Arborcraft supports this by merging the textual documentation before deriving a feature model. However, this is not possible for, for example, extractive and reactive software product line development as not all requirements documentation will be available at the same moment in time.

Apel et al. [9] define an algebra for features and feature composition; they define feature composition to be a merging of trees, not of feature models.

Hartmann and Trew [10] consider feature merging in the context of Multiple Product Line Feature models, which are combination of conventional feature models and context variability models. They consider feature merging to require a significant engineering theory, which can hardly be automated.

VIII. Conclusion

For feature models which are trees with "requires" and "excludes" constraints, we have specified the merging of two parent-compatible feature models as the minimal feature model which has all the products of the constituents and which is parent-compatible with the constituents. This specification is compliant with the semantics of feature models. We have shown how the specification can be implemented, presented a number of properties of the merge operation and gave some examples.

Acknowledgments

This work is supported by the European Commission grants IST-33710 - Aspect-Oriented, Model-Driven Product Line Engineering (AMPLE) and IEF-221280 - Dealing with Imperfect Information in Systemic Concerns of Software Product Line Requirements (DISCS).

References

[1] 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 CMU/SEI-90-TR-21, Software Engineering Institute, Carnegie Mellon University (1990).

[2] K. Czarnecki and U. Eisenecker, Generative Programming: Methods Tools and Applications. Addison-Wesley (2000).

[3] S. Segura, D. Benavides, P. Trinidad and A. Ruiz-Cortés, "Automated merging of feature models using graph transformations", In: R. Lämmel, J. Visser and J. Saraiva (eds.), Proceedings of the Summer School on Generative and Transformational Techniques in Software Engineering (GTTSE'07), Lecture Notes Computer Science, vol. 5235, Springer-Verlag, pp. 489--505 (2008).

[4] K. Chen, W. Zhang, H. Zhao and H. Mei, "An approach to constructing feature models based on requirements clustering", In: 13th IEEE International Conference on Requirements Engineering (RE'05), IEEE Computer Society pp. 31--40 (2005).

[5] M. Acher, Ph. Collet, Ph. Lahire and R. France, "Composing feature models", In: Proceedings of the Second International Conference on Software Language Engineering (SLE'09), Lecture Notes in Computer Science, vol. 5969, Springer-Verlag, pp. 62 - 81 (2010).

[6] P.-Y. Schobbens, P. Heymans, J.-Chr. Trigaux and Y. Bontemps, "Generic semantics of feature diagrams", Computer Networks 51, pp. 456--479 (2007).

[7] N. Weston, R. Chitchyan and A. Rashid, "A framework for constructing semantically composable feature models from natural language requirements. In: 13th International Software Product Line Conference, San Francisco, USA, 2009.

[8] V. Alves, R. Gheyi, T. Massoni, U. Kulesza, P. Borba and C. Lucena, "Refactoring product lines", In: 5th international conference on Generative Programming and Component Engineering (GPCE '06), ACM, pp. 201--210 (2006).

[9] S. Apel, C. Lengauer, B. Möller and C. Kästner, "An algebra for features and feature composition. In: J. Mesequer and G. Rosu (eds.), Algebraic Methodology and Software Technology (AMAST 2008), Lecture Notes in Computer Science, vol 5140, Springer-Verlag, pp. 36--50 (2008).

[10] H. Hartmann and T. Trew, "Using feature diagrams with context variability to model multiple product lines for software supply chains". In: B. Geppert and K. Pohl (eds.), Proceedings of the 12th International Software Product Line Conference (SPLC 2008), pp. 12--21 (2008).

[11] P. van den Broek and I. Galvão, "Analysis of feature models using generalised feature trees". In: D. Benavides, A. Metzger and U. Eisenecker (eds.), 3th International Workshop on Variability Modelling of Software-intensive Systems, ICB-Research Report 29, University of Duisburg-Essen, pp. 29--36 (2009).

[12] D. Benavides, S. Segura and A. Ruiz-Cortés, "Automated analysis of feature models 20 years later: a literature review", Information Systems, in press (2010).