基于语义的特征模型重构方法

张力生[1] 张 悦[1] 雷大江[2]
(重庆邮电大学软件工程学院 重庆 400065)[1] (重庆邮电大学计算机科学与技术学院 重庆 400065)[2]

摘 要 在软件产品线的领域工程开发中,特征模型被广泛用于捕获和组织领域的可复用需求。目前,构建特征模型大多依赖于建模人员的分析,而随着领域需求的日益复杂,构建满足需求的特征模型不仅会增加建模人员的工作量,还会使特征模型的正确性降低。为解决不同特征模型之间建模词汇不统一的问题,提出一种分析特征语义并为语义定义术语的方法。为有效地重构特征模型,提出一种采用描述逻辑语言定义半自动化的重构方法,该重构方法可以推理模型的一致性。基于两个特征模型实例对提出的方法进行验证,实验结果表明该方法可以重构特征模型,并且可以检验重构的特征模型的一致性。

关键词 领域工程,特征模型,重构,语义,描述逻辑
中图法分类号 TP311 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2019.05.021

Feature Model Refactoring Method Based on Semantics

ZHANG Li-sheng[1] ZHANG Yue[1] LEI Da-jiang[2]
(College of Software Engineering,Chongqing University of Posts and Telecommunications,Chongqing 400065,China)[1]
(College of Computer Science and Technology,Chongqing University of Posts and Telecommunications,Chongqing 400065,China)[2]

Abstract In the domain engineering of software product lines development,feature model is widely adopted to capture and organize the reusable requirements. Currently,the construction of feature model relies on the modeler's analysis. With the increasing complexity of domain requirements,building a feature model that satisfies the requirements not only increases the workload of the modeler,but also reduces the accuracy of the feature model. A method for analyzing the semantics and defining semantic terms was proposed to solve the problem of inconsistent modeling vocabulary between different feature models in this paper. To refactor the feature model effectively,a semi-automated refactoring method was defined by using Description Logic. The consistency of the model can be also inferred by this method. The proposed method is verified based on two feature models,and the result shows that the method can refactor the feature model as well as verify the consistence the refactored feature model.

Keywords Domain engineering,Feature model,Refactoring,Semantics,Description logic

1 引言

软件复用[1]是一种重复使用现有软件产品以开发新软件产品或维护旧软件产品的方法,它涉及软件开发的成本、组件、重用水平、重用过程等多个方面,贯穿了软件的生命周期。软件复用包括两个重要活动[2],即可复用软件资产的生产(Development for Reuse)和基于可复用软件资产的应用系统开发(Development with Reuse)。软件产品线[3-5]是一种重要的支持软件复用的软件产品开发方法,它通过提取同一领域的多种软件产品的共性,得到软件产品线的核心资产(Core Asset),然后根据用户对软件产品的需求,从核心资产中取出适用于该软件产品的资产并对其进行复用。为了使用户的需求规范化,易于管理和可读,Davis[6]首次提出将"特征"作为一种组织需求的方式。卡耐基·梅隆大学软件工程研究所(SEI)[7]引入了该方式,提出了面向特征的领域分析方法(Feature-Oriented Domain Analysis,FODA),特征模型就是该方法中的一个重要组成部分。

随着软件产品线的不断发展,使用特征模型对领域需求进行分析的方法已经被广泛接受。目前已经有很多关于特征模型的研究,包括通过启发式建立特征模型的方法研究[8]、管理分布式特征模型的方法研究[9]、特征模型重构[3,10]的研究等。其中,特征模型重构[10]是指在不改变特征模型有效配置的前提下对特征模型进行的任何更改,如删除或添加特征、添加新的约束、更改特征模型的拓扑等。

本文主要讨论基于同一领域的多个特征模型集合,将其中一个特征模型作为基础模型,根据其他特征模型提供的组

到稿日期:2018-04-12 返修日期:2018-07-20 本文受重庆市前沿与应用基础研究计划一般项目(cstc2014jcyjA40049)资助。
张力生(1965-),男,硕士,教授,CCF会员,主要研究方向为软件工程、需求建模、领域建模和软件复用,E-mali:zhangls@cqupt.edu.cn;张 悦(1993-),女,硕士生,主要研究方向为软件工程、语义网和描述逻辑,E-mail:S161201004@stu.cqupt.edu.cn(通信作者);雷大江(1979-),男,博士,副教授,主要研究方向为智能计算、机器学习和数据挖掘等。

件,对基础特征模型进行重构的方法。然而,对于这一类特征模型重构的研究,目前还存在一些问题。虽然大部分的特征模型重构研究假设了所有特征模型是在统一的建模词汇基础上进行建立,但在实际重用特征模型时,模型集合中的每一个特征模型往往是由不同的建模团队建立,导致模型与模型之间的建模词汇并不统一。同时,特征模型由于采用图形化的方法建立特征,在具有结构简单和易于复用等优点的同时缺少严格的语义表示,导致关于特征模型重构的大部分工作只能依靠人工完成,当特征模型的规模逐渐增加时,对应的效率和可靠性会逐渐下降。

针对以上问题,本文提出基于语义的特征模型重构方法。该方法将特征模型中的特征映射到现实世界中得到特征的语义,使用描述逻辑语言对特征模型的语义进行形式化,并借鉴文献[11-12]的思想,分别对形式化后的精化关系和约束关系进行重构;同时,利用描述逻辑提供的推理能力,推理出重构的特征模型中的隐含知识并实现特征模型的一致性验证。

本文第 2 节介绍涉及的背景知识;第 3 节描述基于语义的特征模型重构方法;第 4 节通过实例分析对本文提出的方法进行验证;第 5 节讨论相关工作;最后进行总结与展望。

