特征模型融合研究浅析

易立，赵海燕+，张伟，金芝，梅宏

北京大学信息科学技术学院 软件研究所 教育部高可信软件技术重点实验室 100871

摘 要 在针对特定领域的软件复用活动中，特征模型为软件资产的组织和复用提供了有效的手段。特征模型的构建需要对领域内尽可能多的具体应用软件进行系统化的分析和抽象。随着特征模型的广泛应用和领域的日益复杂，特征模型的规模也日益增大，而大规模特征模型的构建是一件十分困难的工作，亟需自动化方法的支持。一种可能的支持方式是将领域内已有的多个规模较小的特征模型自动融合为一个较大的特征模型，然后人工对融合结果进行微调。基于现有文献中提出的六种特征模型融合方法，本文提出了一个特征模型融合的概念框架，对六种方法进行了比较和分析。另外，本文还指出了现有特征模型融合方法研究中尚未解决的三个问题，并针对每个问题提出了可能的研究思路和设想。

关键词 特征模型；模型融合；算法

中图法分类号 TP311.5 DOI 号：

A Survey on the Research of Feature Models Merging

YI Li, ZHAO Hai-yan+, ZHANG Wei, JIN Zhi, MEI Hong

Institute of Software, School of EECS, Peking University

Key Laboratory of High Confidence Software Technology, Ministry of Education of China

Beijing 100871, China

Abstract Feature models provide an effective way to organize and reuse software assets in a specific domain. Constructing a feature model needs a systematic review and abstraction of as many applications as possible in a domain. With the increasing use of feature models and increasing complexity of domains, feature models become more and more complex, and the construction of complex feature models is a difficult task that would benefit from computer-aided automation. A possible way of automation is to merge existing feature models into a large one, and human developers only need to do some refactoring work. In this paper, we survey six methods of merging feature models. We propose a conceptual framework first, and then analyze and compare the six methods. Finally, we identify three problems in existing research, and propose possible ideas to handle these problems.

Key words feature model; merge; algorithm

本课题得到国家重点基础研究发展计划（973）(No. 2009CB320701)；国家自然科学基金关键项目 (No. 90818026)；国家自然科学基金（No. 60873059）；国家科技支撑计划 (No. 2008BAH32B02)资助. 易立，男，1984年生，博士研究生，E-mail: yili07@sei.pku.edu.cn，主要研究领域为领域工程. 赵海燕，女，1966年生，博士，E-mail: zhhy@sei.pku.edu.cn，副教授，主要研究领域为软件复用. 张伟，男，1978年生，博士，E-mail: zhangw@sei.pku.edu.cn，副教授，主要研究领域为软件复用. 金芝，女，1962年生，博士，E-mail: zhijin@sei.pku.edu.cn, 教授，主要研究领域为软件工程、软件复用. 梅宏，男，1963年生，博士，E-mail: meih@pku.edu.cn，博士，主要研究领域为软件工程、软件复用.

第1作者手机号码: 18685634646, E-mail: yili07@sei.pku.edu.cn

1 引言

软件复用的研究和实践表明，针对特定领域的软件复用活动相对容易取得成功[15]。在针对特定领域的软件复用方法中，Kang 等人[10]首先提出了面向特征的领域分析方法。特征是具有用户/客户价值的软件特点[10, 14]。按照特征之间的依赖关系将特征组织起来，就形成了特定领域的特征模型。特征模型的构建需要对领域内尽可能多的具体应用软件进行系统化的分析和抽象，封装这些软件的功能和非功能特性为不同粒度和层次的特征，提取这些特征之间的共性和变化性，并分析它们之间的依赖关系。

目前，特征模型被广泛应用在多个领域，例如汽车制造和电子产品制造领域。随着领域的日益复杂，特征模型已达到数百甚至数千个特征的规模[5]，构建这种大规模特征模型是一件十分复杂的工作[3, 5, 12]，特征模型建模者亟需自动化工具的支持以降低工作的强度和复杂度。借用软件复用的思想，一种可能的支持方式是复用领域内已存在的多个规模较小的特征模型来构建较大的特征模型。这种复用可分为两步：首先自动将多个已有特征模型融合为一个特征模型，然后人工对这个新的模型进行调整。随着特征建模方法和软件产品线方法（特征模型在大多数情况下作为其核心制品存在）的广泛应用，寻找领域内的可复用特征模型将不再是问题。例如，在线特征模型库 SPLOT 中包含十多个移动电话领域的特征模型，Czenarcki 等研究者发布了三个包含数千特征的嵌入式操作系统领域特征模型[13]。因此，基于特征模型融合的特征模型构建方法成败的关键在于融合得到的结果模型是否具有较好的质量，从而使得后续人工重构工作足够小，与从零开始构建特征模型相比能够显著的节省时间和人力。

本文对现有文献中提出的 6 种特征模型融合方法从三个方面进行比较与分析。首先是输入模型的限制，限制越少，说明融合方法具有越广泛的适用性。其次是输出模型的质量，质量越高，说明输出模型与从零开始人工构建的模型越相似，从而所需后续重构工作越少。最后是融合方法实现所具备的性质，包括时间复杂度和冲突消解等方面。本文基于上述方面提出了一个详细的特征模型融合概念框架，并基于该框架对 6 种方法进行了比较和分析。在比较和分析的基础上，本文还指出了现有方法中存在的三个问题，并对每个问题给出了研究意见和设想。

