软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2016,27(5):1212−1229 [doi: 10.13328/j.cnki.jos.004829]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

一种软件特征模型扩展和演化分析方法*

胡 洁[1,3], 王 青[1,2]

[1](中国科学院 软件研究所 互联网软件技术实验室,北京 100190)

[2](计算机科学国家重点实验室(中国科学院 软件研究所),北京 100190)

[3](中国科学院大学,北京 100049)

通讯作者: 胡洁, E-mail: hujie@itechs.iscas.ac.cn

摘 要: 特征模型是面向特征的软件开发过程的重要概念和制品,该模型以特征为单位,刻画了领域产品的共性和可变性.在日趋频繁的软件演化过程中,保持特征模型的一致演化,对于支持高效的复用开发和按需配置至关重要.目前,大多数的研究是在需求层面进行共性和可变性分析,对特征模型的研究则集中在对共性和可变性的建模上.但是,由于特征变更在建模过程中存在"涟漪"效应,会导致新的共性和可变性演化.现有的分析方法还无法解决这个问题,会导致丢失一些潜在的产品共性,从而影响复用的效率.提出了一种特征模型扩展和演化分析方法.通过扩展特征关联关系和模型演化元操作,实现对特征变更"涟漪"效应的分析.发现潜在的产品共性,提出重构策略和半自动化的共性提取和特征重构支持方法.该方法还针对典型的配置冲突提出了冲突消解规则和策略.最后,通过案例分析验证了该方法的可用性和有效性.

关键词: 软件演化;模型演化;软件定制;共性;可变性;特征模型

中图法分类号: TP311

中文引用格式: 胡洁,王青.一种软件特征模型扩展和演化分析方法.软件学报,2016,27(5):1212−1229. http://www.jos.org.cn/1000-9825/4829.htm

英文引用格式: Hu J, Wang Q. Extensions and evolution analysis method for software feature models. Ruan Jian Xue Bao/Journal of Software, 2016,27(5):1212−1229 (in Chinese). http://www.jos.org.cn/1000-9825/4829.htm

Extensions and Evolution Analysis Method for Software Feature Models

HU Jie[1,3], WANG Qing[1,2]

[1](Laboratory for Internet Software Technologies, Institute of Software, The Chinese Academy of Sciences, Beijing 100190, China)

[2](State Key Laboratory of Computer Science (Institute of Software, The Chinese Academy of Sciences), Beijing 100190, China)

[3](University of Chinese Academy of Sciences, Beijing 100049, China)

Abstract: Feature model is an essential concept and artifact in feature oriented software development (FOSD). It depicts commonality and variability (C&V) of products in terms of features. With increasingly frequent software evolution, keeping the feature model in consistent with the evolution is very important. Most of the related researches usually analyze the C&V on the requirement level, and modeling the analyzed C&V by the feature model. However, since the feature changes may cause the ripple effect during the modeling process, some new commonalities and variability may be derived. The current researches are still not able to resolve this problem, which leads to some potential overlooking commonalities and inefficiency in reuse. This paper proposes an approach to extend the feature model and analyze the software evolution based on the feature model. The extensions of feature dependency and evolution meta-operators can support the ripple effect analysis of the feature changes, as well as the exploration of the potential commonalities. The new approach also develops some refactoring strategies and a semi-automated tool to support commonality extraction and feature refactoring. In addition,

* 基金项目: 国家自然科学基金(91318301, 91218302, 61432001); 国家科技重大专项(2012ZX01039-004)

Foundation item: National Natural Science Foundation of China (91318301, 91218302, 61432001); National Science and Technology Major Project of the Ministry of Science and Technology of China (2012ZX01039-004)

收稿时间: 2014-11-01; 修改时间: 2015-01-06; 采用时间: 2015-03-09

rules and strategies are designed to resolve typical configuration conflicts. Finally, the paper employs a case study to validate the applicability and effectiveness of the presented method.

Key words: software evolution; model evolution; software customization; commonality; variability; feature model

面向特征的软件开发(feature oriented software development,简称 FOSD)[1,2]是一种面向大规模定制化软件开发的过程方法,其基本思想是:以复用为核心,将软件特征作为复用的基本单元,通过特征组合生成满足特定业务场景的软件产品.由于 FOSD 为大规模定制化软件的开发提供了一种有效的过程方法,近年来受到研究人员和工业实践的广泛关注[2].

特征模型(feature model)是 FOSD 的核心概念和制品.该模型在特征基础上描述了领域内产品的共性和可变性(commonality and variability)[3].特征模型主要为 FOSD 提供两个方面的支持:一方面支持领域可复用资产的识别和开发[3];另一方面支持特征模型的配置,根据定制产品的需求选择合适的特征[4].一个设计良好的特征模型,可有效提升定制产品的开发效率和质量.

随着用户需求和产品特征的不断变更,特征模型也持续演化.当前,分析特征模型演化仍然面临诸多挑战,这些挑战主要源自以下几个方面:

(1) 特征变更的“涟漪”效应(ripple effect).

特征并不是孤立存在的,它们在设计或者实现等方面都存在相关性.这些相关性使得单一特征的变更可能会传播至其他特征,形成“涟漪”效应[5,6].由于特征模型面向重用开发,特征间的关联关系更加多样,使得其中的变更传播也更为复杂;

(2) 共性和可变性的变化.

共性和可变性是特征模型的本质,其变化也是特征模型演化的重要组成.共性和可变性设计的一个基本原则是,尽可能地识别产品共性,满足高效的复用开发[7,8].因此,当发生特征变更时,不仅需要考虑是否有新的特征可作为产品共性,还需考虑对特征进行重构挖掘其中的共性成分;

(3) 特征模型配置的变化.

特征变更或者共性和可变性的变化都可改变特征模型的配置.由于特征模型面向不同的定制产品,更新特征模型配置应兼顾不同产品的演化需求,避免破坏相关产品.此外,为了生成有效的定制产品,特征模型的配置还应保持一致[9].由于任意模型操作都有可能引发配置冲突[10],因此,在分析特征模型配置变化时,还需识别其中的冲突并进行消解.

特征模型演化分析一直是 FOSD 研究的重点,但是已有的研究工作多以需求为基础.这些方法通常基于需求模型或者需求规约分析特征的变更,随后将分析结果迁移到特征模型中,并指明共性和可变性的变化[11−13].其中的共性和可变性分析仅仅在需求层面进行,特征模型只是用来描述这些变化.而通过特征模型表达共性和可变性变化时,变更特征引发的“涟漪”效应可导致更多的共性和可变性变化.由于需求模型和需求规约缺乏对特征和特征关联关系的描述[14],从而无法实现对特征变更“涟漪”效应的分析,进而无法发现一些潜在的产品共性.除了上述方法以外,还有一些研究工作关注演化分析的某一个方面,例如诊断特征模型的异常或者错误[10,15,16].这些方法或只能检测模型的异常和错误,或只给出简单的解决方案,不能满足模型演化分析的要求.

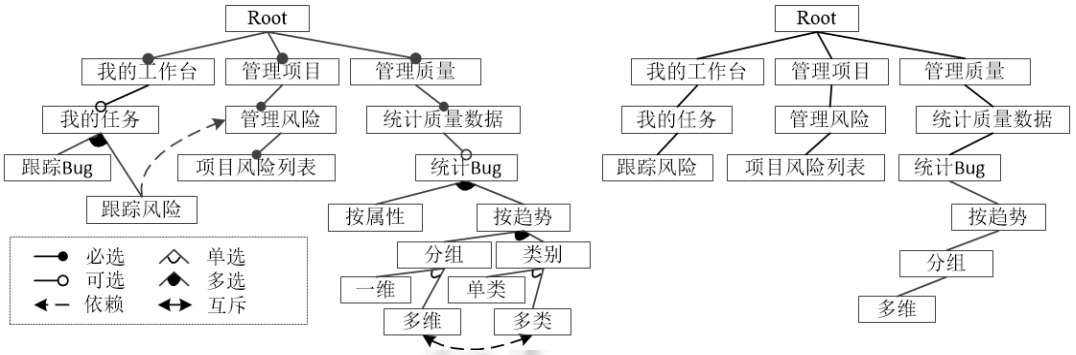
基于上述挑战和现有研究的不足,本文提出一种特征模型扩展和演化分析方法,根据定制产品的特征变更请求,分析、识别和演化特征模型.该方法扩展了特征关联关系和特征模型演化元操作(meta-operator),通过这些扩展,实现了特征变更“涟漪”效应的分析,定位特征变更对特征模型造成的影响;发现潜在的产品共性,建立策略和半自动化的分析工具,支持共性提取和特征重构;同时,该方法还针对典型的模型配置冲突建立规则和策略,消解出现的配置冲突,维护了模型配置的一致性.为了验证该方法的可用性和有效性,我们选取一个具有代表性的软件系统进行案例分析.分析结果表明,本文所提出的方法可以用于实际项目中辅助项目人员有效分析和识别特征模型演化.