2 背景知识

2.1 特征模型的结构和语义

特征有两类[5]:一类是领域特征,用以定义领域中所有成员的共性;另一类是个体特征,用以区分同一领域内的不同成员。本文提到的特征都是指领域特征。文献[7]认为特征是"具有客户价值的软件特点"。文献[13]认为特征用于"定义软件产品线中的通用部分和可变部分"。文献[3]认为特征有两个视角的定义,即外延和内涵:"从特征的内涵可知,特征的本质是一对需求的分割和概念化,即把一个完整的需求集合分割为一组相对独立的需求子集,并为每个需求子集进行命名以对其概念化并方便引用;特征的外延则提供了一个需求子集是否可被认为是一个特征的标准,即这个需求子集是否描述了一种具有充分的用户或客户价值的软件特点"。

通过对上述特征定义的理解,本文从两个视角得出特征的定义:从特征的内涵理解特征,即把产品当作一个整体,产品的所有特征都作为产品的属性,以描述产品的特点;从特征的外延理解特征,即把完整的需求集合中每一个相对独立的需求子集作为产品的特征,每个特征都作为一个整体。

特征模型[14-15]由一组特征和一组特征之间的关系组成。特征捕获软件产品的需求并将产品的所有需求当作一个实体集合,通过分析实体之间的共性和变化性对实体进行分类,将每一类实体抽象为一个特征。具体而言,特征通常具有特征名、绑定状态(BindState)等属性[5]。特征名用于引用特征;绑定状态用于刻画特征的状态,包括被绑定(bound)、被删除(removed)和待确定(undecided)。

对所有需求进行分类后,每一个特征都是需求的子集,两个需求子集中的实体之间会存在联系,将每一个联系作为一个有序对,对所有的有序对进行分类得到多个类别的有序对集合,将每一类有序对集合抽象为一个关系。

在数学研究中,人们常常使用点表示事物,使用点与点之间的连线表示事物之间的某种关系。特征模型基于这种图形化的思想,使用矩形表示特征,使用特征之间的连线表示特征之间的关系。图 1 给出一个特征模型实例。

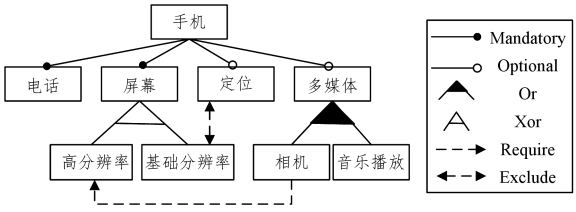


图 1 特征模型实例

Fig. 1 Example of feature model

特征模型定义了特征之间的两类关系[10],即精化关系(Refinement)和约束关系(Constraint)。精化关系表示父特征与子特征之间的关系,包括如下 4 种子关系:

1)Mandatory。子特征必须包含在其父特征出现的产品中。

2)Optional。子特征可以包含在其父特征出现的产品中,也可以不包含在其父特征出现的产品中。

3)Xor。父特征下有多个可选择的子特征,只有一个特征可以包含在其父特征出现的产品中。

4)Or。一个或多个子特征可以出现在其父特征出现的产品中。

精化关系将特征组织为树状的结构[14],但除了父特征与子特征之间的精化关系外,特征之间还存在约束关系,使得特征模型成为一种图状的结构[6]。约束关系包括如下 2 种子关系:

1)Requires。如果特征 a Require 特征 b,那么包含特征 a 的产品也必须包含特征 b。

2)Excludes。如果特征 a Excludes 特征 b,那么包含特征 a 的产品不能包含特征 b。

图 1 的特征模型表示一款手机产品必须包含打电话和屏幕两个特征,可以包含定位和多媒体两个特征,其中屏幕是高分辨率或基础分辨率,多媒体可以包含相机和音乐播放器中的一个或两个特征。如果一个手机产品中已经包含了定位特征,那么屏幕一定不能是基础分辨率,如果多媒体中具有相机特征,那么屏幕必须是高分辨率。

2.2 描述逻辑 $\mathcal{AL}$

描述逻辑(Description Logic,DL)[16]是一种表示知识的形式化语言,本文选用了描述逻辑语言 $\mathcal{AL}$。$\mathcal{AL}$ 的基本元素是概念(concepts)、角色(role)和个体(individual)。概念是对现实世界中具有相同特性的个体集合的抽象,而现实世界中的实例集合是概念的解释域。在解释域中,根据分类标准将具有相同特性的实例集合划分为子集合,每一个子集合是一个概念,使用符号 C 来标记;而每一个角色被解释为该解释域中类与类之间笛卡尔乘积的子集,使用符号 R 来标记。

为了解释类符号和角色符号的语义,描述逻辑定义了解释函数[17] $I:=(\Delta^I,\cdot^I)$,其中 Δ^I 其中是非空的解释域,$\cdot^I$ 是符号对应的语义。符号通过解释函数 I 映射到解释域 Δ^I 中,得到符号的语义 $\cdot^I$。使用类符号 C 表示类,在解释域 Δ^I 中一个具有相同特性的实例集合对应一个类的语义,类符号 C 映射到解释域 Δ^I 中对应集合 C^I 的过程即得到类符号语义

的过程。同理可得角色符号映射到角色语义的过程。

为了在计算机中表示描述逻辑语言中概念和角色的语义，采用数学集合的方法分别表示两者的语义，得到表 1 所列的语法及对应的语义[16]。

表 1　$\mathcal{AL}$ 的语法及语义

Table 1　Syntax and semantics of description logic $\mathcal{AL}$

语法	语义
$\top$	Δ^I
$\bot$	$\varnothing$
$C \sqcap D$	$C^I \cap D^I$
$\forall R.C$	$\{a \in \Delta^I \mid \forall b.(a,b) \in R^I \rightarrow b \in C^I\}$
$\exists R.\top$	$\{a \in \Delta^I \mid \exists b.(a,b) \in R^I\}$

表 1 中，$\top$ 表示顶域(universal concept)，$\bot$ 表示底域(bottom concept)，$\sqcap$ 表示合取(intersection)，$\forall$ 表示值约束(value restriction)，$\exists$ 表示存在量词(limited existential quantification)。