本文后续章节的组织方式如下。第 2 节介绍了特征模型的相关概念。第 3 节阐述了本文所提出的特征模型融合概念框架。第 4 节对现有方法进行了比较和分析。第 5 节对现有方法中存在的问题和可能的解决思路进行了介绍。最后，第 6 节对全文进行了总结。

2 特征模型

特征模型是对特定领域内一系列相似产品的共性和变化性的抽象，由一组特征和它们之间的关系构成。图 1 是一个音频播放软件领域的特征模型示例。特征是具有用户/客户价值的软件特点，特征之间的关系可分为精化关系和约束关系两类。精化关系将特征组织为一棵特征树，特征树上父子特征之间的关系主要可分为以下几种类型：

- 必选特征：一个必选特征必须与它的父特征同时出现或不出现在一个产品中。例如，音频播放软件必包含解码器和运行平台两个特征。
- 可选特征：一个可选特征可以不与其父特征同时出现在一个产品中。
- 异或（xor）组：当一个异或组的父特征出现在一个产品中时（例如图 1 中的运行平台），该组特征中有且仅有一个特征出现在同一产品中（例如图 1 中的手机和桌面）。
- 或（or）组：当一个或组的父特征出现在一个产品中时，该组特征中可以有一到多个特征出现在同一产品中。

约束关系用以表达特征之间额外的依赖。基本的约束关系主要有两种：

- requires：特征 X requires 特征 Y 的语义是当 X 出现在某产品中时，Y 必须出现在同一产品中。
- excludes：互为 excludes 关系的两个特征不能出现在同一产品中。

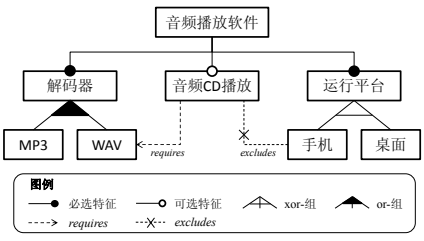


图 1 特征模型示例

从一个特征模型中可以定制出一系列产品，其具体做法是选择一个不违反精化和约束关系的特征集合。例如，特征集合{音频播放软件，解码器，WAV，音频 CD 播放，运行平台，桌面}是图 1 的一个产品，而形如{…, 运行平台，手机，音频 CD 播放, …}的特征集合则不是，因为它违反了模型中的 excludes 约束。

3 概念框架

本节首先给出一些术语定义（3.1 节），然后提出一个特征模型融合的概念框架，以描述和比较现有特征模型融合方法。总体而言，特征模型融合是将两个输入特征模型合并为一个输出特征模型；相应的，概念框架也分为三部分，包括输入特征模型的性质（3.2 节），输出模型的性质（3.3 节），以及融合算法实现方面的特性（3.4 节）。

3.1 术语定义

本小节给出一些术语的定义及符号，以便描述特征模型融合的概念框架。

定义 1 (特征模型的产品集合). 给定特征模型 M，M 的所有合法定制结果构成的集合称为 M 的产品集合，记为 ProdSet(M).

定义 2 (特征精化路径). 给定特征模型中的特征 F 以及 F 的祖先特征 A，沿着精化关系从 A 到 F 所形成的路径称为从 A 到 F 的特征精化路径（以下简称“路径”），记为 Path(A, F). 当 A 与 F 为同一个特征时，称 Path(A, F)为“平凡路径”.

定义 3 (特征集合). 给定特征模型 M 或路径 P，M（或 P）中所有特征构成的集合称为 M（或 P）的特征集合，记为 FSet(M)或 FSet(P).

定义 4 (路径集合). 给定特征模型 M，M 中所有可能的路径构成的集合称为 M 的路径集合，记为 PathSet(M).

定义 5 (相等特征🗁). 称两个特征 X 和 Y 相等（X 和 Y 可以属于不同特征模型），当且仅当 X 和 Y 的名称从字面上完全相同，记为 X = Y.

定义 6 (相等路径). 称 Path(A_1, F_1)和 Path(A_2, F_2)是相等路径并记为 Path(A_1, F_1) = Path(A_2, F_2)，当且仅当下列条件全部满足：

（1）Path(A_1, F_1)和 Path(A_2, F_2)长度相等；

（2）$F_1 = F_2$；

（3）Path(A_1, F_1)和 Path(A_2, F_2)均为平凡路径；或者 P_1 是 F_1 的父特征，且 P_2 是 F_2 的父特征，且 Path(A_1, P_1) = Path(A_2, P_2).

定义 7 (正交路径). 若记 Path(A_1, F_1)为 P_1, Path(A_2, F_2)为 P_2. 称 P_1 和 P_2 是正交路径, 当且仅当 $FSet(P_1) \cap FSet(P_2) = \emptyset$. 记为 $P_1 \perp P_2$,

定义 8 (相交路径). 若 Path(A_1, F_1)和 Path(A_2, F_2)既非相等路径，也非正交路径，则称二者为相交路径.