本文第 1 节介绍特征模型.第 2 节对特征模型进行扩展.第 3 节详细描述特征模型演化分析的过程.第 4 节是案例分析.第 5 节讨论方法的局限性以及案例分析得到的经验.第 6 节介绍本文的相关工作.第 7 节对本文进

行总结.

1 特征模型

特征通常指从用户角度观察到的软件特性,其满足一个或者多个用户需求[3,17].Kang 等人首次将特征引入到了软件产品线工程(software product line engineering)中,通过特征建模,分析和描述领域的共性和可变性[3].特征模型包含层次化的特征,并提供不同类型的关联关系约束特征的选择.图 1 显示了一个软件过程管理系统(简称 Qone)的特征模型片段和对该片段进行配置得到的定制产品片段.



(a) 特征模型片段 (b) 定制产品片段

Fig.1 An example of feature model and custom product

图 1 特征模型和定制产品示例

1.1 特征关联关系

在特征模型中,根据其作用的不同,可将特征关联关系划分为两类:父子约束关系和跨树约束(cross-tree constraint)关系[18].父子约束关系描述了父子特征的配置,即,当父特征被选择时,如何选择子特征.该类型的关联关系一般包括 4 种类型:必选(mandatory)、可选(optional)、单选(alternative)和多选(or).跨树约束关系则约束了非父子特征之间的配置,一般包括依赖(imply)和互斥(exclude)关系.本文将这些类型的关联关系称为配置约束(configuration constraint)关系.表 1 详细描述了每种类型的配置约束关系,并根据图 1 给出了例子.

Table 1 Different types of configuration constraints

表 1 不同类型的配置约束关系

类别	类型	配置约束	示例
父子约束	必选	当 f 的父特征被选择时,f 必须被选择,即,f 为必选特征	“管理项目”
	可选	当 f 的父特征被选择时,f 可被选择或者不选,即,f 为可选特征	“统计 Bug”
	单选	在一个单选特征组中,只有一个特征可以被选择	“一维”,“多维”
	多选	在一个多选特征组中,可有多个特征同时被选择	“跟踪 Bug”和“跟踪风险”
跨树约束	依赖	当 f 被选择时,其所依赖的特征也必须被选择	“跟踪风险”和“管理风险”
	互斥	当 f 被选择时,与其互斥的特征不能被同时选择	“多类”和“多维”

1.2 特征配置约束的一致性

在特征模型中,一个特征的选择可能受到多种配置约束关系的制约.错误或者不谨慎的配置约束关系设置可引发配置冲突或者违背特征既有的配置约束.例如,若两个特征间同时存在依赖和互斥关系,可导致分析人员无法对这两个特征进行选择.针对特征模型中可能存在的配置约束不一致问题,Maben 和 Guo 等人总结了多种典型的配置冲突场景[9,10].我们综合这些工作,将冲突场景归纳为 5 种类型.表 2 描述了每种类型及其相应的后果,其中的完全必选(full mandatory)特征指从该特征自身到根特征(root)路径上的每个特征都为必选特征[9].

Table 2 Five types of configuration conflicts

表 2 5 种不同类型的配置冲突

类型	描述	后果
Type I	一个完全必选特征依赖一个非完全必选特征	非完全必选特征必须在每个产品中被选择
Type II	一个完全必选特征与一个非完全必选特征互斥	非完全必选特征无法被任何一个产品选择
Type III	两个完全必选特征之间存在互斥关系	无法在两个完全必选特征中进行选择
Type IV	同一单选特征组中的特征存在依赖关系	无法在两个特征中进行选择
Type V	两个特征之间同时存在依赖和互斥关系	无法在两个特征中进行选择

图 2 显示了每种冲突场景的例子.在图 2(a)中,由于被完全必选特征 f_1 依赖,可选特征 f_2 将被强制包含在每个产品中,从而违背了该特征与父特征间的可选关系.在图 2(b)中,由于与 f_1 互斥,f_3 无法被任何产品选中.这样的特征也称为死(dead)特征[18].在其余 3 种配置冲突例子中,冲突的配置约束使得无法在两个特征中做出选择.

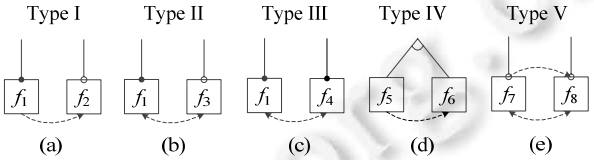


Fig.2 Examples of configuration conflicts

图 2 配置冲突示例

2 扩展特征模型

特征模型的演化分析面临诸多挑战,而在前述特征模型基础上解决这些挑战,可能会导致某些重要的特征模型变化被忽略,尤其是在分析特征变更的“涟漪”效应时.本节针对该不足,补充了 3 种特征关联关系,并对特征模型演化元操作进行相应的扩展.

2.1 扩展特征关联关系

在特征变更的“涟漪”效应中,变更的传播主要通过特征之间的关联关系.识别不同层次、不同类型关联关系上的特征变更传播,是分析“涟漪”效应的关键.但是当前,特征模型只定义了与特征配置相关的关联关系,并未说明特征在设计等其他方面的相关性.仅从这些关联关系分析,可能导致“涟漪”效应分析存在缺失.例如,相似的特征之间也可能存在变更传播,即,当特征 f_i 被修改时,与 f_i 相似的特征也可能被修改.由于特征模型缺乏对此类关联关系的表达,从而无法在模型中识别该类的变更传播.

虽然已有很多研究工作对特征关联关系进行了扩展[19−21],例如描述特征操作时或者运行时的相关性,但这些扩展仍然关注于特征配置,是对特征配置约束关系的细化,无法为“涟漪”效应分析提供更多可用的关联关系.

在分析特征关联关系时,我们受到需求关联关系相关研究的启发.由于特征是需求在系统功能层次上的表达,需求关联关系也可用于特征之间[20].例如,相似关系描述完成任务相似的需求,也可用于关联功能相似的特征.在需求关联关系的研究中,有两个具有代表性的模型:Dahlstedt 需求关联关系模型[22]和 Pohl 需求关联关系模型[23].但是,并不是所有类型的关联关系都可用于分析“涟漪”效应.根据 Zhang 等人对这些模型中的关联关系调查和分析的结果[24],有 3 种类型的关联关系——前置(precedes)、限制(constrains)和相似(similar_to)常用于实际项目中“涟漪”效应的分析.基于该成果,我们将这 3 种关联关系补充到特征模型中,并重新定义:

- 前置:描述特征在业务流程中的前后关系;
- 限制:描述了一个特征约束另一个特征的功能或行为;
- 相似:描述一组功能相似或者使用相同数据的特征.

为与配置约束关系区分,我们为扩展的关联关系设计了新的符号,如图 3 所示,在图 3(a)中,数据备份前需要设置系统采用的数据库,因此“设置数据库”前置于“备份数据”.在图 3(b)中,由于只有系统授权的角色才能创建

项目,因此“创建项目”受限于“设置角色和权限”;在图 3(c)中,“我的风险列表”、“项目风险列表”和“风险视图”为一组相似的特征,它们都通过列表的形式展示风险.

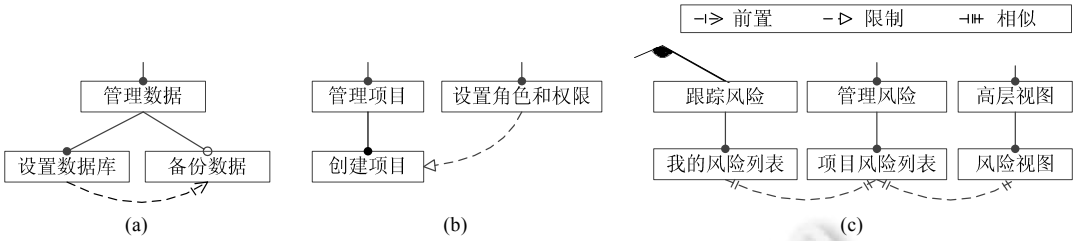


Fig.3 Examples of extended feature dependencies

图 3 扩展特征关联关系示例 我的风险

2.2 扩展特征模型演化元操作

在演化过程中,特征模型的变化通常可通过一个或一组模型元操作表达.根据前面补充的 3 种关联关系,我们在现有研究的基础上[10,25,26]对特征模型演化元操作进行了扩展.表 3 显示了扩展后的特征模型演化元操作.