描述逻辑知识库[18]用于描述知识，它由两个构件组成，即术语断言(Terminological Box，TBox)和实例断言(Assertion Box，ABox)。TBox 引入应用领域的词汇表，ABox 包含命名个体的断言。TBox 通常以应用领域的词汇描述概念，一般包含蕴涵(inclusion)和等式(equality)两种形式。蕴涵表示为 $C \sqsubseteq D$ 或 $R \sqsubseteq S$，等式表示为 $C \equiv D$ 或 $R \equiv S$。ABox 通常以应用领域外延的词汇描述个体，一般包含概念断言和关系断言两种形式。概念断言表示为 $C(a)$，关系断言表示为 $R(a, b)$。

3　基于语义的特征模型重构

3.1　特征模型的语义表示

在对特征模型进行研究时，往往使用特征名引用特征。虽然可以假设建模团队在构建特征模型时是基于统一的建模词汇表，但这只能统一同一个团队的建模词汇。在实际的建模过程中，不同的建模团队使用不同的建模词汇表，导致不同建模团队建立的特征模型之间的建模词汇并不一致。若要将不同团队建立的特征模型进行复用，需要解决建模词汇不一致的问题。

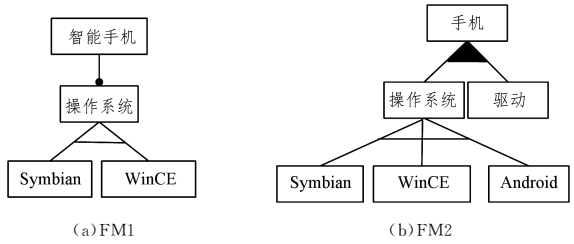
特征模型是一种图形化的表示方法，当基础特征模型通过其他特征模型提供的特征和约束进行重构时，由于缺乏精确的语义表示，导致无法精确定义重构特征模型的规则且无法定义自动推理规则以检验重构后的特征模型的一致性。为了解决这个问题，本文选择了描述逻辑语言 $\mathcal{AL}$，并给出了使用 $\mathcal{AL}$ 表示特征模型语义的方法。

3.1.1　基于语义的特征表示

对比特征的实质是对比特征的语义，但在通常情况下，不同的特征模型是由不同的建模团队建立，导致模型之间的建模词汇并不统一，因此无法直接通过特征名的字符串对比判断两个特征是否相同。为了解决该问题，本节提出一种基于语义的特征表示方法。

本文借鉴了描述逻辑中解释函数 $I:=(\Delta^I, \cdot^I)$ 的思想，其中，Δ^I 是非空解释域，即存在于客观世界的论域；$\cdot^I$ 是功能解释域，即特征的语义。每一个特征 *Feature* 通过解释函数 I 首先映射到论域 Δ^I，论域根据特征的分类标准将论域中的个体集合划分为多个子集，得到特征对应的语义 $Feature^I$。

图 2 给出了特征模型 FM1 和 FM2，两个模型由两个不同的建模团队建立，其中一个团队使用了特征名“智能手机”表示手机，另一个团队使用特征名“手机”表示手机，即出现了使用不同特征名表示同一个特征语义的情况。



(a)FM1　　(b)FM2

图 2　智能手机的特征模型

Fig. 2　Feature model of smartphone

由图 2 可知，FM1 对应的特征集合为{智能手机，操作系统，Symbian，WinCE}，FM2 对应的特征集合为{手机，操作系统，Symbian，WinCE，Android，驱动}。建模人员将以上两个特征集合存入计算机。计算机基于字符串对比的方法对特征进行自动对比，得到特征名不同的特征集合{智能手机，手机，驱动，Android}和特征名相同的特征集合{操作系统，Symbian，WinCE}。建模人员根据计算机自动对比的结果，对特征名不同的特征进行语义分析，得到客观世界中对应的产品集合，且使用点分别表示每一个产品，并为语义定义语义术语，其过程如图 3 所示。

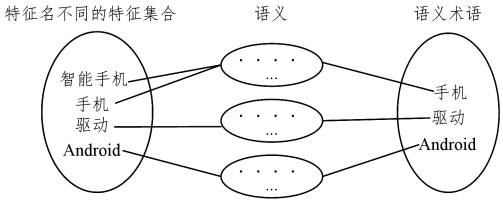


图 3　特征的语义表示

Fig. 3　Semantic representation of feature

因为存在使用不同特征名表示相同特征语义的情况，所以建模人员将特征名不同的特征集合中的每一个特征依次映射到论域中，得到每一个特征的解释域，即特征的语义。当得到特征名不同的所有特征的语义后，建模人员为每一个语义定义唯一的语义术语。基于复合函数的思想可得，每一个特征对应唯一的语义术语。

为特征名不同的特征定义语义术语之后，建模人员对特征名相同的特征集合进行分析，依次判断其中每一个特征在 FM1 和 FM2 中的语义是否相同，分析语义的过程与图 3 相同。

将每一个特征与其语义术语之间存在的映射关系进行表示，最终得到如表 2 所列的特征语义术语表。根据特征的语义术语表，使用语义术语替换特征名，该方法解决了在对多个特征模型进行复用时，特征模型之间建模词汇不一致的情况。

为了表示这一方法，对文献[5]的特征元模型进行扩展，得到图 4 所示的元模型。扩展的语义为：在复用多个特征模

型时，对具有相同语义的特征定义唯一的语义术语，通过特征名与语义术语的对应关系得到语义术语表。

表 2 特征的语义术语表

Table 2 Semantic terms of feature

特征名	特征的语义术语
智能手机	手机
操作系统	操作系统
Symbian	Symbian
WinCE	WinCE
手机	手机
Android	Android
驱动	驱动