3.2 输入

本小节讨论融合方法对输入特征模型的性质所作的要求，这些要求分别针对精化关系和约束关系而提出。

首先是**层次一致性**。文献[5]首先给出了层次一致性的定义。给定两个特征模型 M_1 和 M_2，并记 $Root_1$、$Root_2$ 分别为 M_1 和 M_2 的根特征。如果下列条件成立：

$$\forall F(F \in FSet(M_1) \cap FSet(M_2) \rightarrow Path(Root_1, F) = Path(Root_2, F)).$$

则称这两个特征模型具有层次一致性。实际应用场景中的输入特征模型往往不满足层次一致性，因而本文关注每种特征模型融合方法能否处理以及如何处理层次不一致的输入。

其次是**根特征相等性**。该性质表明各输入特征模型是否需要具备相等的根特征。

最后是能否**处理约束关系**。约束关系是保证特征模型定制结果正确性和一致性的重要手段，因此恰当的融合约束关系对特征模型融合方法的实用性起关键作用。本文主要关注每种特征模型融合方法能否处理输入模型中的约束关系。

3.3 输出

本小节讨论输出模型应具备的性质和特点，主要从输出模型的产品集合、规模、层次结构等方面进行描述。

构建特征模型的目的是快速定制特定领域内的一系列产品，也就是说，一个特征模型对应一个产品集合。特征模型融合是构建特征模型的一种特殊方法。因此，本文首先关注输入输出模型所对应**产品集合**之间的关系。给定两个输入特征模型 X_1 和 X_2，且 Y 表示输出模型，则 ProdSet(Y)可能满足如下两个等式之一：

$$ProdSet(Y) \subseteq ProdSet(X_1) \cap ProdSet(X_2), \quad (1)$$

🗁 此定义遵循特征模型融合研究的惯例，并不适用于普遍情况下特征相等性的判定。

$$\mathrm{ProdSet(Y)} \supseteq \mathrm{ProdSet(X_1)} \cup \mathrm{ProdSet(X_2)}, \quad (2)$$

本文称符合等式（1）的输出模型具有“产品交集语义”，而符合等式（2）的输出模型具有“产品并集语义”；当等式中等号成立时，分别称输出模型具有“产品严格交集语义”和“产品严格并集语义”。

输出模型的**规模和层次结构**的特性与模型可读性有关。首先考虑模型的规模，即模型所包含的特征数量。对于具有产品交集语义的输出模型，应该有下式成立：

$$|\mathrm{FSet(Y)}| \leq \min\{\,|(\mathrm{FSet(X_1)}|, |\mathrm{FSet(X_2)}|\,\}$$

同理，具有产品并集语义的输出模型规模应该满足：

$$|\mathrm{FSet(Y)}| \leq |\mathrm{FSet(X_1)}| + |\mathrm{FSet(X_2)}|$$

当上述等式不满足时，则认为输出模型的规模是异常的。

接下来考虑输出模型的层次结构。针对特征模型融合而言，本文关注输入和输出模型的层次结构是否尽量保持一致。如果输出模型的层次结构与输入模型相去甚远，则会降低输出模型的可理解性。本文将输入输出模型的层次结构一致性分为以下几种情况讨论：

- 保持公共路径：在各输入模型中都出现的公共路径应该一致的在输出模型中出现。其形式化定义为：

$$\forall \mathrm{P}(\mathrm{P} \in \mathrm{PathSet(X_1)} \cap \mathrm{PathSet(X_2)}$$

$$\rightarrow \mathrm{P} \in \mathrm{PathSet(Y)})$$

- 保持唯一公共路径：各输入模型中都出现的公共路径在输出模型中出现且仅出现一次。
- 保持独特路径：对于产品并集语义而言，如果一个输入模型中的一条路径与另一输入模型的所有路径均正交，则称该路径为独特路径。所有的独特路径均应该完整的出现在输出模型中。其形式化定义为：

$$\forall \mathrm{P}(\mathrm{P} \in \mathrm{PathSet(X_i)} \wedge \mathrm{P} \perp \mathrm{PathSet(X_j)}$$

$$\rightarrow \mathrm{P} \in \mathrm{PathSet(Y)}), \text{其中 } \mathrm{i,j} = 1,2, \mathrm{i} \neq \mathrm{j}.$$

上式中的$\mathrm{P} \perp \mathrm{PathSet(X_j)}$是

$$\forall \mathrm{Q}(\mathrm{Q} \in \mathrm{PathSet(X_j)} \rightarrow \mathrm{P} \perp \mathrm{Q})$$

的简写。

3.4 融合算法实现

概念框架在融合算法实现方面的内容包括主要实现技术、时间复杂度、冲突消解能力以及算法思想的类型四部分内容。其中，一种方法的实现难度由主要实现技术以及是否有成熟的工具支持这些技术而决定。时间复杂度则是衡量具体算法的重要指标。

对不同输入模型之间存在的冲突进行消解的能力也是融合方法实现的重要指标。本文基于特征模型缺陷的概念定义输入模型之间的冲突。根据文献[14]中关于缺陷的分类原则，一个特征模型中可能存在的缺陷可分为如下三类：