Table 3 Meta-Operators of feature models after extending feature dependencies

表 3 扩展特征关联关系后的特征模型演化元操作

关联关系	元操作		
	插入	删除	移动
必选	$InsMan(f_i,f_j)$	$DelMan(f_i,f_j)$	$MovToMan(f_i,f_j)$
可选	$InsOpt(f_i,f_j)$	$DelOpt(f_i,f_j)$	$MovToOpt(f_i,f_j)$
单选	$InsAlt(f_i,f_j)$	$DelAlt(f_i,f_j)$	$MovToAlt(f_i,f_j)$
多选	$InsOr(f_i,f_j)$	$DelOr(f_i,f_j)$	$MovToOr(f_i,f_j)$
依赖	$InsImpl(f_i,f_j)$	$DelImpl(f_i,f_j)$	–
互斥	$InsExcl(f_i,f_j)$	$DelExcl(f_i,f_j)$	–
前置	$InsPred(f_i,f_j)$	$DelPred(f_i,f_j)$	–
局限	$InsCons(f_i,f_j)$	$DelCons(f_i,f_j)$	–
相似	$InsSim(f_i,f_j)$	$DelSim(f_i,f_j)$	–

表 3 从操作类型和特征关联关系类型两个维度定义了 22 个演化元操作,每个元操作由操作名称和目标特征构成.在配置约束关系相关的操作中:父子约束关系的插入和删除操作同时包括特征的插入和删除以及父子配置约束关系的改变,例如,$InsOpt(f_i,f_j)$表示为 f_j 插入新的可选子特征 f_i;跨树约束关系相关操作关注特征间依赖或者互斥关系的改变,例如,$InsImpl(f_i,f_j)$表示在 f_i 和 f_j 之间插入依赖关系,使得 f_i 依赖 f_j;扩展关联关系相关操作关注特征间前置、限制或者相似关系的改变,例如,$InsSim(f_i,f_j)$表示在 f_i 和 f_j 之间插入相似关系.

3 基于特征模型的演化分析

在扩展特征关联关系和演化元操作的基础上,我们设计了一个流程,实现基于特征模型的演化分析,图 4 显示了该流程的主要阶段和活动.该过程以特征变更请求集合为输入,这些特征变更请求来自于不同的产品,其定义如下:

定义 1(特征变更请求). 一个特征变更请求(记作 *cr*)是一个四元组:

$$cr=\langle f,f',changType,product\rangle.$$

其中,

- f 表示变更前的特征.当变更类型为“添加”时,f 为请求添加特征的父特征.
- f' 表示变更后的特征.当变更类型为“删除”时,f' 为请求删除特征的父特征.
- $changeType$={“添加”,“修改”,“删除”}表示特征变更的类型.

- *product* 表示提出该变更的产品.由于变更可能由多个产品同时提出,该元素为产品集合.

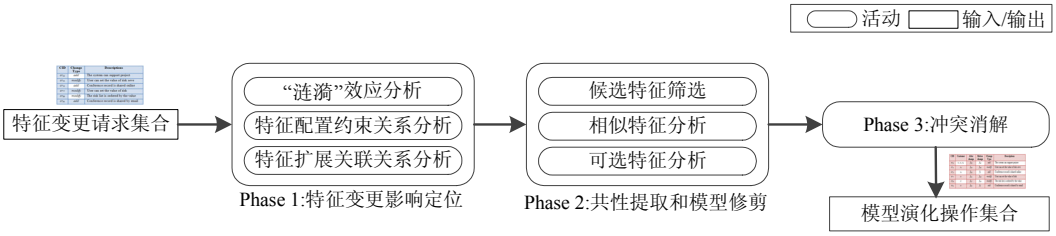


Fig.4 Overview of the evolution analysis based on the feature model

图 4 基于特征模型的演化分析过程

由于特征的添加、修改和删除总是伴随着特征关联关系的改变,因此,特征变更请求隐含变更特征与其他特征关联关系的变化.实际上,特征间关联关系的改变也可单独作为一类变更请求提出.而满足关联关系的变更,也必然导致特征的修改.例如,移除特征 f_i 和 f_j 之间的互斥关系,这两个特征之中至少一个需被修改.由于本文重点从特征的添加、修改和删除研究特征模型演化,而对特征关联关系的变更如何改变特征模型,则不在本文的研究范围内.

整个分析过程包含为 3 个阶段:特征变更影响定位、共性提取和模型修剪以及冲突消解.特征变更影响定位阶段分析输入特征变更对特征模型造成的影响,包括识别“涟漪”效应引发的其他特征变更,根据特征变更分析特征配置约束关系和扩展关联关系的变化;共性提取和模型修剪阶段在定位分析的基础上对插入的可选特征和形成的相似特征进行共性分析,并根据分析结果修剪特征模型;冲突消解阶段在发生配置约束变化的特征间进行配置冲突的检测和消解.分析过程结束时,输出特征模型演化操作集合.根据该集合可得到演化后的特征模型.

3.1 特征变更影响定位

本阶段的目标是分析输入特征变更对模型造成的影响,并在特征模型中进行定位,包括分析这些变更引发的“涟漪”效应、根据特征变更分析特征配置约束关系和扩展关联关系的变化.

3.1.1 “涟漪”效应分析

分析“涟漪”效应时,主要利用扩展的前置、限制和相似这 3 种关联关系.根据这些类型的关联关系,我们设计了规则识别受“涟漪”效应影响的特征.

规则 1. 如果模型中的特征 f 符合下列条件之一,其可能受到“涟漪”效应影响发生变更.

- 当 f 的前置特征发生变更时,即,为 f 添加新的前置特征或者修改前置特征;
- 当限制 f 的特征发生变更时,即,为 f 添加新的限制特征,修改或者删除限制特征;
- 当与 f 相似的特征被修改时.

根据特征变更请求的定义,除识别受影响的特征外,还需确定这些特征的变更类型.在“涟漪”效应的相关研究中,多数工作都只识别受到影响的实体[27−30],并未进一步探讨实体变更的类型.通过追溯“涟漪”效应发生的过程我们发现,受影响特征的变更类型与两个因素相关:一是影响该特征的特征变更类型,二是识别该特征所依据的关联关系类型.综合这两个因素,可得到变更类型决策矩阵,见表 4.

Table 4 Decision matrix of change types of the affected features

表 4 受影响特征变更类型的决策矩阵

	添加	修改	删除
前置	修改	修改	删除
限制	修改	修改	修改
相似	–	修改	–

利用该矩阵,可形成如下规则:

规则 2. 对于通过规则 1 识别的受影响特征 *f*,其变更 *cr* 的类型可通过如下规则定义:

- 如果 *f* 受添加类型变更的影响,并通过前置或者限制关系识别,则 *cr* 的类型为修改;
- 如果 *f* 受修改类型变更的影响,并通过前置、限制或者相似关系识别,则 *cr* 的类型为修改;
- 如果 *f* 受删除类型变更的影响,并通过前置关系识别,则 *cr* 的类型为删除;
- 如果 *f* 受删除类型变更的影响,并通过限制关系识别,则 *cr* 的类型为修改.

根据规则 1 和规则 2 分析后,可将分析结果提交给项目分析人员确认,保证结果的准确性.图 5 显示了两个“涟漪”效应分析的例子:图 5(a)中的特征来自于图 3(c),当“项目风险列表”被修改时,与之相似的两个特征“我的风险列表”和“风险视图”也需被修改;图 5(b)中,“定时任务”前置于“定时任务消息”,当删除“定时任务”时,“定时任务消息”也需被删除.

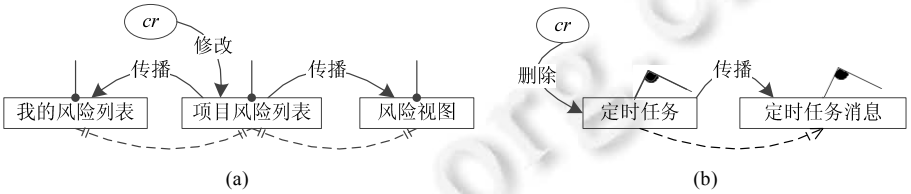


Fig.5 Examples of analyzing the ripple effect of the feature changes

图 5 分析特征变更“涟漪”效应示例

3.1.2 特征配置约束关系分析

特征的变更,改变了特征模型的配置.由于特征变更请求来自于不同的定制产品,直接根据这些请求进行模型操作,可能会破坏其他产品所需的特征配置.例如,删除特征“定时任务”由某一产品提出,如果直接从模型中删除该特征,其他需要该特征的产品则无法得到满足.为了避免此类现象的发生,我们根据不同类型的特征变更定义了配置更新策略,从而兼顾不同产品对特征模型的配置要求.

- 策略 1:当 *cr* 的类型为添加时,直接将请求添加的特征插入到特征模型中,定义配置约束关系.
- 策略 2:当 *cr* 的类型为修改时,将变更后特征插入到特征模型中,同时保留变更前特征.通过定义或者更新配置约束关系,满足不同产品对二者的需求.
- 策略 3:当 *cr* 的类型为删除时,更新请求删除特征的配置约束关系,而不是直接从模型中移除该特征.

根据不同的更新策略,我们进一步根据特征变更的上下文设计了详细的规则为变更涉及特征定义或者更新配置约束关系.图 6 通过图形化的方式描述了该规则.

