软件学报 ISSN 1000-9825, CODEN RUXUEW
*Journal of Software*,2016,27(1):26−44 [doi: 10.13328/j.cnki.jos.004929]

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

# 基于特征模型的软件产品自动导出方法综述*

于文静[1,2], 赵海燕[1,2], 张 伟[1,2], 金 芝[1,2]

[1](高可信软件技术教育部重点实验室(北京大学),北京 100871)

[2](北京大学 信息科学技术学院 软件研究所,北京 100871)

通讯作者: 赵海燕, E-mail: zhhy.sei@pku.edu.cn

**摘 要**: 在针对特定领域的软件复用中,产品导出是主要活动之一.产品导出指的是,开发人员基于领域中可复用的软件制品开发出所需的软件产品.在产品导出过程中,产品导出效率决定了软件复用的收益.在诸多影响产品导出效率的因素中,手工进行产品导出是拉低产品导出效率的主要因素之一,其最终会导致软件复用收益降低.为了提高产品的导出效率,相关研究提出了一些自动导出软件产品的方法.在这些方法中,一种普遍采用的指导思想是基于特征模型自动导出软件产品.在诸多使用该思想进行产品导出的方法中,各方法所使用的实现方式差异很大.为了给基于特征模型自动导出软件产品提供更好的支持,基于现有研究,提出了一个分类框架,并使用该框架对现有基于特征模型自动导出软件产品的方法进行了分类和比较.另外,还进一步指出了现有研究中的不足,并提出解决这些不足的设想.

**关键词**: 软件复用;特征模型;产品导出;追踪关系

**中图法分类号**: TP311

中文引用格式: 于文静,赵海燕,张伟,金芝.基于特征模型的软件产品自动导出方法综述.软件学报,2016,27(1):26−44. http://www.jos.org.cn/1000-9825/4929.htm

英文引用格式: Yu WJ, Zhao HY, Zhang W, Jin Z. A survey of the feature model based approaches to automated product derivation. Ruan Jian Xue Bao/Journal of Software, 2016,27(1):26−44 (in Chinese). http://www.jos.org.cn/1000-9825/4929.htm

## A Survey of the Feature Model Based Approaches to Automated Product Derivation

YU Wen-Jing[1,2], ZHAO Hai-Yan[1,2], ZHANG Wei[1,2], JIN Zhi[1,2]

[1](Key Laboratory of High Confidence Software Technology (Peking University), Ministry of Education, Beijing 100871, China)

[2](Institute of Software, School of Electronics Engineering and Computer Science, Peking University, Beijing 100871, China)

**Abstract**: One of the basic activities in domain-specific software reuse is product derivation, which is deriving individual software products from the reusable software artifacts produced beforehand in the domain. The efficiency of product derivation decides the benefits of software reuse. Among all of the factors affecting the efficiency of product derivation, derivation being carried out manually is a major aspect with negative impacts that reduces the benefits of software reuse as a result. To improve the efficiency of product derivation, some approaches have been proposed to automate the derivation activity. A widely adopted idea in the approaches is automating the derivation activity based on feature models. In the approaches sharing the idea above, the implementation methods differ widely from one to another. To provide better support for feature model-based automated product derivation, this paper proposes a framework for classifying and analyzing these approaches. The paper also points out the problems in the existing researches and the possible solutions to the problems.

**Key words**: software reuse; feature model; product derivation; traceability

* 基金项目: 国家重点基础研究发展计划(973)(2015CB352201); 国家自然科学基金(60873059, 61272163); 可信软件基础研究重大研究计划(91318301)

Foundation item: National Program on Key Basic ResearchProject of China (973) (2015CB352201); National Natural Science Foundation of China (60873059, 61272163); Major Research Project Based on Trusted Software (91318301)

收稿时间: 2015-07-23; 修改时间: 2015-09-15; 采用时间: 2015-10-10; jos 在线出版时间: 2015-11-03

CNKI 网络优先出版: 2015-11-04 17:10:10, http://www.cnki.net/kcms/detail/11.2560.TP.20151104.1710.009.html

软件复用是提高软件生产效率和质量的有效方法.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.这里的领域,指的是一组具有相似和相近软件需求的应用系统所覆盖的功能区域[1].在针对特定领域的软件复用过程中,开发活动分为两个阶段:领域工程和应用工程.领域工程中的活动用于生产领域中可复用的软件制品;应用工程中的活动用于基于可复用软件制品开发出所需的软件产品,或称导出所需软件产品.