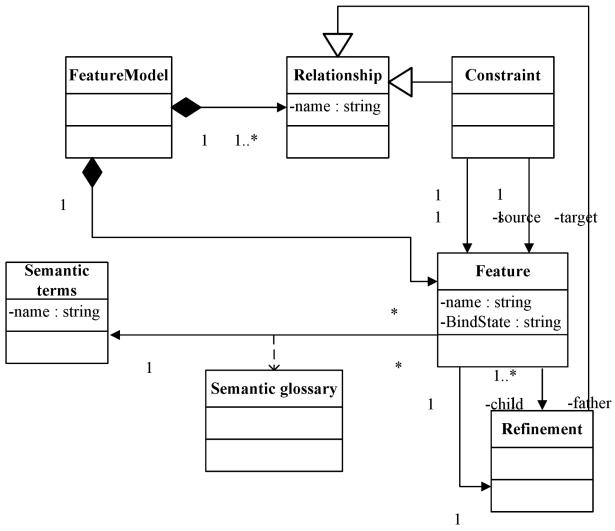


图 4 扩展的特征元模型

Fig. 4 Extended Feature meta-model

3.1.2 特征模型的语义表示

特征模型是一种对领域需求构建模型的图形化方法，在具有描述直观、易于理解等优点的同时缺少精确的语义表示，导致不能基于语义对特征模型定义重构规则，且不能进行自动推理以检验重构的特征模型的一致性。在 3.1.1 节，建模人员基于特征的语义，为特征定义对应的语义术语，并使用特征的语义术语引用待重用特征模型中的特征，本节将讨论如何使用描述逻辑语言对特征模型的语义进行表示。

定义 1(特征的语义表示) 使用描述逻辑的概念 *Feature* 表示特征模型中所有特征节点的集合，那么概念断言 $Feature(f_i)$中的实体 f_i 就对应特征模型节点集合中的某一个特征节点，其中 f_i 是特征的语义术语。

定义 2(关系的语义表示) 在特征模型中存在 6 类关系，使用描述逻辑的角色 *hasMandatory*，*hasOptional*，*hasOr*，*hasXor*，*hasRequires*，*hasExcludes* 分别表示特征模型的关系 *Mandatory*，*Optional*，*Or*，*Xor*，*Requires*，*Excludes*。同时，由于元模型的特征拥有属性绑定状态(BindState)，而在描述逻辑语言中将属性看作一种角色，因此使用角色 *hasState* 表示属性 BindState。

对于一个特征模型实例而言，由于描述逻辑的关系断言可以表示特征之间的关系，因此可以使用 $hasMandatory(f_i, f_j)$，$hasOptional(f_i, f_j)$，$hasXor(f_i, f_j)$，$hasOr(f_i, f_j)$表示精化关系，使用 $hasExcludes(f_i, f_j)$，$hasRequires(f_i, f_j)$表示约束关系，使用 $hasState(f_i$，bound/removed/undecided)表示特征的属性，其中 f_i，f_j 表示不同特征的语义术语。

当使用描述逻辑语言表示全部特征模型的语义后，根据特征模型关系的语义，为每一类关系定义推理规则，并将这些规则加入描述逻辑知识库中。对于 Xor 和 Or 关系，一个父特征下可以有多个子特征，但由于多个子特征可以分解为两个子特征的情况，为了描述简洁，本文只讨论两个子特征的情况，规则中的 f_i，f_j 和 f_k 表示不同特征的语义术语。

规则 1 hasMandatoryRule：

$\forall f_i \forall f_j\ Feature(f_i) \wedge Feature(f_j) \wedge hasMandatory(f_i, f_j) \wedge$ hasState(f_i，bound)$\rightarrow hasState(f_j$，bound)

规则 2 hasOptionalRule：

$\forall f_i \forall f_j\ Feature(f_i) \wedge Feature(f_j) \wedge hasOptional(f_i, f_j) \wedge$ hasState(f_i，bound)$\rightarrow hasState(f_j$，undecided)

规则 3 hasXorRule：

$\forall f_i \forall f_j \forall f_k\ Feature(f_i) \wedge Feature(f_j) \wedge Feature(f_k) \wedge hasXor(f_i, f_j) \wedge hasXor(f_i, f_k) \wedge hasState(f_i$，bound$) \wedge hasState(f_j$，bound$) \rightarrow hasState(f_k$，removed)

规则 4 hasOrRule：

$\forall f_i \forall f_j \forall f_k\ Feature(f_i) \wedge Feature(f_j) \wedge Feature(f_k) \wedge hasOr(f_i, f_j) \wedge hasOr(f_i, f_k) \wedge hasState(f_i$，bound$) \wedge hasState(f_j$，bound$) \rightarrow hasState(f_k$，undecided)

规则 5 hasRequiresRule：

$\forall f_i \forall f_j\ Feature(f_i) \wedge Feature(f_j) \wedge hasRequires(f_i, f_j) \wedge hasState(f_i$，bound$) \rightarrow hasState(f_j$，bound)

规则 6 hasExcludesRule：

$\forall f_i \forall f_j\ Feature(f_i) \wedge Feature(f_j) \wedge hasExcludes(f_i, f_j) \wedge hasState(f_i$，bound$) \rightarrow hasState(f_j$，removed)

3.2 精化关系重构

在同一个领域中存在多个特征模型，将其中一个特征模型作为基础特征模型，其他的模型都为基础特征模型提供需要重构的特征以及特征之间的关系。由于多个特征模型可以为基础特征模型提供需要的重构元素，其本质可以分解为一系列单个特征模型为基础特征模型提供需要的重构元素的过程，因此本文只讨论两个特征模型进行重构的情况。

首先将两个待复用的特征模型转换为描述逻辑知识库。设基础特征模型对应的描述逻辑知识库为 K_{base}，为基础特征模型提供重构元素的特征模型对应的描述逻辑知识库为 K_{provide}。若只考虑精化关系，则特征模型是一种树状结构，而再考虑约束关系，特征模型就是图的结构[11]。在特征模型中，精化关系表示父特征与子特征的关系，可以将其看作一种有序的关系。由于有向树在无向树的基础上考虑了节点之间的次序，因此使用有向树可以完整地表示模型中父特征与子特征的有序关系。其中，有向树中的节点表示特征模型的特征，有向树中的有向边表示特征模型的精化关系。