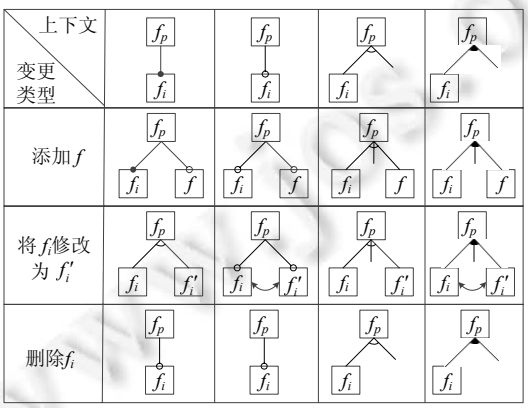


Fig.6 Rules for updating the configuration constraints of the features

图 6 特征配置约束关系更新规则

规则 3. 对于一个特征变更请求 *cr*,变更前特征 *f* 和变更后特征 *f*′的配置约束关系可通过如下规则定义或者更新:

- 当 *cr* 的类型为添加时,如果 *f* 的子特征不为单选或者多选特征组,则可将 *f*′插入为 *f* 的可选子特征;反之,将 *f*′直接插入到特征组中.
- 当 *cr* 的类型为修改时,可将 *f*′插入到 *f* 的邻居节点:
  1) 如果 *f* 为必选特征,则 *f* 和 *f*′被定义为一个单选特征组;
  2) 如果 *f* 为可选特征,则 *f*′被定义为可选特征并与 *f* 互斥;
  3) 如果 *f* 为单选特征组成员,则 *f*′插入到同一特征组;
  4) 如果 *f* 为多选特征组成员,则 *f*′插入到同一特征组并与 *f* 互斥.
- 当 *cr* 的类型为删除时,当且仅当 *f* 为必选特征时,其被修改为可选特征.

在该规则中,考虑到特征变更请求来自不同的定制产品,插入特征的配置约束默认定义为非必选特征.此外,对于修改类型的变更,如果 *f* 为可选特征或者多选特征组成员,则 *f* 与 *f*′互斥.这样的设计可避免 *f* 和 *f*′同时出现产品中.图 7 显示了一个定义和更新特征配置约束关系的例子,该例源于图 5(a)所示的特征变更,f_{mrl},f_{prl} 和 f_{rv} 分别为变更“我的风险列表”、“项目风险列表”和“风险视图”得到的特征.根据规则 3,可将它们插入到变更前特征的邻居节点.由于 3 个变更前特征均为必选特征,更新配置约束关系后,其分别与 f_{mrl},f_{prl} 和 f_{rv} 形成单选特征组.

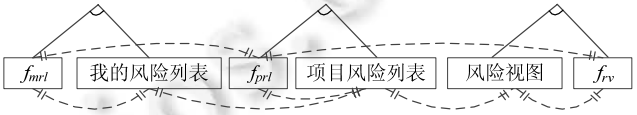


Fig.7 An example of updating the configuration constraints of the features

图 7 定义和更新特征配置约束关系示例

3.1.3 特征扩展关联关系分析

除了定义和更新变更涉及特征的配置约束关系以外,还需对这些特征的前置、限制和相似关系的变化进行分析.其中,特征变更请求隐含的特征间前置、限制和相似关系的改变随特征插入在特征模型中直接定义.此外,由于修改类型变更中的变更后特征与变更前特征功能相似,可认为二者之间存在相似关系,如图 7 中的 f_{mrl} 和“我的风险列表”.类似的,一组相似特征变更后的特征之间也可存在相似关系,如图 7 中的 f_{mrl},f_{prl} 和 f_{rv}.因此,在定位过程中还需在模型中为这些特征建立相似关系.

3.2 共性提取和模型修剪

保持良好的共性和可变性设计,是 FOSD 高效复用的前提条件,也是特征模型演化分析的关键.由于不同产品提出的特征变更请求相异,定位这些变更的影响减少了模型中的必选特征,降低了产品共性.本节进一步分析变更涉及的特征,通过共性提取和模型修剪,挖掘更多的特征用于后续产品的复用开发.

3.2.1 候选特征筛选

在变更涉及的特征中,并不是所有特征都有可能被定义为产品共性.分析这些特征,可发现其中两类特征可作为共性分析的候选:定位时插入的可选特征和形成的相似特征.其中,定位时插入的可选特征可直接作为产品共性的来源.对于一组相似的特征,可利用这些特征之间的共性部分进行复用开发.这些相似的特征由两种方式构成:一种由定位添加类型变更插入的可选特征组成,另一种由定位修改类型变更中的变更前和变更后特征组成.实际上,一组相似特征变更后的特征之间也存在相似关系,例如图 7 中的 f_{mrl},f_{prl} 和 f_{rv}.但是这些特征的父特征不同,对其重构或者修剪,可能破坏特征模型的层次结构.因此,我们不考虑这些相似特征.上述两种类型的候选特征均可从定位阶段输出的模型演化操作集合中,根据操作名称和操作目标筛选得到.由于两类特征存在交叉,我们首先分析相似特征而后在可选特征中进行选择.

3.2.2 相似特征分析

相似特征的分析以特征的共性和可变性分析为基础,通过重构的方式挖掘其中共性部分,同时修剪特征模

型.本文采用自然语言描述特征,因此,对特征进行共性和可变性分析需首先解析特征的内容.我们采用了Fillmore的“格语法(semantic grammar)”[31]对特征内容进行语义解析,该方法曾应用于Niu和Guo等人的研究中,通过格语法中的“语义格(semantic case)”解析需求描述中的人、地点、频率等上下文信息,进而识别需求的共性和可变性[32,33].我们也采用同样的方式解析特征.表5列出了本文解析特征时所用到的语义格.

Table 5 Semantic cases of features[33]

表5 特征语义格[33]

语义格	描述
agentive	实施特征的参与者
dative	特征行为影响的参与者
objective	特征行为影响的对象
instrumental	特征行为涉及的方式或者工具
factitive	特征行为引发的对象或者事物
locative	特征行为发生的地理或空间位置
temporal	特征行为发生的频率和持续时间
conditional	特征行为的触发条件

根据特征语义格,可对每组相似特征的内容进行解析和对比,识别特征的共性和可变性.表6显示了两组相似特征组的解析结果.特征 f_1,f_2 和 f_3 均是在添加风险时设置风险值,它们的可变点(variation point)发生在temporal,即,风险值的范围不同;f_4 和 f_5 通过不同的方式共享会议记录,其可变点发生在instrumental.

Table 6 Examples of parsing features

表6 特征语义解析示例

ID	Agentive	Objective	Instrumental	Temporal
f_1	用户	设置风险值	–	从低到高
f_2	用户	设置风险值	–	1~10
f_3	用户	设置风险值	–	1~5
f_4	用户	共享会议记录	通过在线	–
f_5	用户	共享会议记录	通过 E-mail	–

根据共性和可变性分析的结果,可对一组相似的特征进行重构.为保持重构前后的特征功能和配置约束,我们根据相似特征构成的方式定义了重构规则:

规则4. 对于一组相似的特征 $F=\{f_1,f_2,\ldots,f_n\}$,根据其共性和可变性分析结果,可将这些特征重构为一组父子特征 $F'=\{\{f_{ch_1},f_{ch_2},...,f_{ch_n}\},f_p\}$ f_p 为父特征,取自 F 的共性部分;$\{f_{ch_1},f_{ch_2},...,f_{ch_n}\}$ 为 f_p 的子特征,取自 F 中每个特征的可变部分.

- f_p 的配置约束可按照如下规则定义:
  1) 当 F 完全由可选特征构成时,将 f_p 定义为可选特征;
  2) 当 F 由变更前和变更后特征构成时,f_p 的配置约束与变更前特征一致.
- 对于子特征 $\{f_{ch_1},f_{ch_2},...,f_{ch_n}\}$ 的配置约束,如果 F 之间存在互斥关系,将 $\{f_{ch_1},f_{ch_2},...,f_{ch_n}\}$ 定义为 f_p 的一组单选特征;否则,将这些特征定义为 f_p 的一组多选特征.

图8显示了对表6特征 f_4 和 f_5 重构的结果.

Fig.8 An example of refactoring similar features

图8 重构相似特征示例

根据规则 4,f_4 和 f_5 重构为一组父子特征.父特征来自于二者的共性部分,即“共享会议记录”.子特征为共享的方式,可通过在线共享或者 E-mail 共享.由于 f_4 和 f_5 均为可选特征且非互斥,因此将“共享会议记录”定义为可选特征,两个子特征定义为一组多选特征.

3.2.3 可选特征分析

一个特征是否为可作为产品的共性,取决于多方面因素[7].在选择前,分析人员应建立决策准则对每个特征进行评价.根据一些典型的决策影响因素,我们设计了规则,从定位插入的或者重构生成的可选特征中选取产品共性,并在模型中修改其配置约束关系.

规则 5. 对于定位插入或者重构生成的可选特征 f,当该特征满足以下一个或者多个条件时可定义为产品共性,并在特征模型中将其更新为必选特征:

- 请求 f 的产品数量占所有产品的比例超过预定阈值 t_1;
- f 符合行业标准的程度大于预定阈值 t_2;
- f 符合产品战略的程度大于预定阈值 t_3.