- 不一致(Inconsistency)：无法从特征模型中定义出一个合法产品，使得特征模型中的所有约束关系都得到满足。
- 死特征(Dead Feature)：存在不属于任何合法产品的特征。
- 伪可选特征(False Optional Feature)：存在必须出现在所有合法产品中的可选特征。

基于特征模型的缺陷，本文定义输入模型之间的冲突如下：给定两个分别不存在缺陷的输入模型，如果融合结果模型存在缺陷，则称两个输入模型存在冲突。根据融合结果模型中存在的缺陷类型，冲突也可相应分为“不一致冲突”、“死特征冲突”、“伪可选特征冲突”三类。一种特征模型融合方法的冲突消解能力由劣到优依次分为三个级别：

- 无：该融合方法没有考虑任何类型的冲突。
- 检测：该融合方法可以检测某些类型冲突的存在。
- 修复：该融合方法可以修复某些类型的冲突。

最后，现有特征模型融合研究主要提出了三类不同的算法实现思想，分别是简单组合法（3.4.1节），规则法（3.4.2 节），以及逻辑公式法（3.4.3节）。下面对三类算法的基本框架进行介绍。

3.4.1 简单组合法

简单组合法的基本思想是引入一些额外的特征和关系将各输入模型简单拼接在一起，从而得到输出模型。此类算法在文献[8, 9, 11]中提出。框架如算法 1 所示（FM 是特征模型的缩写）。

算法 1. 基于简单组合的融合算法框架

```
function merge (input1: FM, input2: FM): FM {
1.  output ← new FM();
2.  root ← new Feature();
3.  output.set_root (root);

4.  // 添加额外特征（REF_TYPE 是四种精化关系之一）
5.  root.add_child (extraFeature1, REF_TYPE);
    …;
```

```
6.  extraFeature_i.add_child (extraFeature_j, REF_TYPE);
    …;

7.  // 将输入模型的根特征（亦即整个输入模型）放置到某个额外特征下
8.  extraFeature_k.add_child (input_1.get_root(), REF_TYPE);
9.  extraFeature_m.add_child (input_2.get_root(), REF_TYPE);

10. // 添加额外约束关系
11. output.add_constraint (extraConstraint_1);
    …;

12. return output;
}
```

基于简单组合的算法首先创建输出特征模型及其根特征（第 1 至 3 行）。然后根据需要创建额外的特征，这些额外特征既可以是根特征的直接子特征（如第 5 行所示），也可以互相精化形成层次结构（如第 6 行所示）。接下来的步骤是将输入模型整体放置到某个额外特征之下（第 7 到 9 行），换言之，此类算法的特点是输入模型将完整的再现于输出模型中。最后根据需要添加额外的 requires、excludes 约束关系，就完成了特征模型的融合。

3.4.2 规则法

规则法的基本思想是定义一系列规则来处理输入模型中相匹配的特征（即相等特征）。这些规则分为两类，分别处理精化关系和约束关系。因此算法也分为两步，首先匹配并输出精化关系得到特征树，然后匹配并添加约束关系，融合特征树和整个特征模型的算法框架如算法 2 所示。

算法 2. 基于规则的融合算法框架

```
function merge_tree (root_1: Feature, root_2: Feature): Feature {
1.  if (root1 ≠ root2)  return null;
2.  root ← root1.copy();

3.  // 首先处理公共子特征
4.  common ← root_1.children() ∩ root_2.children();
5.  for each (Feature c ∈ common) {
6.      // 递归融合公共子特征
7.      child ← merge_tree (root_1. child(c), root_2. child(c));
8.      // 基于规则，根据 root_1、root_2 和 c 之间的精化关系，
        // 计算 root 和 child 之间的精化关系
9.      ref ← compute_refinement_by_rule (root_1, root_2, c);
10.     root.add_child (child, ref);
11. }

12. // 然后处理非公共子特征
13. for each (Feature f_1 ∈ root_1.children() \ common) {
14.     // 同样基于规则并根据 root_1 和 f_1 之间的精化关系，
        // 计算 root 和 f_1 之间的精化关系
15.     ref_1 ← compute_unique_ref_by_rule (root_1, f_1);
16.     root.add_child (f_1, ref_1);
17. }
18. /// 将 13 至 17 行的 f_1, root_1 相应替换为 f_2, root_2 即可处理第二个输入模型的非公共子特征，在此略去

19. return root;
}

function merge (input_1: FM, input_2: FM): FM {
20. output ← new FeatureModel();

21. // 首先融合特征树
22. root ← merge_tree (input_1.root, input_2.root);
23. output.set_root (root);

24. // 然后融合约束关系得到完整特征模型
25. for each (Constraint c_1 ∈ input_1.constraints) {
26.     for each (Constraint c_2 ∈ input_2.constraints) {
27.         c ← compute_constraint_by_rule (c_1, c_2);
28.         output.add_constraint (c);
29.     }
30. }

31. return output;
}
```