在针对特定领域的软件复用中,通过复用提升所需软件产品的开发效率是主要目标,也是软件复用收益的来源.在基于可复用软件制品导出所需软件产品时,如果导出过程不够高效,软件复用的收益将无法保证[2−4].影响产品导出效率的因素有很多,手工进行产品导出是拉低产品导出效率的主要因素之一[2−4].为了提高软件产品的导出效率,现有研究给出了一些自动导出软件产品的方法.在这些方法中,一种普遍采用的指导思想是基于特征模型自动导出软件产品,具体来说,

- 在领域工程阶段,使用特征模型作为领域需求模型建模领域中的共性、变化性需求,并建立特征模型和可复用软件制品(包括可复用的分析、设计、实现制品)之间的追踪关系.
- 在应用工程阶段,通过对特征模型进行定制(保留特征模型中的一部分特征,去掉其他特征),得到一个特征模型定制结果,也就是一个产品的需求;然后,根据特征模型和可复用软件制品间的追踪关系,自动导出特征模型定制结果对应的各种软件制品(分析、设计、实现制品),从而最终得到软件产品.

我们将使用上述思想的方法统称为基于特征模型的软件产品自动导出方法.这些方法虽然指导思想相同,但实现方式之间存在很大差异,而目前尚无针对这些方法的系统分析与比较.为了给基于特征模型自动导出软件产品提供更好的支持,本文提出了一个分类框架,并依照框架内容,从追踪关系组织方式、制品导出方式两个方面入手,对现有基于特征模型自动导出软件产品的方法进行分析和比较.在此基础上,指出现有研究中的不足,并提出解决这些不足的设想.

需要指出的是,本文的关注点集中于基于特征模型自动导出软件产品的相关研究.虽然这些研究所使用的具体实现技术涉及到模型转换[5,6]、模型间追踪关系[7,8],但本文不会包含专门论述模型转换、模型间追踪关系研究的论文,而仅包含目标为基于特征模型进行软件产品自动导出的论文.这样做的原因是:虽然模型转换、模型间追踪关系在本文所关注问题的具体实现技术中有所涉及,但二者的相关研究分支众多,论文数量也十分庞大.在这些研究中,如果研究本身的目标并不是基于特征模型进行软件产品的自动导出,就没有与本文所调研的其他研究进行比较的价值.而且,包含这样的研究反而会分散本文的关注点.

本文第 1 节介绍基于特征模型自动导出软件产品的相关知识.第 2 节阐述用于对现有基于特征模型自动导出软件产品方法进行分类的分类框架.第 3 节根据分类框架对现有基于特征模型自动导出软件产品方法进行分析和比较.第 4 节对现有方法中存在的问题和可能的解决思路进行介绍.最后,第 5 节对全文进行总结.

## 1 基于特征模型的软件产品自动导出

本节对基于特征模型自动导出软件产品的相关内容进行介绍.首先介绍特征模型的相关概念,然后介绍基于特征模型自动导出软件产品的相关概念和基本思想.

### 1.1 特征模型

特征模型是对领域内一系列软件产品的共性、变化性需求的抽象表示.在领域工程阶段,开发人员通过对领域内一系列软件产品的需求进行分析,最终构建出特征模型.在应用工程阶段,开发人员通过对特征模型进行定制(保留特征模型中的一部分特征、删除其他特征),得到特征模型定制结果.一个合法的特征模型定制结果就是领域内一个软件产品的需求模型.