由于特征变更的定义中包括请求特征的产品信息,因此请求特征的产品比例可以根据特征变更中的产品属性计算得到,其他两项因素则需要分析人员进一步收集数据.但是在应用规则前,应邀请领域专家确定各个影响因素的阈值.除了规则 5 列举的决策因素外,由于项目背景和环境不同,还可能存在其他决策因素,例如客户的优先级.本文不再对这些因素逐一列举.如果需要添加新的决策因素,可参照规则 5 的方式定义.

3.3 冲突消解

为了保证特征模型配置的一致性,本阶段进行配置冲突检测和消解.由于我们假设方法输入特征模型的配置是一致的,配置冲突只可由配置约束相关的特征模型变化引发[9].由此,冲突检测可从前面两个阶段发生配置约束关系变化的特征出发,而不是遍历整个特征模型.

配置冲突的检测主要依据表 2 所描述的冲突类型.发现冲突后,一般可通过删除或者修改其中一种配置约束关系达到消解的目的.因此对于同一类型配置冲突,可有多种消解策略.根据不同的冲突类型,我们列举了几种典型的消解策略:

- 策略 1:检查发生配置冲突的两个特征,移除依赖关系.该策略适用于消解 Type I,Type IV 和 Type V 类型的配置冲突.
- 策略 2:检查发生配置冲突的两个特征,移除互斥关系.该策略适用于消解 Type II,Type III 和 Type V 类型的配置冲突.
- 策略 3:将发生配置冲突的完全必选特征改为非完全必选特征,即,将该特征或者其到根节点路径上的任意特征更改为非必选特征.该策略适用于消解 Type I,Type II 和 Type III 类型的配置冲突.
- 策略 4:检查发生配置冲突的两个特征,消除特征之间单选约束.该策略适用于消解 Type IV 类型的配置冲突.

在检测过程中,一旦发现配置冲突,分析人员就需要根据冲突类型选择合适的消解策略.改变特征配置约束关系后,应仔细检查特征的功能和行为,保证特征间正确的关联关系.由于执行消解策略可再次引入新的配置约束关系变化,因此冲突消解完成后,还需根据消解冲突引发的配置约束关系变化继续检测,直到不存在任何配置冲突为止.

图 9 显示了一个消解配置冲突的例子.

Fig.9 An example of conflict resolution

图 9 冲突消解示例

可选特征“JBCM”是在定位阶段插入的可选特征,该特征描述了系统采用一个第三方的工具和方案管理配置库,而非系统内置的配置管理功能.它与系统原有特征“管理配置库”互斥.由于“管理配置库”为一个完全必选特征,因此,“JBCM”与“管理配置库”间被检测出 Type II 冲突.消解该冲突可采用策略 2 或者策略 3.分析人员选择了策略 3,将“管理配置库”改变为可选特征.

冲突消解后,即完成了特征模型的演化分析过程,输出结果为模型的演化操作集合,根据该集合对输入的特征模型操作,可得到演化后的特征模型.项目分析人员和开发人员可根据该特征模型进行后续的重用开发活动.

4 案例分析及验证

为了验证本文提出方法的可用性和有效性,我们建立了半自动化分析工具 FMEVO-Tool(工具的具体实现可参考 http://itechs.iscas.ac.cn/cn/membersHomepage/hujie/FMEvolutionAnalysis.rar).通过该工具,在一个实际的软件系统上进行案例分析.由于本文重点在介绍方法,本节只描述案例分析的过程和结果.

案例分析对象来自于一个软件过程管理系统 Qone,该系统与 Microsoft Project 系统相似,主要为组织或者机构内部的软件过程管理提供协作平台.Qone 系统的主要功能包括软件项目管理和过程管理等,例如项目计划、项目监控和软件度量.截至目前,该系统已经应用于超过 200 个组织和机构.由于客户所采用的软件过程管理方式存在差异,项目组通过定制开发的方式满足客户对系统的特定要求.在定制化开发过程中,特征模型用于对系统的共性和可变性建模,支持复用开发和系统配置.随着近年来客户数量的不断增长和用户需求的快速变化,该系统特征模型的演化更加频繁.

4.1 收集数据

我们通过 Qone 项目组收集了系统的特征模型,并根据第 2.1 节的扩展,补充特征间的前置、限制和相似关系.当前,该模型共包含 579 个特征.产品共性主要体现在项目管理、过程管理等软件过程管理常规功能,可变性则分布于项目数据、系统设置等功能.这些特征可根据客户环境进行选择和设定.图 1 所示的特征模型片段即来自该系统.项目组还提供了来自 7 个定制产品的 106 个特征变更请求,去重后得到 95 个变更请求.表 7 按照产品和变更类型显示了这些特征变更请求的统计.

Table 7 Summary of feature change requests from different products

表 7 定制产品的特征变更统计结果

产品	添加	修改	删除	总计
p_1	4	15	1	20
p_2	5	3	0	8
p_3	13	8	2	23
p_4	4	6	0	10
p_5	6	0	0	6
p_6	6	1	0	7
p_7	14	17	1	32
总计	52	50	4	106
去重	41	50	4	95

4.2 定位特征变更的影响

利用规则 1 和规则 2,我们在特征模型中分析了 95 个特征变更造成的“涟漪”效应.18 个特征变更被识别并经过分析人员确认,表 8 显示这些特征变更的信息.它们以修改类型为主,多数是根据修改类型的特征变更,利用相似关系识别.图 5 显示了其中的两个变更.

识别“涟漪”效应引发的特征变更后,共有 113 个特征变更需要进一步分析.利用规则 3,可对变更涉及特征的配置约束关系进行定义和更新.分析结果如下所示:

- 对于 41 个添加类型的特征变更,31 个请求添加的特征插入为可选特征,其余 10 个特征分别插入到 2 个单选特征组和 8 个多选特征组中.
- 对于 67 个修改类型的特征变更请求,54 个必选特征和其 64 个变更后特征组成了单选特征组,3 个变更

后特征插入到变更前特征所在的多选特征组中,并与变更前特征互斥.

- 对于 5 个删除类型的特征变更请求,只有一个特征的配置约束关系被改变,即,将“定时任务”从必选特征改为可选特征.

特征变更隐含的扩展关联关系变化随插入操作在特征模型中定义.而后,对插入的特征进行相似关系的分析,共发现 126 个相似特征.

Table 8 Statistics of the feature changes identified by analyzing the ripple effect

表 8 分析“涟漪”效应识别的特征变更统计结果

关联关系	修改											删除	总计
	cr_{13}	cr_{15}	cr_{17}	cr_{24}	cr_{52}	cr_{55}	cr_{57}	cr_{76}	cr_{78}	cr_{81}	cr_{94}	cr_{45}	
前置	0	0	0	0	0	0	0	1	0	0	0	1	2
限制	1	1	1	0	0	1	0	0	0	1	0	0	5
相似	0	0	0	4	2	0	1	0	2	0	2	0	11
总计	17											1	18

4.3 提取共性和修剪特征模型

在变更涉及的特征中筛选候选特征,得到 31 个可选特征和 126 个相似特征.在分析相似特征时,我们根据表 5 所示的语义格解析这些特征.通过对比每组相似特征的语义格后,得到这些特征的共性和可变性,进而根据规则 4 在特征模型中重构这些特征.分析结果如下所示:

- 对 124 个变更前特征和变更后特征组成的相似特征组重构,可得到 57 个父特征和 124 个子特征,其中,54 个父特征被定义为必选特征,3 个父特征仍然在原有的多选特征组中.124 个子特征分别定义为单选特征组.
- 对两个可选特征组成的相似特征组重构,可得到一个父特征和 2 个子特征.父特征被定义为可选特征.由于重构前特征间不存在互斥关系,因此两个子特征被定义为一个多选特征组.图 8 中的例子即源自于这两个特征.

重构相似特征后,共有 30 个可选特征提交给分析人员.Qone 项目组主要通过请求特征的客户比例和特征是否符合行业标准作为共性决策依据.根据这两项因素的阈值对特征进行选择,只有一个特征“维护项目资产库”满足规则.该特征与行业的软件过程管理标准相一致.

为了验证本阶段分析的有效性,我们选取了共性比例(commonality ratio,简称 CRatio)作为评价指标.CRatio 表示被定义为产品共性的特征的百分比,可通过模型中必选特征数量与所有特征数量相比得到.产品共性发生改变时,CRatio 也随之变化.因此,可以通过对比分析前后的 CRatio 显示分析方法是否可以挖掘出潜在的共性.CRatio 越高,越有利于复用开发.

表 9 显示了前一阶段和本阶段分析完成后 CRatio 的统计结果.特征变更定位阶段向模型插入的特征均为非必选特征,同时,将模型中原有的必选特征修改为非必选特征.本阶段则对定位插入的可选特征和形成的相似特征进一步分析,提高了模型中的必选特征数量.特征模型中的特征总数也随相似特征的重构而上升.对比两个阶段,可观察到 CRatio 提升了 4.1%,验证了本阶段对潜在共性挖掘的有效性.