基于规则的算法首先合并具有相同根特征的输入特征树。其中，公共子特征被递归的合并（第 7 行），并根据规则得到合并后的精化关系（第 9 行）；此外，非公共子特征也被分别加到输出结果中，同样的，依据规则来决定输出何种精化关系（第 15 行）。算法的第二步是合并约束关系（第 24 到 30 行），此过程中也需要依靠规则来决定输出何种约束关系（第 27 行）。

综上所述，基于规则的融合方法需要定义三组规则，分别处理公共子特征的精化关系，非公共子特征的精化关系，以及约束关系。现有研究[1, 6, 12]主要在具体规则上存在差异，其他方面几乎完全相

同。

3.4.3 逻辑公式法

逻辑公式法的基本思想是：首先将输入特征模型转换为逻辑公式，然后根据输入逻辑公式定义输出逻辑公式，最后将输出逻辑公式反向转换为特征模型。在这个过程中，第一步和第三步都有专门的研究者提出了有一定实用性的算法，因此现有特征模型融合研究主要关注如何定义输出逻辑公式。下面分别就这三个步骤进行介绍。

特征模型转换为逻辑公式. 文献[4]提出了一种将特征模型转换为一阶谓词逻辑公式的方法。特征模型中每个特征对应逻辑公式的一个变量，而每个精化或约束关系对应一个或多个逻辑蕴含式（如表 1 所示），最终整个特征模型表示为所有逻辑蕴含式的合取。

表 1 特征之间关系与逻辑蕴含式对照表

关系	逻辑蕴含式
特征 C 是特征 F 的必选子特征	$F \to C \wedge C \to F$
特征 C 是特征 F 的可选子特征	$C \to F$
特征 F 有一组互为 OR 关系的子特征 C_1, C_2, …, C_n	$F \to C_1 \vee C_2 \vee \ldots \vee C_n$
特征 F 有一组互为 XOR 关系的子特征 C_1, C_2, …, C_n	$F \to C_1 \vee C_2 \vee \ldots \vee C_n$ 以及 $\bigwedge_{1\le i<j\le n}(C_i \wedge C_j \to \text{false})$
X requires Y	$X \to Y$
X excludes Y	$X \wedge Y \to \text{false}$

将特征模型表示为逻辑公式以后，该特征模型的一个合法产品即满足该逻辑公式的一组变量赋值，其中变量赋值为真意味着对应特征在产品中出现，赋值为假则相反。整个逻辑公式即等价于整个产品集合。

定义输出逻辑公式. 设输入模型 X_1 和 X_2 对应的逻辑公式分别为Φ_1和Φ_2，那么一种基于输出模型的产品集合语义来定义输出逻辑公式的方法是：首先重写输入模型的逻辑公式为

$$\Gamma_i = \Phi_i \wedge \bigwedge_{f \in FSet(X_i) \setminus FSet(X_j)} \neg f$$

其中$i, j = 1,2; i \neq j$.

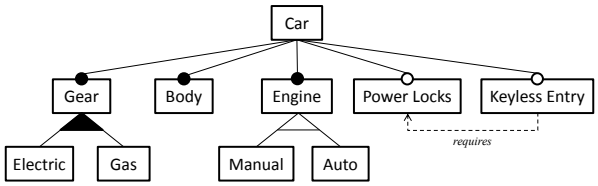
因为上式右边出现的任何 f 均不在模型 X_i 中，所以其取值始终为 false，故Γ_i与Φ_i等价。

然后定义满足严格产品交集和严格产品并集语义输出逻辑公式Φ_Y分别为：

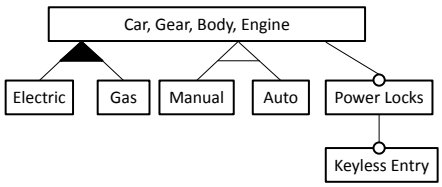
$$\Phi_Y = \Gamma_1 \wedge \Gamma_2$$

$$\Phi_Y = \Gamma_1 \vee \Gamma_2$$

逻辑公式转换为特征模型. 文献[7]提出了一种根据逻辑公式构造特征模型的算法，其基本思想是先构造一个逻辑蕴含图（有向图），然后逐步恢复各类精化关系，最终把逻辑蕴含图变为一个有向森林，即特征森林。该算法主要有两个缺点。首先，由于无法在逻辑公式中区分精化关系和约束关系，该算法将所有的约束关系都当作精化关系看待，因此特征模型中可能包含不正确的精化关系。其次，对于两两之间相互蕴含的一组特征，该算法无法给出它们之间的层次结构，只能当作一个“复合特征”处理。图 2 给出了一个示例，其中(a)是原始特征模型，转换为逻辑公式后，再次转为特征模型得到(b)。可以看到，由于上述缺点的存在，逻辑公式法得到的输出特征模型还需人工重构方能得到满意结果。



(a) 原始特征模型



(b) 与(a)等价的逻辑公式转换得到的特征模型

图 2 逻辑公式与特征模型相互转换示例

4 相关研究比较

基于上一节提出的概念框架，本节对现有特征模型融合研究工作进行介绍和比较。表 2 给出了相关研究工作的概览，其中在“输入模型”部分，用“+”表示该方法可以处理层次不一致、根特征不一致的输入以及输入模型中的约束关系；“−”则相反。从表 2 中可以看到，不同实现算法对融合方法的特性有重要的影响，因此下面根据实现算法分类进行讨论。