特征模型的元模型最早由 FODA 方法[9]提出,之后,FORM 方法[10]、FeatuRSEB 方法[11]及 FODM 方法[12]等均对 FODA 方法中特征模型的元模型进行了扩展,极大地丰富了特征模型中的元素种类、提升了特征模型的描述能力.对于目前特征模型中众多种类的元素,本节仅对其中一些常用的元素进行介绍,而不会对各种方法中出现的特征模型元素加以一一列举和说明.因为特征模型本身并不是本文论述的重点,本节仅意在使人们对特

征模型有初步的了解,以便于后文产品自动导出方法的说明.

特征模型由一组特征和它们之间的关系构成.特征是一种具有客户/用户价值的软件特点,本质上是一组相互紧密关联的软件需求[12].特征之间的关系分为精化和约束两种.

- 精化关系将不同粒度的特征组织成树状结构,一个父特征可以被精化为一组粒度更细的子特征.在对特征模型进行定制时,当父特征被保留时,对于其每个子特征而言:若子特征是必选特征,则该子特征必须被保留;若子特征是可选特征,则该子特征可被保留也可不被保留.
- 约束关系用于描述特征间语义上的依赖关系,是定制特征模型时必须遵守的规则,用于保证特征模型定制结果的合法性.如果一个特征模型定制结果违反了约束关系,该定制结果就是不合法的,即,其所表示的需求内容存在冲突、不完整等问题.约束关系主要包含 require,exclude,xor 和 or 这 4 种.“特征 *X* require 特征 *Y*”表示:若 *X* 被保留,那么 *Y* 也必须被保留.“特征 *X* exclude 特征 *Y*”表示:*X* 和 *Y* 不能同时被保留.xor 和 or 是存在于父子特征之间的约束关系.xor 表示:当父特征被保留时,其子特征中有且只有一个特征可以被保留;or 表示:当父特征被保留时,其子特征中至少有一个特征应该被保留.

图 1 是音频播放软件领域的特征模型示例.在该特征模型中,

(1) 特征“音频播放软件”精化为“解码器”、“CD 播放”和“运行平台”这几个子特征.其中,“CD 播放”特征是可选的,表示该特征在定制时不一定被保留,即,一个音频播放软件中不一定包括 CD 播放这个功能;

(2) 特征“CD 播放”require 特征“WAV”表示,当“CD 播放”特征被包含在一个音频播放软件中时,“WAV”特征也必须被包含在该软件当中.该约束存在的原因是,播放 CD 时必须使用 WAV 解码器;

(3) 特征“CD 播放”exclude 特征“手机”表示,“CD 播放”特征与“手机”特征不能同时被包含在一个音频播放软件中.该约束存在的原因是,手机不能被用来播放 CD;

(4) “解码器”与其两个子特征间的 or 关系表示的是,播放软件中的解码器可以是 MP3 解码器或 WAV 解码器.一个产品所包含的解码器可以是其中一种,也可以是两种;

(5) “运行平台”与其两个子特征间的 xor 关系表示的是:播放软件的运行平台可以是手机操作系统,也可以是桌面操作系统.一个音频播放产品只能运行在其中一种操作系统上.

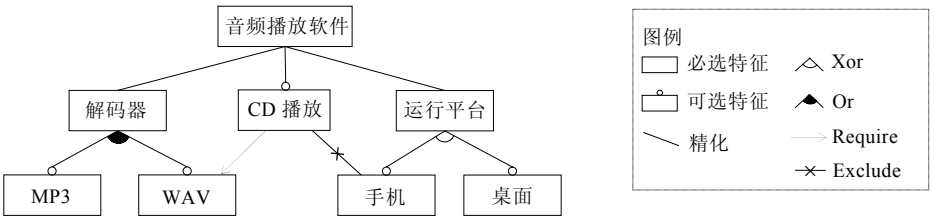


Fig.1 An example of feature models

图 1 特征模型的示例

### 1.2 基于特征模型的软件产品自动导出

一个软件产品由多种软件制品组成,如用况、构件图、类图、代码等.在应用工程中,基于可复用软件制品导出一个新的软件产品,本质上为逐步导出组成新软件产品的各种软件制品.在后文中,将这些需要导出的组成新产品的各种软件制品(用况、类图、代码等),均称为目标制品.