Table 9 Comparison of the CRatio between this phase and the previous phase

表 9 对比本阶段与前一级阶段 CRatio

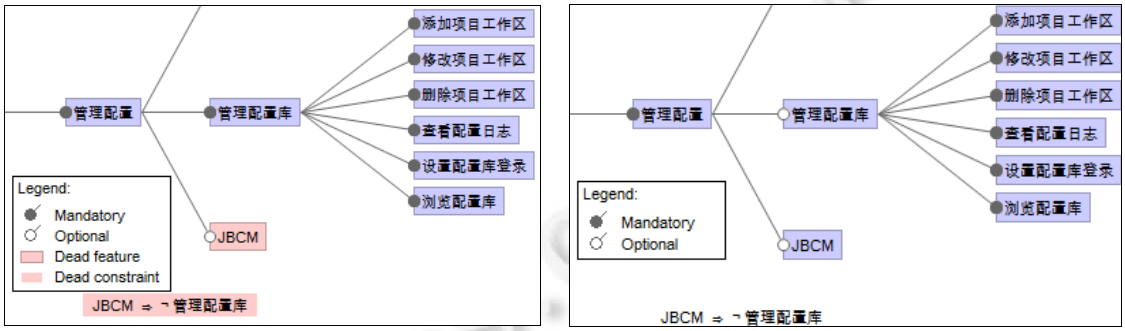
阶段	必选特征	特征总数	CRatio (%)
定位特征变更的影响	290	687	42.2
提取共性和修剪特征模型	345	745	46.3 (+4.1)

通过上述的分析可以发现,在演化过程中是否能够提取出共性,主要取决于两个条件:一是是否可从变更涉及的特征中发现候选特征,二是这些特征是否符合共性分析和决策规则.如果存在符合条件的特征,按照本文描述的分析过程即可发现其中潜在的共性.

4.4 消解冲突

在发生配置约束关系变化的特征之间检测配置冲突,只有一个 Type II 类型的冲突被发现并被消解.图 9 所示例子即来源于此.

我们利用两个现有的工具 FeatureIDE[34]和 RequiLine[35]验证冲突消解的效果,这两个工具都可以对特征模型进行图形化编辑,检查模型是否存在配置冲突.我们分别将冲突消解前后的特征模型输入到工具中.图 10 显示了在 FeatureIDE 中消解冲突前后的特征模型片段.在冲突消解前,可选特征“JBCM”的插入导致其与一个完全必选特征“管理配置库”互斥.在图 10(a)中,FeatureIDE 识别出该冲突.经过冲突消解后,图 10(b)显示,该冲突已被移除.最后对特征模型所有特征的配置约束关系检查,不存在配置约束冲突.



(a) 冲突消解前 (b) 冲突消解后

Fig.10 Screenshots before and after conflict resolution in FeatureIDE

图 10 冲突消解前后的 FeatureIDE 截图

4.5 扩展效果分析

演化分析完成后,我们统计了每个分析阶段特征模型的变化,进一步分析扩展特征关联关系对演化分析的影响和效果.表 10 显示了每个分析阶段后特征模型中每种关联关系相关特征的数量,多数的变化都发生在前两个阶段,尤其是必选、可选、单选和相似关系特征的数量.由于输入的特征变更多以添加和删除类型为主,定位这些变更增加了可选、单选和相似特征,同时削减了的输入模型中的必选特征.而在共性提取和模型修剪中,对相似特征的重构增加必选特征的数量,并消除了这些特征间的相似性,减低了相似特征的数量.

Table 10 Statistics of the features according to different dependency types during the evolution analysis

表 10 演化分析中不同类型关联关系相关特征统计结果

关联关系	输入	phase 1	phase 2	phase 3
必选	345	290	345	345
可选	52	84	82	84
单选	14	134	140	140
多选	168	179	178	178
依赖	65	81	81	81
互斥	12	20	14	14
前置	130	146	146	146
限制	92	95	95	95
相似	100	226	116	116

为了验证模型扩展对演化分析的影响和效果,我们分别在扩展特征关联关系前后的特征模型基础上运行分析方法,并统计了所有分析阶段取得的配置约束关系相关的演化操作,如图 11 所示.对比两种场景可观察到:在多数演化操作类型上,扩展后分析取得的操作数量都大于扩展前.尤其在 InsMan,InsAlt,DelAlt 和 MovToAlt 数量上,前者显著大于后者.二者之间的差距主要由于扩展特征关联关系增强了“涟漪”效应的分析,从而发现了 17 个修改类型的特征变更.定位这些特征变更,使得更多的特征被插入到或者移动到单选特征组中.这些操作也

为共性提取提供了更多候选特征,使得更多的特征被重构,提高了必选特征的数量.

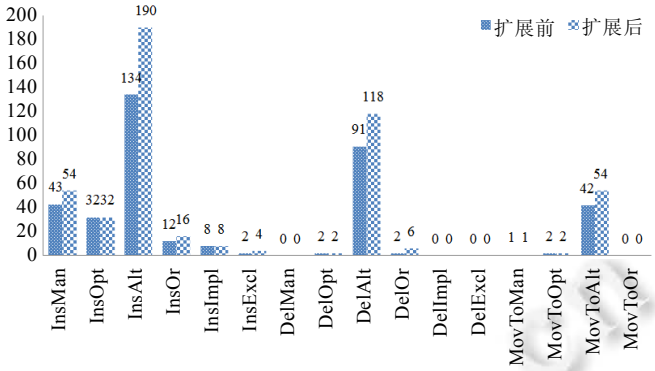


Fig.11 Model operations obtained by the evolution analysis based on the feature models before and after extending the feature dependency types

图 11 在扩展特征关联关系前后的特征模型基础上进行演化分析得到的模型操作

上述分析结果表明了扩展特征关联关系对演化分析的影响,同时也验证了扩展对于演化分析的必要性.即,在扩展基础上分析特征模型演化,可得到更为全面的演化操作.

5 讨 论

尽管案例分析验证了方法的可用性和有效性,但是该方法在某些方面仍然存在局限性.此外,在案例分析过程中,通过和项目人员的讨论,我们也得到一些经验.

5.1 方法的局限性

方法的局限性主要体现在 3 个方面:扩展特征关联关系、“涟漪”效应分析和冲突消解策略.

(1) 扩展特征关联关系与特征配置约束关系的相关性

对比配置约束关系和扩展关联关系,可以发现依赖与前置或者限制之间可存在一定的相关性.对于具有前置或者限制关系的特征,其可能在配置上也存在依赖关系.例如在图 3(a)中,完全必选特征“设置数据库”前置于可选特征“备份数据”,“备份数据”也依赖于“设置数据库”.而在一些研究[9,34]中,若可选特征依赖于完全必选特征,则该依赖关系被视为冗余,需在特征模型中消除.因此,图 3(a)中并没有显示二者间的依赖关系.但是,如果在“涟漪”效应分析只采用依赖关系,某些特征变更将无法被发现.例如,修改“设置数据库”增加系统支持的数据库类型,可引起“备份数据”的变更,如果不补充前置关系,则无法识别该类变更传播.因此,补充前置和限制关系对于“涟漪”效应分析是必要的.

(2) “涟漪”效应分析

由于特征是系统的高层抽象,其变更带来的“涟漪”效应具有不确定性[6,29,30].虽然我们设计了规则发现可能受影响的特征和特征变更的类型,但是为了提高分析的准确性,我们建议在利用规则分析后,将分析结果提交给分析人员,通过人工检查的方式保证结果的准确性.

此外,在分析“涟漪”效应时,我们只考虑从扩展的 3 种关联关系出发.实际上,父子特征之间也可能存在变更的传播.由于特征模型的树形结构,父子特征间的变更相比扩展关联关系上的变更传播更为复杂.修改模型高层的特征,可能导致其所有后继特征发生变化.本文中不考虑父子之间的变更传播,如有父特征变更影响子特征的场景,需在输入中逐一为受影响的子特征定义特征变更请求.

(3) 冲突消解策略

我们根据不同的配置冲突类型设计了消解策略,除了这些策略以外,还可能存在其他的选择,例如,将 Type I 中的非完全必选特征更新为完全必选特征.这样的操作虽然解决了配置冲突,但是该特征并不是所有产品都需

要的,违背其实际的配置要求.因此,我们并未将类似的策略纳入方法中.

在消解配置冲突时,我们为分析人员提供了多种消解策略.最终选择哪一种策略,则需要分析人员根据实际情况做出决策.由于执行不同消解策略花费的工作量不同,对系统和定制产品造成的影响也不尽相同.因此,分析人员可通过收集影响因素建立统一的评价方法,对每一种消解策略进行影响分析,从而选择合适的策略消解冲突.