定义 3(有向树) 若使用有向树 D 表示描述逻辑知识库 K 中的特征和特征间的精化关系，则节点 v 对应知识库的概念断言 $Feature(f_i)$，将节点 v 命名为 f_i，即 $v.name = f_i$。有向树中节点与节点间的有向边对应知识库中的关系断言 $hasMandatory(f_i, f_j)$，$hasOptional(f_i, f_j)$，$hasXor(f_i, f_j)$

和 $hasOr(f_i, f_j)$。其中，有向边 e 的边名为关系断言中的关系名，即 $e.name = hasMandatory/hasOptioal/hasXor/hasOr$；有向边的方向为节点 f_i 指向节点 f_j。

定义 4（节点的对比）　给定两个需要对比的有向树 D_{base} 和 $D_{provide}$，设 v_b 和 v_p 分别是 D_{base} 和 $D_{provide}$ 中的任意节点，当且仅当 $v_b.name = v_p.name$ 时，$v_b = v_p$。设 e_b 和 e_p 分别是 D_{base} 和 $D_{provide}$ 中的有向边，只有当 $e_b.name = e_p.name$ 且边的父、子节点都各自相同时，$e_b = e_p$。

算法 1 为使用有向树对特征模型精化关系进行重构的算法。由于在特征模型中根特征的名称与当前领域的名称相同[5]，因此首先取出两个有向树的根节点 v_{rootb} 和 v_{rootp} 进行对比，只有当两棵有向树的根特征相等时，$D_{provide}$ 才可以为 D_{base} 提供重构的组件。然后基于先序遍历的思想，依次从有向树 $D_{provide}$ 和 D_{base} 中取出每一层的节点，分别存入 *nProvideNode* 和 *nBaseNode* 中，其中 n 表示根节点下的第 n 层。取出 n 值相等的节点集合 *nProvideNode* 和 *nBaseNode*，从 *nProvideNode* 中依次取出节点 v_p，判断 *nBaseNode* 中是否存在节点 v_b 与之相等：若两节点名相同，则根据表 3 的组合规则[11]，更改指向 v_b 的有向边的边名 e_b。若无节点 v_b 与 v_p 相等，找到节点 v_p 与其父节点之间的有向边 e_p，根据表 4 的插入规则[11]向 D_{base} 插入节点 v_p 和有向边 e_p。插入节点后，从 *nProvideNode* 中取出下一节点继续与 *nBaseNode* 对比。当 $D_{provide}$ 的最后一层节点对比完毕时，得到重构后的有向树 D'_{base}。

表 3　组合规则

Table 3　Rules of merging

Base	Provide			
	Mandatory	Optional	Xor	Or
Mandatory	Mandatory	Optional	Xor	Or
Optional	Optional	Optional	Optional	Optional
Xor	Or	Optional	Xor	Or
Or	Or	Optional	Or	Or

表 4　插入规则

Table 4　Rule of inserting

Base	Provide			
	Mandatory	Optional	Xor	Or
叶节点	Mandatory	Optional	Mandatory	Optional
Mandatory/Optional	Mandatory	Optional	Mandatory	Optional
Xor	Xor/Mandatory	Xor	Xor	Xor
Or	Mandatory	Or	Mandatory	Or

算法 1　精化关系重构的算法

输入：D_{base} 和 $D_{provide}$

输出：重构的有向树 D'_{base}

1. IF $v_{rootb}.name = v_{rootp}.name$
2. FOR EACH nProvideNode and nBaseNode
3.　FOR EACH v_p and v_b
4.　　IF $v_p.name = v_b.name$
5.　　find e_b, e_p;
6.　　according to The Rule of Merging, change the e_b;
7.　　ELSE
8.　　find e_p;
9.　　according to the Rule of Inserting insert v_p, e_p into D_{base};
10.　END IF
11.　END FOR
12.　END FOR
13. ELSE
14.　break;
15. END IF

待 $D_{provide}$ 中的所有节点都与 D_{base} 对比之后，将重构后的 D'_{base} 反向转换为描述逻辑语言中的概念断言和角色断言，并将其加入至 K_{base} 中得到重构了精化关系的 K'_{base}。

3.3　约束关系重构

得到重构了精化关系的描述逻辑知识库 K'_{base} 后，根据 $K_{provide}$ 中的约束关系，对 K'_{base} 进行重构。

定义 6　分别将 K'_{base}，$K_{provide}$ 中表示约束关系的两类角色断言 $Exclude(f_1, f_2)$ 和 $Require(f_1, f_2)$ 放入 $ConSet_{base}$ 和 $ConSet_{provide}$ 中。

算法 2 为根据 $K_{provide}$ 提供的约束关系，重构 K'_{base} 得到重构了约束关系 K''_{base} 的算法。首先从 $ConSet_{provide}$ 中依次取出角色断言并得到角色断言中的特征实例 f_1 和 f_2 组成的数对 c_p，在 $ConSet_{base}$ 中查找是否存在与特征实例 f_1 和 f_2 相同的数对 c_b。如果数对 c_b 与 c_p 相同，但其对应的角色名 r_b 和 r_p 不相同，则两个知识库存在冲突，将存在冲突的两个角色断言从各自的知识库中删除并以数对形式保存在需要修改的角色断言集合 *ChangeSet* 中；如果不存在数对 c_b 与 c_p 相同，则将 $K_{provide}$ 中对应的角色断言加入至 K'_{base} 中。当 $ConSet_{provide}$ 中的所有断言都与 $ConSet_{base}$ 中的断言进行判断，然后建模人员根据产品的实际情况，对 *ChangeSet* 中的角色断言集合进行判断，并选择正确的角色断言手动添加至 K'_{base} 中，最终得到重构后的 K''_{base}。

算法 2　约束关系重构的算法

输入：K'_{base} 和 $K_{provide}$

输出：重构了约束关系的 K_{base}''

1. FOR EACH c_p
2.　IF $c_b = c_p$
3.　　IF $r_b \neq r_p$
4.　　put $r_b(f_1, f_2)$ and $r_b(f_1, f_2)$ into ChangeSet;
5.　　ELSE
6.　　put $r_p(f_1, f_2)$ into $ConSet_{base}$;
7.　END IF
8. END FOR