在针对特定领域的软件复用过程中,为了尽可能地提高软件产品的导出效率,相关研究提出了一些自动导出方法,用于导出组成所需软件产品的各种目标制品.在这些方法中,一种普遍采用的指导思想是基于特征模型进行目标制品的自动导出.具体来说:在领域工程中,使用特征模型建模领域中的共性、变化性需求,并建立特征模型和可复用软件制品间的追踪关系;在应用工程中,开发人员通过定制得到一个合法的特征模型定制结果后,便可根据特征模型和可复用软件制品间的追踪关系,自动导出特征模型定制结果对应的各种目标制品,最终得

到所需的软件产品.

在基于特征模型自动导出目标制品的相关研究中,Kang 等人[10]、Gomaa 等人[13]、Botterweck 等人[14]以及 Krueger[15,16]均对上述自动导出目标制品的基本思想进行了描述;Filho[17]基于上述基本思想实现了手机软件产品的自动导出.但以上几个研究均未说明实现上述基本思想的具体方法.另有一些研究则对上述基本思想给出了各自不同的实现,我们在调研了这些研究的基础上,从自动导出目标制品的两个关键步骤——建立特征模型和可复用软件制品间的追踪关系以及利用追踪关系导出目标制品入手,对这些研究进行分析和比较,希望为基于特征模型自动导出软件产品的进一步研究提供基础.

## 2 分类框架

本节给出基于特征模型自动导出软件产品方法的分类框架,该框架从两个方面对现有方法进行分类:特征模型和可复用软件制品间追踪关系的组织方式以及根据追踪关系自动导出目标制品的导出方式.下面对框架的内容进行详细介绍.

### 2.1 追踪关系组织方式的分类方法

在应用工程中,基于特征模型自动导出目标制品的过程是:根据特征模型定制结果,自动地对可复用制品进行一系列操作,最终得到目标制品.其中,特征模型和可复用软件制品之间的追踪关系,是根据特征模型定制结果自动地对可复用制品进行操作的依据.因此,特征模型和可复用软件制品之间的追踪关系,其实描述的是特征的绑定状态(被保留/删除)和对可复用制品所做操作之间的对应关系;每个追踪关系由两部分信息组成,分别描述的是特征的绑定状态和与其对应的对可复用制品所做的操作.

在自动导出目标制品的过程中,特征模型定制结果是推导过程的源端,目标制品是推导过程的目的端,推导过程的方向是从特征模型定制结果到目标制品.以推导过程的方向为参照,设特征模型到可复用制品方向为正向,可复用制品到特征模型方向为反向.基于上述设定,特征模型和可复用制品间追踪关系的组织方式分为正向式和反向式两种.

- 正向式指的是,从特征模型出发对追踪关系进行组织.具体来说,按照特征的绑定状态对追踪关系进行聚集或分离,即:将追踪关系中特征绑定状态信息相同的追踪关系聚集在一起,将特征绑定状态信息不同的追踪关系互相分离开来.
- 反向式指的是,从可复用软件制品出发对追踪关系进行组织.具体来说,按照可复用制品的结构对追踪关系进行排布,即:将追踪关系直接标注于被操作的可复用制品上;或者,将结构上存在关联的可复用制品对应的追踪关系聚集在一起,将结构上无关的可复用制品对应的追踪关系分离开来.

正向式和反向式是对同一内容的不同组织方式,二者并无明显的优劣之分.所以,开发者可以根据自己的偏好对追踪关系组织方式进行选择.