5.2 案例分析经验

进行案例分析后,我们与 Qone 项目组探讨方法的使用和应用结果时,得到一些意见和建议.综合这些意见和建议,我们得到如下几点经验:

(1) 特征重构的价值分析

解析和重构相似特征是共性提取和模型修剪的重要步骤,在案例分析中,约有 126 个特征需要进行解析和重构.由于执行这些分析需要花费一定的工作量,对于定制规模较小的软件而言,重构虽然提升了复用,但是重构带来的开销可能高于收益.对于一些需要快速交付的产品,先重构后开发的方式也可能延迟交付时间.因此,在进行特征解析和重构时需要进行价值分析,平衡解析和重构的开销和收益.

(2) 在分析前对特征变更聚类

由于领域内包含众多的定制产品,可能出现对同一特征不同语义格的变更请求或者同一语义格的不同请求.针对此类变更请求,项目人员建议我们在演化分析前对变更进行聚类,从而提高分析的效率.

6 相关工作

本文的相关工作主要涉及两个方面:特征关联关系的扩展和特征模型的演化分析.

(1) 特征关联关系的扩展

特征模型最早由 Kang 在 1998 年提出[3],作为面向特征的领域分析(feature oriented domain analysis)方法的核心.由于特征建模具有简单、灵活、易理解等优势,使其一经提出就吸引了众多的研究人员,并逐步成为重要的共性和可变性建模方法之一.此后,很多研究在该模型的基础上进行扩展,以满足不同的应用场景.综合特征模型分析的系统化调研[18]和特征模型扩展工作的总结[2,36],可将扩展归纳为 3 个方面:特征的属性、特征的基数和特征的关联关系.

本文所提出的扩展从属于特征关联关系方面,因此,本节主要总结该方面的研究,并进行对比.

特征关联关系的扩展工作中,Lee 等人根据特征模型在设计可适应(adaptable)和可配置(configurable)资产时存在的缺陷,从特征操作的角度引入了多种关联关系[19].补充的特征关联关系包括使用(usage)、修改(modification)和激活(activation).Zhang 等人面向特征分析了需求间的关联关系,增加了静态和动态特征关联关系[20].其中,静态关联关系包括精化(refinement)和限制(constraint),动态特征关联关系主要为影响(influence)和交互(interaction).Peng 等人对特征模型进行了更严格地定义,给出了 3 种特征关联关系:使用(usage)、决定(decide)和配置依赖(configdepend)[21].通过这 3 种关联关系,更为细致地描述运行时和绑定时(binding)特征间的交互性及配置约束.

上述研究工作进一步增强了特征模型对特征相关性的表达能力,然而对于“涟漪”效应分析而言,关键是识别不同层次和类型关联关系上的特征变更传播.这些扩展只是对特征在配置和重用上相关性的进一步刻画,并没有从其他角度进行补充.将这些扩展得到关联关系与已有变更传播研究[24]对比可发现,采用这些扩展的特征模型仍然不足以支撑“涟漪”效应的分析.与这些研究不同,本文所提出的扩展主要面向特征变更的“涟漪”效应,分析特征间不同抽象层次或者不同方面的关联关系,从中选择可用于“涟漪”效应的关联关系类型补充到特征模型中,实现了在特征模型基础上分析特征变更“涟漪”效应.此外,扩展的关联关系还可用于识别潜在的产品共性,为共性提取和模型修剪提供更多的候选特征.

(2) 特征模型的演化分析

特征模型演化分析的研究主要包括演化影响分析、重构和模型诊断等多个方面.

在演化影响分析方面:Riebisch 主要以特征模型为桥梁建立需求与其他软件制品的跟踪关系,通过需求的变更分析演化对特征模型以及其他制品造成的影响[11];Peng 等人分析了影响特征模型演化的因素,识别可能发生的模型变化和受影响的特征,并对模型变化进行价值分析和排序[12].这些方法主要从需求工程的角度分析特征模型演化.

其他方面,Botterweck 等人归纳了特征模型演化的元操作集合 EvoOperators[25];Hwan 等人提出多种特征模型的操作,例如删除节点、删除子树[26];Thum 等人引入了重构、归纳(generalization)、特化(specialization)和随意性编辑(arbitrary edit)描述演化前后特征模型的变化[37];Alves 等人基于特征模型提出了产品线重构模式[38].还有许多研究人员采用 SAT(satisfiability solving),BDD(binary decision diagram)和 CSP(constraint satisfaction problem)对模型的有效性进行诊断[15,16,39].Guo 等人提出在演化时通过设定策略修复不一致的特征模型配置[10].

上述研究中,其中一部分以需求规约或者需求模型分析特征模型的演化.这些研究只是在需求层面上进行演化分析,通过特征模型表达共性和可变性的变化,并非是从特征模型本身的角度考虑演化.由于需求模型缺乏对特征和特征关联关系的描述,分析过程中可能忽略特征变更的“涟漪”效应,无法发现由此引发的共性和可变性变化.与这些工作不同的是,本文通过扩展特征关联关系实现了在特征模型基础上进行特征变更的“涟漪”效应分析,并通过共性提取和模型修剪发现更多的产品共性.此外,本文提出的方法还考虑了模型配置冲突的发现和消解.与上述研究中的模型诊断的研究相比,本文仅从发生配置约束变化的特征出发检测配置冲突.与 Guo 等人的研究类似,本文同样假设输入的特征模型不存在任何异常,配置冲突只可由模型演化操作引发,并不需要遍历整个特征模型进行检测.但是对于冲突消解,上述工作并未说明如何消解不同类型的配置冲突,Guo 等人的研究也仅描述如何修复删除特征造成的配置异常.本文则定义了详细的策略应对每一种配置冲突.

由于识别共性和可变性的变化是特征模型演化分析重要组成,因此我们在研究过程中也参考了共性和可变性分析的相关工作.分析相似特征时,采用 Guo 等人分析功能性需求的方法[33],通过“格语法”对特征进行语义分解.在对可选特征进行共性分析时,我们总结了相关工作中分析产品共性的决策因素[7,17,33,40],在这些因素基础上设计了规则辅助开发人员决策.

7 总 结

本文在分析当前特征模型演化分析的挑战和现有方法存在的问题基础上,提出了一种软件特征模型演化分析方法.该方法扩展了特征关联关系和模型演化元操作,通过这些扩展,支持在特征模型基础上分析和识别特征变更的“涟漪”效应引发的特征变更,发现和提取潜在的产品共性.同时,该方法还对演化过程中出现的配置冲突进行消解.为了验证该方法的可用性和有效性,实现了半自动化的分析工具,并选取了一个具有代表性的软件系统 Qone 进行案例分析.分析结果表明,我们所提出的方法可以用于实际的项目并辅助项目人员进行特征模型演化分析.

在研究过程中,我们也发现了该方法存在的不足,并在案例分析中收获了一些经验教训.这些都为我们指明了下一步的研究方向,包括:对特征变更进行聚类分析,提高方法的效率;对特征解析和重构进行价值分析;建立冲突消解策略选择方法;评价特征模型演化对软件开发的影响.此外,我们也将持续关注 Qone 项目,并计划采用更多的方式验证方法的有效性.同时,也希望在更多的软件系统和业务场景中进行应用,不断改进方法.

References:

[1] Apel S, Kastner C. An overview of feature-oriented software development. Journal of Object Technology, 2009,8(4):1−36. [doi: 10.5381/jot.2009.8.5.c5]

[2] Zhang W, Mei H. Feature-Oriented software reuse technology—State of the art. Chinese Science Bulletin, 2014,59(1):21−42 (in Chinese with English abstract).

[3] Kang K, Cohen S, Hess J, Novak W, Peterson A. Feature-Oriented domain analysis (FODA) feasibility study. Technical Report, CMU/SEI-90-TR-21, Software Institute, Carnegie Mellon University, 1990.

[4] Czarnecki K, Helsen S, Eisenecker U. Staged configuration using feature models. In: Proc. of the 3rd Int'l Conf. on Software Product Lines. Berlin: Springer-Verlag, 2004. 266−283. [doi: 10.1007/978-3-540-28630-1_17]

[5] Yau SS, Collofello JS, MacGregor T. Ripple effect analysis of software maintenance. In: Proc. of the 2nd Int'l Conf. on Computer Software and Application. Los Alamitos: IEEE Computer Society, 1978. 60−65. [doi: 10.1109/CMPSAC.1978.810308]

[6] Zhang L, Qian GQ, Li L. Software stability analysis based on change impact simulation. Chinese Journal of Computers, 2010,33(3): 440−451 (in Chinese with English abstract). [doi: 10.3724/SP.J.1016.2010.00440]

[7] Pohl K, Bockle G, Van Der Linden F. Software Product Line Engineering: Foundations, Principles, and Techniques. Berlin: Springer-Verlag, 2005. [doi: 10.1007/3-540-28901-1]

[8] Ardis M, Weiss DM. Defining families: The commonality analysis. In: Proc. of the 21st Int'l Conf. on Software Engineering. Los Alamitos: IEEE Computer Society, 1999. 671−672. [doi: 10.1145/302405.302929]

[9] Maben T, Lichter H. Deficiencies in feature models. In: Proc. of the Workshop on Software Variability Management for Product Derivation-Towards Tool Support. Berlin: Springer-Verlag, 2004.