3.4　特征模型的检验

选用描述逻辑语言的目的是精确表示特征模型的语义，同时，利用描述逻辑定义的 6 种关系的语义推理规则，可以自动推理得到特征模型的隐含知识，并根据隐含知识检验重构的特征模型的一致性。

在特征模型中，由于根特征必须是被绑定状态，因此在重构的特征模型中添加表示根特征的绑定状态是 bound 的角色断言。当根特征的状态确定后，由 3.1 节的推理规则分别推理出每一个特征的状态，并根据表 5 的状态保留规则，保留正确的特征状态。同时建模人员对存在冲突的状态人为进行修改。将修改后的知识库反向转换为特征模型，得到重构后的特征模型。

表 5 判断特征状态的规则

Table 5 Rule of judging feature's state

状态 1	状态 2		
	bound	undecided	removed
bound	bound	bound	冲突
undecided	bound	undecided	removed
removed	冲突	removed	removed

4 实例分析

4.1 实验准备

Protégé 是一种表示知识的工具，可以实现知识库的建立，虽然其本身不具有推理功能，但可以借助一些推理插件进行推理。SWRL(Semantic Web Rule Language)是一种结合 OWL 和 RuleML，用于定义规则的语言。SWRL Tab 是 Protégé 的一款插件，支持 SWRL 的编辑和推理。本文采用 Protégé 3.4.8 版本，并使用插件 SWAR Tab 对重构的特征模型进行推理验证，该方法可以分为以下几个步骤：

1)采用 Protégé 中的 OWLClasses 建立特征模型元模型中的类 Feature，FeatureSemantic 和 State，即特征模型形式化后对应的描述逻辑知识库 TBox 中的概念。

2)采用 Protégé 中的 Properties 建立特征模型元模型中的关系和类的属性，即特征模型形式化后对应的描述逻辑知识库 TBox 中的角色(见 2.1 节)，将每一个角色名作为 Properties 中的 object properties。

3)采用 Protégé 中的 Individuals 建立特征语义术语，即特征模型形式化后对应的描述逻辑知识库 ABox 中的概念断言，同时，建立类 State 中 bound，removed，undecided 3 种状态。

4)点击 FeatureSemantic 的每一个特征实例，根据重构的特征模型，在对应的属性下添加特征实例，即建立特征实例之间的二元关系(需要注意的是，由于 Xor 关系除了表示父特征与子特征的关系外，还表示子特征之间的互斥关系，因此若特征模型中存在 Xor 关系，则除了在其属性 *hasXor* 中添加子特征外，同时在子特征属性 *hasExclude* 中添加另外的子特征以表示子特征之间的互斥关系)。根特征的状态默认为 bound，则在根特征实例的属性 *hasState* 中添加状态 bound。

5)使用插件 SWAR Tab，按照 SWRL 的语法描述 2.1 节中的 6 个规则，得到如下 SWRL 形式的规则并进行自动推理，从而得到每一个特征的绑定状态。

SRWL 形式的规则 1 hasMandatoryRule：

$FeatureSemantic(?x) \wedge FeatureSemantic(?y) \wedge hasMandatory(?x,?y) \wedge hasState(?x, \text{bound}) \rightarrow \text{hasState}(?y, \text{bound})$

SRWL 形式的规则 2 hasOptionalRule：

$FeatureSemantic(?x) \wedge FeatureSemantic(?y) \wedge hasOptional(?x,?y) hasState \wedge (?x, bound) \rightarrow hasState(?y, \text{undecided})$

SRWL 形式的规则 3 hasOrRule：

$FeatureSemantic(?x) \wedge FeatureSemantic(?y) \wedge FeatureSemantic(?z) \wedge hasOr(?x,?y) \wedge hasOr(?x,?z) \rightarrow hasState(?y, undecided) \wedge hasState(?z, \text{undecided})$

SRWL 形式的规则 4 hasXorRule：

$FeatureSemantic(?x) \wedge FeatureSemantic(?y) \wedge FeatureSemantic(?z) \wedge hasXor(?x,?y) \wedge hasXor(?x,?z) \rightarrow hasState(?y, undecided) \wedge hasState(?z, \text{undecided})$

SRWL 形式的规则 5 hasRequiresRule：

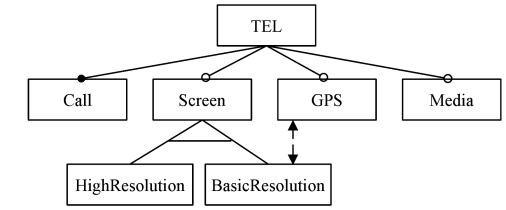
$FeatureSemantic(?x) \wedge FeatureSemantic(?y) \wedge hasRequires(?x,?y) \wedge hasState(?x, bound) \rightarrow hasState(?y, \text{bound})$

SRWL 形式的规则 6 hasExcludesRule：

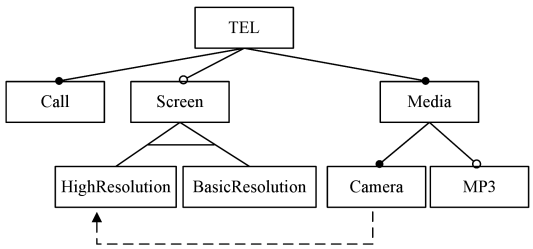
$FeatureSemantic(?x) \wedge FeatureSemantic(?y) \wedge hasExcludes(?x,?y) \wedge hasState(?x, bound) \rightarrow hasState(?y, \text{removed})$

4.2 重构手机特征模型案例

由于手机产品线特征模型是文献中常用的特征模型[10-11,19]，因此本文选取两个手机特征模型实例对提出的重构方法进行验证。给定两个手机特征模型，经过特征的语义分析后，使用语义术语替换原有特征名，得到待复用的特征模型 FM1 和 FM2，如图 5 所示。将特征模型 FM1 作为待重构的模型，特征模型 FM2 作为为 FM1 提供重构组件的模型，当两个特征模型转换为描述逻辑知识库时，FM1 与 FM2 分别对应描述逻辑知识库 K_{base} 和 $K_{provide}$。