### 2.2 目标制品导出方式的分类方法

在基于可复用软件制品导出目标制品时,导出方式分为减量式和增量式两种.这两种方式实际上代表着两种不同的可复用制品管理思想,两种方式中被管理的可复用制品的形态也是不同的.

- 减量式指的是,目标制品以通过不断缩减领域工程中构建出来的可复用软件制品进行导出.

具体来说,在领域工程阶段,开发人员所要构建的可复用软件制品是:一些面向整个领域的软件制品(后文简称为领域制品),如面向领域的用况模型(领域用况模型)、面向领域的体系结构(领域体系结构)等.一个领域制品,就是领域中所有软件产品的同种类目标制品的并集,同时还可能包含领域制品元素的可变性信息(元素是可选元素还是必选元素)和元素间的约束关系信息(元素间的互斥、依赖关系).例如,领域用况模型就是领域中所有软件产品的用况模型的并集,同时还可能包含领域用况模型中元素的可变性信息和元素间的约束关系信息.在构建领域制品的同时,开发人员还需构建特征模型和领域制品之间的追踪关系,即,特征绑定状态和对领域制品所做操作之间的对应关系.在减量式目标制品导出方式中,追踪关系的语义是:给定一个合法的特征模型定制

结果,如果定制结果中特征的绑定状态和追踪关系中描述的特征绑定状态相同,则不做任何操作;反之,则删除追踪关系中所示的领域制品元素.在应用工程阶段,在通过定制得到一个合法的特征模型定制结果后,以特征模型定制结果、领域制品、特征模型和领域制品之间的追踪关系为输入,根据追踪关系的语义,删除领域制品中的某些元素,便得到目标制品.减量式目标制品导出方式的优点是:在构建领域制品时,开发人员可以对领域制品所包含的元素及其之间的关系进行整体性的考虑,从而能够在一定程度上确保最终导出的目标制品的质量.上述方式的缺点在于:领域制品本身可能会非常复杂,有时甚至是难以构建出来的,所以构建和维护领域制品都会花费大量的人力和时间.因此,减量式目标制品导出方式的瓶颈在于领域制品的构建和维护.

- 增量式指的是,目标制品以通过不断叠加领域工程中构建出来的可复用软件制品进行导出.

具体来说,在领域工程阶段,开发人员所要构建的可复用软件制品是:对于每一种目标制品,均构建一个核心制品和一组对于核心制品的修改(后文简称修改).其中,一个核心制品是,领域中一个只具备基本功能的软件产品所包含的某种软件制品;一个修改指的是,对核心制品进行增加、删除、替换其中元素的操作.例如,若目标制品是产品的体系结构,那么核心制品就是领域中一个只具备基本功能的产品的体系结构,一个修改就是向核心体系结构中增加构件、删除接口等的操作.在实践中,有些增量式方法将核心制品设置为空,这样就无需构建核心制品,仅构建一组修改.在构建修改的同时,开发人员还需构建特征模型和这组修改之间的追踪关系,即,特征绑定状态和每个修改应如何使用的对应关系.在增量式目标制品导出方式中,追踪关系的语义是:给定一个合法的特征模型定制结果,如果定制结果中特征的绑定状态和追踪关系中描述的特征绑定状态相同,则将追踪关系中所示的修改施加在核心制品上;反之,则不做任何操作.在应用工程阶段中,在通过定制得到一个合法的特征模型定制结果后,以特征模型定制结果、核心制品、一组修改、特征模型和这组修改间的追踪关系为输入,根据追踪关系的语义,依次将某些修改施加在核心制品上,便导出了目标制品.与减量式相比,该方式的优点是无需构建面向整个领域的软件制品,避免了构建和维护领域制品带来的问题.但该种方式也存在缺点:在领域工程中,针对每一种目标制品,开发人员构建的可复用软件制品是一个核心制品和一组修改.当修改的数目不断增大时,开发人员很难对这些修改形成一个全局的观察和认识,这会导致:在目标制品自动导出过程中,如果对核心制品施加的修改的集合相同,但修改的施加顺序不同,那么导出的目标制品可能不完全相同,其中可能存在错误的目标制品.因此,如何保证在所施加的修改集合相同时,无论施加顺序如何,导出的所有目标制品均相同,仍然是待解决的问题.