基于简单组合的方法[8, 9, 11]最易实现且具有最低的时间复杂度，只需要把两棵输入特征树的根特征连接起来，并求输入约束关系的并集即可。方

法[11]在实现产品交集语义时，需要遍历所有公共特征，因此时间复杂度为 O(min(m, n))。方法[8, 9]需要恰好遍历所有输入特征一次，因此时间复杂度为 O(m+n)。此外，此类方法也是是除了逻辑公式方法以外，唯一可以实现“严格产品交（并）集”语义，且对输入特征模型限制最少的一类算法。然而，此类算法的主要不足体现在输出模型的质量和冲突的消解上。图 3(a)和 3(b)给出了两个层次不一致，且存在冲突的示例输入模型。其中，层次不一致体现为公共特征B在两个输入模型中的父特征不相同。冲突则体现为输入模型 1 蕴含约束关系 B requires D（由显式约束关系 B requires C 和由精化导致的约束关系 C requires D 得出），但输入模型 2 显式声明 B excludes D。图 3(c)给出了方法[11]的严格产品并集语义的输出，其基本思想是用一个虚拟的根特征R以及一个异或关系将两个输入特征树连接起来。可以看到输入约束关系原封不动的加入到输出中，因此冲突并没有得到解决。此外，其输出模型中存在大量冗余，即所有的公共特征（即 A、B、C、D）以及所有公共路径都在输出模型中出现两次，因此模型规模是异常的。方法[8, 9]得到的输出模型类似，在此不再赘述。

基于规则的方法[1, 6, 12]具有多项式时间复杂度，其主要实现技术是遍历输入特征树（图）并变换为输出特征树（图）。此类方法的主要优势在于输出模型的质量较好，而主要缺点在于对输入模型的限制较多，且在冲突消解上仍然存在不足。图 3(d)给出了方法[1, 12]在示例输入模型上的运行结果（方法[6]由于不接受层次不一致的输入模型，所以无法运行）。可以看到输入输出模型的层次结构相当一致，因此输出模型具有较强的可读性。值得注意的是，导致层次不一致的公共特征（如特征 B）将会在输出模型中多次出现，因此表 2 将相应方法的“模型规模”标注为“当层次不一致时异常”，此时输出模型需要人工重构后才可使用。另外，此类方法同样缺乏冲突消解能力。表 2 将方法[1]和[12]的冲突消解能力标注为“部分修复”，是因为对于最简单的显式约束 X requires Y 和 X excludes Y 的情形，规则会将此冲突消解为“X 和 Y 不存在显式约束”，从而达到修复的目的。然而，一旦这两个约束之一是传递得到（如图 3 中的示例）或者是由精化关系隐式蕴含，那么这两种方法将无法检测或修复此冲突。

基于逻辑公式的方法[2]在冲突检测上具有明显优势，且输出模型的产品集合语义和规模都可以良好的保持，同时对于输入模型的限制最少。由于特征模型被转化为逻辑公式，因此可以利用逻辑公式可满足性求解器检测输出模型的不一致，并可以删除死特征。该方法的主要缺点是不易保持层次结构，如图 2 所示，由于特征树上多个特征可能被合并为一个特征，因此输入输出模型的层次结构可能有较大差异。由于方法的实现需要求解可满足性以及寻找最小蕴含(Prime Implicant)，因此实现难度较大，而且时间复杂度是指数级的。

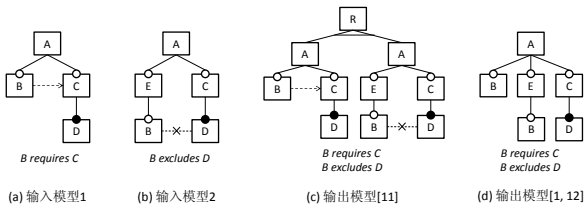


图 3 部分特征模型融合算法的运行示例（产品并集语义）

表 2 特征模型融合相关研究概览

参考文献	[11]	[8] [9]	[12]	[1]	[6]	[2]
输入模型						
层次不一致	+	+	+	+	−	+
根特征不一致	+	+	−	−	−	+
约束关系	+	+	+	−	+	+
输出模型						
产品集合语义	严格交集，严格并集	严格并集	并集	严格交集，并集	并集	严格交集 严格并集
规模	异常		当层次不一致时异常		正常	正常
层次保持	公共路径，独特路径		唯一公共路径，独特路径			*（无法保持）*
算法实现						
类别	简单组合		规则			逻辑公式

主要实现技术	无（直接连接输入根特征）		图变换	树遍历	逻辑公式可满足性(SAT)求解； 最小蕴含(Prime Implicant)计算
冲突检测	无		部分修复不一致	无	检测不一致，修复死特征
时间复杂度	严格交集 O(min(m, n)) 严格并集 O(1)	O(m+n)	O(mn)		$O(2^{m+n})$