(a)FM1



(b)FM2

图 5 待复用的两个特征模型

Fig. 5 Two feature models to be reused

将 K_{base} 和 $K_{provide}$ 中的精化关系分别转换为有向树 D_{base} 和 $D_{provide}$，根据算法 1 得到重构了精化关系的有向树 D'_{base}，如图 6 所示。

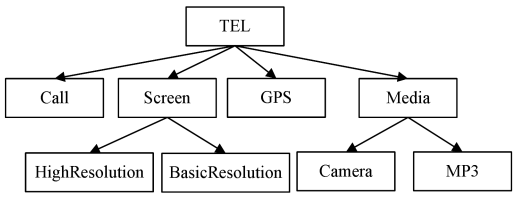


图 6 重构精化关系得到的有向树 D'_{base}

Fig. 6 Directed tree D'_{base} obtained by refactoring refinement relationships

将 D'_{base} 逆向转换为描述逻辑语言的概念断言和角色断

言，然后加入 K_{base} 得到重构精化关系后的描述逻辑知识库 K'_{base}。知识库 K'_{base} 的 ABox 为：

{*Feature*(Tel)，*Feature*(Call)，*Feature*(Screen)，*Feature*(HighResolution)，*Feature*(BasicResolution)，*Feature*(GPS)，*Feature*(Media)，*Feature*(Camera)，*Feature*(MP3)，*Mandatory*(Tel，Call)，*Optional*(Tel，Screen)，*Xor*(Screen，HighResolution)，*Xor*(Screen，BasicResolution)，*Mandatory*(Tel，Media)，*Mandatory*(Media，Camera)，*Optional*(Media，MP3)，*Excludes*(GPS，BasicResolution)}

根据算法 2，利用根据描述逻辑知识库 $K_{provide}$ 中的约束关系对 K'_{base} 中的约束关系进行重构，得到重构后的 K''_{base}。由于根特征的绑定状态默认为 bound，因此将根节点 *hasState*(*TEL*，bound)也加入 K''_{base} 中，最终得到了待自动推理的知识库 $K_{refactorinng}$。知识库 $K_{refactorinng}$ 的 *ABox* 为：

{*Feature*(Tel)，*Feature*(Call)，*Feature*(Screen)，*Feature*(HighResolution)，*Feature*(BasicResolution)，*Feature*(GPS)，*Feature*(Media)，*Feature*(Camera)，*Feature*(MP3)，*Mandatory*(Tel，Call)，*Optional*(Tel，Screen)，*Xor*(Screen，HighResolution)，*Xor*(Screen，BasicResolution)，*Mandatory*(Tel，Media)，*Mandatory*(Media，Camera)，*Optional*(Media，MP3)，*Excludes*(GPS，BasicResolution)，*Requires*(Camera，HighResolution)，*hasState*(TEL，bound)}

根据 3.1 节的 5 个步骤，在 Protégé 中建立知识库 $K_{refactorinng}$ 的 TBox 和 ABox，并使用插件 SWRL Tab 中的推理规则进行推理，由于在推理之前已经给定了根特征的绑定状态以及特征之间的关系，因此可得到图 7 所示的推理结果。

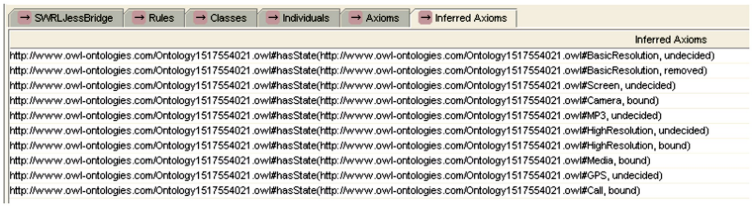


图 7　在 Protégé 中推理得到的结果

Fig. 7　Result of reasoning in Protégé

建模人员根据表 5 的状态保留规则，对每个特征保留一个状态，从而得到处理后的特征状态集合，即{(BasicResolution，removed)，(Screen，undecided)，(Camera，bound)，(MP3，undecided)，(HighResolution，bound)，(Media，bound)，(GPS，undecided)，(Call，bound)}。因为本案例无状态的冲突，所以省略处理冲突的步骤。将特征状态集合加入至知识库 $K_{refactorinng}$ 中，并将该知识库反向转换为特征模型，从而得到了重构的特征模型。

5　相关工作

软件产品线方法的研究热点之一是面向特征的软件复用技术，在该技术中，构建特征模型是一种重要且常用的技术。文献[5]介绍了面向特征的软件复用的基本概念、核心思想，以及相关的关键技术和重要成果。由于特征模型已经被广泛应用于多个领域，随着领域的日益复杂，模型中包含的特征和特征之间的关系达到成千上万的规模，构建这种大规模的特征模型是一项复杂的工作[20]，因此利用产品线中已有的特征模型对其进行重构的研究越来越重要。文献[3]根据软件产品线生产的产品集合，给出了 10 种特征模型需要重构的情况及重构的方法，然而由于产品集合中的产品众多，需要将所有产品与特征模型进行对比后才能判断该特征模型是否需要重构，因此这种方法会耗费大量的人力，且准确率会随着产品的增多而下降。文献[14]提出了一种多个特征模型自动融合为一个特征模型，再对融合得到的新模型进行人为重构的方法，该方法虽然解决了通过产品判断模型是否需要重构导致的效率低的问题，但存在无法匹配输入模型中相同或相似特征的问题。本文基于映射的思想，复用特征模型集合，提出建模人员可以将待重构的特征模型集合映射到现实世界中以得到特征的语义并对语义规定术语的方法，通过语义术语引用对应的特征，解决了文献[14]中无法直接匹配输入模型中相同或相似特征的问题。为了自动分析特征模型的一致性，文献[4]提出了将特征模型转换为本体语言的方法，并给出了使用本体语言定义的推理规则。本文利用本体语言中的描述逻辑语言，对特征模型进行形式化，并将待重构的特征模型集合中的任意一个模型作为基础模型，将其他模型都作为为基础模型提供重构的特征和关系的模型，从而提出了先重构精化关系再重构约束关系的算法，然后基于描述逻辑知识库给出了推理规则，根据该规则可推理出重构的特征模型中的隐含知识，建模人员可以根据隐含知识检查重构的特征模型的一致性。