## 3 基于特征模型的软件产品自动导出方法

本节对现有研究中基于特征模型的软件产品自动导出方法进行详细的介绍和分析.首先介绍综述文献的获取过程;接着,根据不同的目标制品导出方式对已有方法进行介绍;然后,根据不同的追踪关系组织方式对已有方法进行介绍.

### 3.1 文献获取过程

本文第 3 节将总共介绍 29 篇文献,这些文献的获取过程如下.

(1) 使用 DBLP(http://www.dblp.org/search/index.php),以 product 和 derivation 为关键字进行搜索,检索题目中同时包含这两个关键字的论文.DBLP 是德国特里尔大学负责开发和维护的数字书目索引与图书馆项目,它提供计算机领域科学文献的搜索服务,其所检索的外部数据库包含了 ACM,IEEE 等计算机领域数据库.以 product 和 derivation 同时为关键字进行文章题目检索的原因是:

① 在针对特定领域的软件复用研究中,product derivation 专指基于可复用制品导出所需软件产品;

② 考虑到有的论文题目中可能不会将 product derivation 作为短语使用,而是采用如 the derivation of products 等将两个词分开使用的方式,故检索条件设置为同时包含 product 和 derivation;

③ 这一步检索的目的在于获得一个初始论文集合,而不是尽可能找到所有相关论文,故检索时仅搜索题目中含有关键字的论文.得到初始论文集合后,就可以此为开端,通过该集合中论文的参考文献进一步获取论文.

(2) 对第(1)步所得文献进行阅读,在文献阅读过程中,对所读文献的参考文献中可能和自动导出软件产品相关的文献进行检索,得到文献后循环执行本步,即:阅读文献并根据参考文献查找新的可能相关的文献,直到不再获取新的文献为止.在阅读文献的过程中,同时对所读文献进行筛选,仅保留论文主要内容是基于特征模型自动导出软件产品的文献.

(3) 以第(2)步中筛选所得文献的第 1、第 2 作者为关键字,在 DBLP 上检索以他们为著者的论文,对所得结果进行阅读,并筛选出论文主要内容是基于特征模型自动导出软件产品的文献.以文献的第 1、第 2 作者为关键字进行检索,是以作者作为线索进行相关文献的查找.一位研究者通常会在同一问题上开展一系列的研究工作,通过作者查找文献,可以了解到其系列工作的进展情况.

(4) 通过前 3 步总共筛选得到 92 篇文献,对这些文献进行精读.在阅读这 92 篇文献的过程中,仅保留详细叙述产品自动导出实现技术的文献,去掉粗略描述导出的指导思想、导出流程的文献,最终得到 29 篇文献.这 29 篇文献接下来将会较为详细地加以介绍.

### 3.2 目标制品导出方式分类结果及方法介绍

根据目标制品导出方式对现有研究分类的结果见表 1.在所综述的 29 篇文献中,有的文献仅详述了追踪关系构建方法,而没有详细叙述目标制品导出过程,且通过论文内容无法推断出所使用的导出方式是增量式还是减量式,如基于变化性建模框架的产品导出方法(COVAMOF)[45],因而该方法没有出现在表 1 中.