[10] Guo JM, Wang YY, Trinidad P, Benavides D. Consistency maintenance for evolving feature models. Expert Systems with Applications, 2012,39(5):4987−4998. [doi: 10.1016/j.eswa.2011.10.014]

[11] Riebisch M. Supporting evolutionary development by feature models and traceability links. In: Proc. of the 11th IEEE Int'l Conf. and Workshop on the Engineering of Computer-Based Systems. Los Alamitos: IEEE Computer Society, 2004. 370−377. [doi: 10.1109/ECBS.2004.1316721]

[12] Peng X, Yu YJ, Zhao WY. Analyzing evolution of variability in a software product line: From contexts and requirements to features. Journal of Information of Software Technology, 2011,53(7):707−721. [doi: 10.1016/j.infsof.2011.01.001]

[13] Asadi M, Bagheri E, Mohabbati B, Gasevic D. Requirements engineering in feature oriented software product lines: An initial analytical study. In: Proc. of the 16th Int'l Conf. on Software Product Line. New York: ACM Press, 2012. 36−44. [doi: 10.1145/2364412.2364419]

[14] Buhne S, Lauenroth K, Pohl K. Why is it not sufficient to model requirements variability with feature models. In: Proc. of the Workshop on Automotive Requirements Engineering. 2004. 5−12.

[15] White J, Benavides D, Schmidt CD, Trinidad P, Dougherty B, Ruiz-Cortes A. Automated diagnosis of feature model configurations. Journal of Systems and Software, 2010,83(7):1094−1107. [doi: 10.1016/j.jss.2010.02.017]

[16] Benavides D, Trinidad P, Ruiz-Cortes A. Automated reasoning on feature models. In: Proc. of the 17th Int'l Conf. on Advanced Information Systems Engineering. Berlin: Springer-Verlag, 2005. 491−503. [doi: 10.1007/11431855_34]

[17] Chen K, Zhang W, Zhao HY, Mei H. An approach to constructing feature models based on requirements clustering. In: Proc. of the 13th Int'l Conf. on Requirements Engineering. Los Alamitos: IEEE Computer Society, 2005. 31−40. [doi: 10.1109/RE.2005.9]

[18] Benavides D, Segura S, Ruiz-Cortes A. Automated analysis of feature models 20 years later: A literature review. Information Systems, 2010,35(6):615−636. [doi: 10.1016/j.is.2010.01.001]

[19] Lee K, Kang KC. Feature dependency analysis for product line component design. In: Proc. of the 8th Int'l Conf. on Software Reuse. Berlin: Springer-Verlag, 2004. 69−85. [doi: 10.1007/978-3-540-27799-6_7]

[20] Zhang W, Mei H. Feature-Driven requirement dependency analysis and high-level software design. Requirements Engineering, 2006,3(11):205−220. [doi: 10.1007/s00766-006-0033-x]

[21] Peng X, Zhao WY, Xue YJ, Wu YJ. Ontology-Based feature modeling and application-oriented tailoring. In: Proc. of the 9th Int'l Conf. on Software Reuse. Berlin: Springer-Verlag, 2006. 87−100. [doi: 10.1007/11763864_7]

[22] Dahlstedt AG, Persson A. Requirements interdependencies: State of the art and future challenges. In: Proc. of the Engineering and Managing Software Requirements. Berlin: Springer-Verlag, 2005. 95−116. [doi: 10.1007/3-540-28244-0_5]

[23] Pohl K. Process-Centered Requirements Engineering. New York: John Wiley & Sons, Inc., 1997.

[24] Zhang H, Li J, Zhu LM, Jeffery R, Liu Y, Wang Q, Li MS. Investigating dependencies in software requirements for change propagation analysis. Journal of Information and Software Technology, 2014,56(1):40−53. [doi: 10.1016/j.infsof.2013.07.001]

[25] Botterweck G, Pleuss A, Dhungana D, Polzer A, Kowalewski S. Evofm: Feature-Driven planning of product-line evolution. In: Proc. of the ICSE Workshop on Product Line Approaches in Software Engineering. New York: ACM Press, 2010. 24−31. [doi: 10.1145/1808937.1808941]

[26] Hwan C, Kim P, Czarnecki K. Synchronizing cardinality-based feature models and their specializations. In: Proc. of the 1st European Conf. on Model Driven Architecture—Foundations and Applications. Berlin: Springer-Verlag, 2005. 331−348. [doi: 10.1007/11581741_24]

[27] Lehnert S. A taxonomy for software change impact analysis. In: Proc. of the 12th Int'l Workshop on Principles of Software Evolution and the 7th Annual ERCIM Workshop on Software Evolution. New York: ACM Press, 2011. 41−50. [doi: 10.1145/2024445.2024454]

[28] Li Y, Li J, Yang Y, Li MS. Requirement-Centric traceability for change impact analysis: A case study. In: Proc. of the Int'l Conf. on Software Process. Berlin: Springer-Verlag, 2008. 100−111. [doi: 10.1007/978-3-540-79588-9_10]

[29] Lock S, Kotonya G. An integrated probabilistic framework for requirement change impact analysis. Australasian Journal of Information Systems, 2007,6(2):38−63.

[30] Fu Y, Li MQ, Chen FZ. Impact propagation and risk assessment of requirement changes for software development projects based on design structure matrix. Int'l Journal of Project Management, 2012,30(3):363−373. [doi: 10.1016/j.ijproman.2011.08.004]

[31] Fillmore CJ. The case for case. In: Universals in Linguistic Theory. Rinehart & Winston, 1968. http://www.researchgate.net/publication/230875990_The_Case_for_Case

[32] Niu N, Easterbrook S. Extracting and modeling product line functional requirements. In: Proc. of the 16th Int'l Conf. on Requirements Engineering. Los Alamitos: IEEE Computer Society, 2008. 155−164. [doi: 10.1109/RE.2008.49]

[33] Guo JM, Wang YL, Zhang Z, Nummenmaa J, Niu N. Model driven approach to developing domain functional requirements in software product lines. IET Software, 2012,6(4):391−401. [doi: 10.1049/iet-sen.2010.0072]

[34] Thum T, Kastner C, Benduhn F, Meinicke J, Saake G, Leich T. FeatureIDE: An extensible framework for feature-oriented software development. Experimental Software and Toolkits, 2014,79(1):70−85. [doi: 10.1016/j.scico.2012.06.002]

[35] Massen T, Lichter H. RequiLine: A requirements engineering tool for software product lines. In: Proc. of the Workshop on Software Product-Family Engineering. Berlin: Springer-Verlag, 2004. 168−180. [doi: 10.1007/978-3-540-24667-1_13]

[36] Nie KM, Zhang L, Fan ZQ. Systematic literarture review of software product line variability modeling techniques. Ruan Jian Xue Bao/Journal of Software, 2013,24(9):2001−2019 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/4433.htm [doi: 10.3724/SP.J.1001.2013.04433]

[37] Thum T, Batory D, Kastner C. Reasoning about edits to feature models. In: Proc. of the 31st Int'l Conf. on Software Engineering. Los Alamitos: IEEE Computer Society, 2009. 254−264. [doi: 10.1109/ICSE.2009.5070526]

[38] Alves V, Gheyi R, Massoni T, Kulesza U, Borba P, Lucena C. Refactoring product lines. In: Proc. of the 5th Int'l Conf. on Generative Programming and Component Engineering. New York: ACM Press, 2006. 201−210. [doi: 10.1145/1173706.1173737]

[39] Czarnecki K, Wasowski A. Feature diagrams and logics: There and back again. In: Proc. of the 11th Int'l Conf. on Software Product Line. Alamitos: IEEE Computer Society, 2007. 23−34. [doi: 10.1109/SPLINE.2007.24]

[40] Moon M, Yeom K, Chae HS. An approach to developing domain requirements as a core asset based on commonality and variability analysis in a product line. IEEE Trans. on Software Engineering, 2005,31(7):551−569. [doi: 10.1109/TSE.2005.76]

附中文参考文献:

[2] 张伟,梅宏.面向特征的软件复用技术——发展与现状.科学通报,2014,59(1):21−42. [doi: 10.1360/972013-341]

[6] 张莉,钱冠群,李琳.基于变更传播仿真的软件稳定性法分析.计算机学报,2010.33(3):440−451. [doi: 10.3724/SP.J.1016.2010.00440]

[36] 聂坤明,张莉,樊志强.软件产品线可变性建模技术系统综述.软件学报,2013,24(9):2001−2019. http://www.jos.org.cn/1000-9825/4433.htm [doi: 10.3724/SP.J.1001.2013.04433]

胡洁(1982－),女,安徽芜湖人,博士,主要研究领域为软件定制化开发,软件需求演化.

王青(1964－),女,博士,研究员,博士生导师,CCF 高级会员,主要研究领域为软件质量管理,过程建模,知识管理,需求管理,软件协同工作.