注：假设输入模型所包含的特征数量分别为 m 和 n

5 问题与展望

本节针对现有特征模型融合算法中存在的问题与不足，提出可能的研究设想。

5.1 特征匹配问题

任何特征模型融合算法都依赖于各输入模型之间相同特征的正确匹配。现有算法均假设特征匹配由人工完成，这就要求建模人员浏览并理解各输入模型中的所有特征。毫无疑问，上述要求将带来巨大的工作量，因此严重影响了特征模型融合算法的实用价值。一个自然的想法是利用计算机自动进行特征匹配，然而当前的需求工程与领域工程研究尚缺乏此方面的工作。不过在数据库研究领域，一个相似的问题——即“数据库模式匹配”（Schema Matching）——已有了大量的研究工作。这些研究工作提出了对较一般的有向图进行匹配的算法。由于特征模型往往以特征树加上有向的约束关系的形式存在，因此在形式上亦属于有向图，从而可能从上述研究工作中得到启发。

5.2 融合算法实现

现有研究工作主要提出了三种不同的融合算法实现策略，即简单组合策略，基于规则的策略，以及基于逻辑公式的策略。这三种策略各自有明显的不足，要么难以进行冲突检测，要么输出模型的可读性不强。一种可能的解决方案是综合基于规则的策略和基于逻辑公式的策略，实现一种混合策略。具体的设想是，首先将输入模型看作一般的有向树，采用基于规则的策略生成结果有向树。同时采用基于逻辑公式的策略，计算出结果模型对应的逻辑公式并将其表为一系列逻辑蕴含式的合取（如表 1 所示），此时亦可检测不一致冲突和死特征。最后根据表 1，将每一个逻辑蕴含式对应为有向树上的精化关系，或者是横跨有向树的约束关系，从而得到结果特征模型。此过程的关键步骤是将融合后的逻辑公式表为一系列逻辑蕴含式的合取，但注意到表 1 中所列举的各逻辑蕴含式均可表达为析取式，因此问题等价于求解结果逻辑公式的合取范式，从而在理论上是可行的。

5.3 产品集合语义

现有特征模型融合方法提出了两种产品集合语义，即“产品交集”与“产品并集”语义。其主要问题是难以妥善处理输入特征模型中的非公共特征。图 4 给出了一个示例。两个输入模型(a)和(b)分别包括两个非公共特征，(c)和(d)是现有研究工作中两种可能的产品并集融合结果，而(e)是产品交集融合结果。在融合具有非公共特征的特征模型时，其结果应该允许这些非公共特征在不违反原有约束的前提下进行适当的组合，而现有语义并不满足此性质。例如， (a)和(b)的理想融合结果应该能够表达“普清触摸屏、高清非触摸屏”这样的需求，同时保持输入模型的原始约束，从而避免“普清又高清的屏幕”和“触摸且非触摸的屏幕”这样的非法需求。但是通过图 4 可以看到，现有的语义要么根本不允许非公共特征的组合，要么破坏了原有约束。

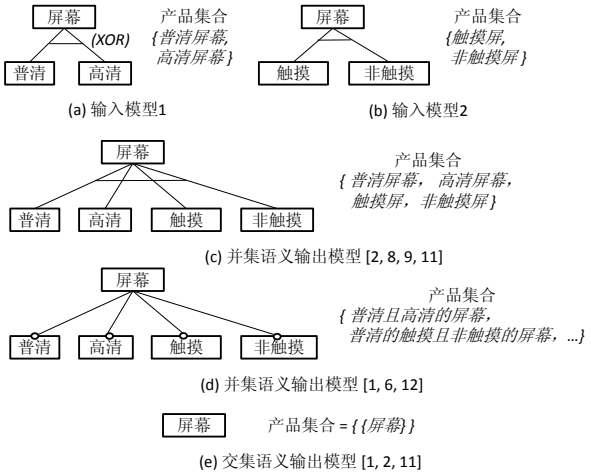


图 4 现有特征模型融合方法的产品集合语义示例

一种可能满足上述要求的语义可以基于产品的笛卡尔积来进行定义，即给定输入特征模型 X_1 和 X_2，输出特征模型 Y 的产品集合满足如下语义：

$$\mathrm{ProdSet(Y)} \subseteq \mathrm{ProdSet(X_1)} \times \mathrm{ProdSet(X_2)}$$

笛卡尔积允许输入模型中的非公共特征在不破坏原有约束关系的前提下进行组合。然而，对于公共特征而言，必须加入其它限制才能保证施加于公共特征上的原有约束关系不被破坏。另外，最终的语义可能难以用规则进行描述，因此 5.2 节所设想的混合方法可能会更适用。

6 结束语

本文对现有特征模型融合研究进行了比较和分析。特征模型融合是一种将同一领域内多个相似特征模型合并为一个特征模型的方法，在特征模型构建和演化中具有重要作用。本文从输入、输出和实现三个方面对六种特征模型融合研究工作进行了分析，得出的结论是：基于简单组合的方法最易实现且适用性最广，然而输出模型的质量不佳；基于规则的方法输出模型的质量最佳，且较易实现，然而适用性受到一定限制；基于逻辑公式的方法适用性广且具备冲突检测能力，但实现代价高且输出模型质量仍不尽如人意。

基于上述比较和分析，本文识别出特征模型融合研究中的三个亟需解决的问题。一是如何匹配输入模型中的相同或相似特征的问题，二是如何结合不同策略来寻求一种在输入、输出和实现三方面都令人满意的方法的问题，三是恰当处理输入模型中的非公共特征的问题。只有良好的解决了这三个问题，特征模型融合算法才能在现实领域的特征模型构建和演化中具备良好的实用性。

参 考 文 献

[1] Acher, M., Collet, P., Lahire, P., France, R.: Composing Feature Models. In: 2nd International Conference on Software Language Engineering (SLE'09). Volume 5969 of LNCS. (2009) 62–81.