结束语　本文提出一种基于语义的领域特征模型重构方法。在深入分析特征的定义后，基于映射的思想将特征映射到现实世界中以得到特征的语义并对语义规定术语。使用语义术语引用特征的方法，解决了由于建模词汇不一致导致的模型之间无法直接对比的问题。同时利用描述逻辑精确的语义表示对特征模型进行重构和推理，并根据推理结果验证模型的一致性。但是进行推理的过程中，由于特征模型的 Xor 关系包含父特征与子特征、子特征与子特征两类关系的语义，而本文定义的关于 Xor 的推理规则未表示子特征与子特征之间存在互斥的语义，因此在 Protégé 中建立特征与特征之间的关系时，将每一个 Xor 关系的子特征形成一个集合，在集合中的每一个特征的属性 *hasExclude* 下添加集合中除本身以外的其他特征，以补充推理规则中缺失的语义。这一步骤会增加建模人员的工作量，所以下一步的工作将继续研究推理的规则和工具，并对目前存在的问题进行改进和完善。

参 考 文 献

[1] IRSHAD M，PETERSEN K，POULDING S. A Systematic Literature Review of Software Requirements Reuse Approaches[J].

Information & Software Technology,2018,93:223-245.

[2] KARLSSON E A. Software reuse:a holistic approach[M]. John Wiley & Sons,Inc. 1995:129-144.

[3] TANHAEI M,HABIBI J,MIRIAN-HOSSEINABADI S H. A Feature Model Based Framework for Refactoring Software Product Line Architecture[J]. Journal of Computer Science and Technology,2016,31(5):951-986.

[4] MEGHA,GOEL S,KAUR K. Analyzing inconsistencies in software product lines using an ontological rule-based approach[J]. Journal of Systems & Software,2017,137:605-617.

[5] ZHANG W,MEI H. Feature-oriented software reuse technology-state of the art[J]. Chinese Science Bull,2014,59(11):21-42. (in Chinese)
张伟,梅宏. 面向特征的软件复用技术——发展与现状[J]. 科学通报,2014,59(1):21-42.

[6] DAVIS A M. The Design of a Family of Application-Oriented Requirements Languages[J]. Computer,1982,15(5):21-28.

[7] KANG K C,COHEN S G,HESS J A,et al. Feature-oriented domain analysis (foda) feasibility study[J]. Georgetown University,1990,4(4):206-207.

[8] ACHER M,BAUDRY B,NASR S B. Breathing ontological knowledge into feature model synthesis:an empirical study[J]. Empirical Software Engineering,2016,21(4):1794-1841.

[9] MORITANI B I,LEE J. An approach for managing a distributed feature model to evolve self-adaptive dynamic software product lines[C]// International Systems and Software Product Line Conference. 2017:107-110.

[10] TANHAEI M,HABIBI J,MIRIAN-HOSSEINABADI S H. Automating Feature Model Refactoring:A Model Transformation Approach[J]. Information & Software Technology,2016,80(C):138-157.

[11] NIE K M,ZHANG L. A Software product line domain requirement model construction method based on model difference and model composition[J]. Chinese Journal of Computers,2014,37(3):539-550. (in Chinese)
聂坤明,张莉. 基于模型对比和组合的软件产品线领域需求建模[J]. 计算机学报,2014,37(3):539-550.

[12] SHEN G H,ZHANG W,HUANG Z Q,et al. Description-logic-based feature modeling and verification[J]. Journal of Computer Research and Development,2013,50(7):1501-1512. (in Chinese)
沈国华,张伟,黄志球,等. 基于描述逻辑的特征语义建模及验证[J]. 计算机研究与发展,2013,50(7):1501-1512.

[13] BÜRDEK J,KEHRER T,LOCHAUM,et al. Reasoning about product-line evolution using complex feature model differences[J]. Automated Software Engineering,2016,23(4):687-733.

[14] YI L,ZHAO H Y,ZHANG W,et al. Research on the merging of feature models[J]. Chinese Journal of Computers,2013,36(1):1-9. (in Chinese)
易立,赵海燕,张伟,等. 特征模型融合研究[J]. 计算机学报,2013,36(1):1-9

[15] USMAN M,IQBAL M Z,KHAN M U. A Product-line Model-driven Engineering Approach for Generating Feature-based Mobile Applications[J]. Journal of Systems & Software,2016,123:1-32.

[16] BAADER F,CALVANESE D,MCGUINNESS D L,et al. The Description Logic Handbook:Theory,Implementation and Applications[J]. Kybemetes,2003,32(9-10):43-95.

[17] CALVANESE D. Unrestricted and Finite Model Reasoning in Class-Based Representation Formalisms[J]. AI Communications,1996,9(4):225-226.

[18] ARENAS M,BOTOEVA E,CALVANESE D,et al. Knowledge base exchange:The case of OWL 2 QL[J]. Artificial Intelligence,2016,238:11-62.

[19] LIAN X L,ZHANG L. Multi-Objective Optimization Algorithm for Feature Selection in Software Product Lines[J]. Journal of Sofeware,2017,28(10):2548-2563. (in Chinese)
连小利,张莉. 面向软件产品线中特征选择的多目标优化算法[J]. 软件学报,2017,28(10):2548-2563.

[20] BATORY D,BENAVIDES D,RUIZCORTES A. Automated analysis of feature models:challenges ahead[J]. Communications of the ACM,2006,49(12):45-47.