[2] Acher, M., Collet, P., Lahire, P., France, R. Managing multiple software product lines using merging techniques. 2010.

[3] Antkiewicz M., Czarnecki K. FeaturePlugin: Feature Modeling Plug-In for Eclipse. In: Proceedings of the 204 OOPSLA Workshop on Eclipse Technology.

[4] Batory, D.S.: Feature models, grammars, and propositional formulas. In: SPLC'05. Volume 3714 of LNCS. (2005) 7–20.

[5] Batory, D., Benavides, D., Ruiz-Cortés, A.: Automated analysis of feature models: Challenges ahead. Communications of the ACM December (2006)

[6] Broek van den, Pim and Galvao, Ismˆenia and Noppen, Joost (2010) Merging Feature Models. In: 14th International Software Product Line Conference, 14 September 2010, Jeju Island, South Korea.

[7] Czarnecki, K., Wasowski, A.: Feature diagrams and logics: There and back again. In: SPLC 2007. (2007) 23–34.

[8] Hartmann, H., Trew, T., Matsinger, A.: Supplier independent feature modeling. In: SPLC'09, IEEE Computer Society (2009) 191–200.

[9] Hartmann, H., Trew, T.: Using feature diagrams with context variability to model multiple product lines for software supply chains. In: SPLC'08, IEEE (2008) 12–21.

[10] Kang, K.C., Cohen, S., Hess, J., Nowak, W., Peterson, S. Feature-oriented Domain Analysis (FODA) Feasibility Study. Technical Report CMU/SEI-90-TR-21, 1990.

[11] Schobbens, P.Y., Heymans, P., Trigaux, J.C., Bontemps, Y.: Generic semantics of feature diagrams. Comput. Netw. 51(2) (2007) 456–479.

[12] Segura, S., Benavides, D., Ruiz-Cortés, A., Trinidad, P.: Automated merging of feature models using graph transformations. In: GTTSE '07. Volume 5235 of LNCS., Springer-Verlag (2008) 489–505.

[13] She, S., Lotufo, R., Berger, T., Wąsowski, A. and Czarnecki, K.: Reverse Engineering Feature Models. In: ICSE '11.

[14] Zhang, W., Mei, H., Zhao, H.Y. A Feature-oriented Approach to Modeling Requirements Dependencies. In: Proceedings of the 13th IEEE International Conference on Requirements Engineering (RE `05), 2005, 273–282.

[15] Li Ke-Qin, Chen Zhao-Liang, Mei Hong, Yang Fu-Qing. An Outline of Domain Engineering. Computer Science, 1999, 26(5), 21-25 (in Chinese).
(李克勤，陈兆良，梅宏，杨芙清．领域工程概述．计算机科学，2006, 26(5): 21-25.)

Background

In the research and practice of software reuse, people have found that domain-specific software reuse is a preferable and feasible way, because domains are usually cohesive (i.e. it is relatively easier to produce a reusable software artifact) and stable (i.e. it is more possible to reuse a reusable software artifact). The first phase of domain-specific software reuse is domain analysis that systematically analyzes commonality and variability among the requirements of applications in a domain.

Domain analysis produces a reusable domain model.

Feature-oriented domain analysis is widely adopted in the academic and industrial world since 1990. The term “feature” is defined as “a cohesive set of requirements” and “a software characteristic having sufficient user/customer value”. The domain model is called *feature model* that consists of features and dependencies between features.

With the increasing complexity of domains, constructing feature models faces scalability problems nowadays. It takes enormous effort to construct real feature models that contain thousands of features and thousands of dependencies. Instead of constructing feature models from scratch, a feasible practice is to merge (and so reuse) existing feature models into a large one, in a (semi-) automated way.

Researchers have proposed six methods of merging feature models in recent years. However, there is still a lack of uniform framework to describe and compare these methods. In this paper, we propose a conceptual framework that defines a method of merging feature models in three aspects. We then compare existing methods based on the framework. Such a systematic analysis allows us to find new problems and new ideas, such as combining the advantages of several methods to design a new method. Besides, we identify three problems that significantly hinder the usability of existing feature model merging methods in practice, and propose some ideas to handle these problems.

This work is supported by the National Basic Research Program of China (973) (Grant No. 2009CB320701), Key Project of National Natural Science Foundation of China (Grant No. 90818026), National Natural Science Foundation of China (Grant No. 60873059), and the National Key Technology R&D Program (Grant No.2008BAH32B02).