**Table 1** Classification of the approaches according to the method of deriving target artifacts

**表 1** 根据目标制品导出方式对现有研究分类的结果

| 目标制品导出方式 | 方法名称 |
|---|---|
| 减量式 | 1. 基于规则的用况模型-特征模型自动映射方法[18]<br>2. 基于预处理的代码导出方法(GenArch)[19–21]<br>3. 基于模板的特征模型映射方法[22]<br>4. 基于 Kutulu 的体系结构导出方法[23]<br>5. 领域用况模型建模管理方法(PLUSS)[24,25]<br>6. 基于模型转换的类图导出方法[26]<br>7. 基于颜色标注的代码导出方法(CIDE)[27]<br>8. 基于模型转换的产品导出方法[28,29] |
| 增量式 | a. 基于片段的用况模型导出方法(MSVCM)[30]<br>b. 基于逐步精化思想的产品导出方法(AHEAD)[31]<br>c. 基于变化集的体系结构管理方法[32,33]<br>d. 模型驱动的行为制品导出方法[34]<br>e. 以特征为单位的代码导出方法[35]<br>f. 基于特征模型实现建模的产品导出方法[36]<br>g. 基于 VML 的体系结构导出方法[37,38]<br>h. 基于增量的产品导出方法[39]<br>i. 基于 XVCL 的产品导出方法[40,41]<br>j. 面向特征的产品开发方法(FODM)[12,42,43]<br>k. 基于 TDL 的用况模型导出方法[44] |

下面对所综述的方法进行详细介绍.

#### 3.2.1 减量式导出目标制品的方法

领域用况模型建模管理方法(PLUSS)[24,25](表 1 中方法 5)可用于减量式导出产品的用况模型.在领域工程阶段,开发人员需首先构建出面向整个领域的领域用况模型.在领域用况模型中,每个领域用况的内容采用表格的形式进行描述,一个领域用况是一张表格.图 2 是 PLUSS 方法中一个领域用况的示例.表格中的每一行描述用户和软件系统的一次交互.表格的行号用于标记交互的发生顺序、交互的可选性以及交互之间的约束关系.行号中,数字表示交互的发生顺序.同时,若行号中的数字不带括号,表示该顺序位置上的交互在定制时需至少保留 1 个,反之则表示交互可不必保留;若相同数字编号后无字母,那么这些相同数字的交互之间存在互斥关系,反之表示这些交互之间不存在约束关系.在建立起领域用况模型后,开发人员需要根据领域用况模型建立特征模型,

并建立特征模型和领域用况模型间的追踪关系.根据领域用况模型构建特征模型的方法是:为领域用况中的每个交互均创建一个特征;然后,根据需要再创建一些特征,用以将领域用况交互对应的特征组织成一棵特征树.因此,在特征模型构建起来后,每个领域用况交互都与一个特征自然建立起了追踪关系.在给定一个合法的特征模型定制结果后,根据追踪关系,从领域用况模型中去掉被删除特征对应的用况交互,便得到软件产品的用况模型.

| | 行号 | 用户行为 | 系统行为 |
|---|---|---|---|
| 必选交互 | 1 | … | … |
| 两个交互中必须且只能选 1 个 | 2 | … | … |
| | 2 | … | … |
| 两个交互中至少选 1 个 | 3a | … | … |
| | 3b | … | … |
| 可选交互 | (4) | … | … |
| 两个交互中可选 0~2 个 | (5)a | … | … |
| | (5)b | … | … |
| 两个交互中可选 0~1 个 | (6) | … | … |
| | (6) | … | … |

Fig.2 An example of the domain use cases in PLUSS

图 2 PLUSS 方法中一个领域用况的示例

基于规则的用况模型-特征模型自动映射方法[18](表 1 中方法 1)用于减量式导出软件产品的用况模型.为了描述领域用况模型,该方法提出扩展 UML 中用况模型的元模型来描述领域用况模型中需求的变化性.具体来说,对用况间关系 include 增加基数标注{0..1}或{1..1},来表示被包含用况对于主用况来说是可选或必选的用况;对用况间关系 extend 增加基数标注{0..1}或{1..1},来表示扩展用况对于被扩展用况是可选或必选的.在建立起领域用况模型后,该方法给出了一组映射规则,用于将领域用况模型自动地映射为特征模型,同时生成领域用况模型和特征模型间的追踪关系.映射规则是:将每个用况映射成一个特征;每个 include/extend 关系映射成一个特征间的精化关系;include/extend 上的基数{0..1},{1..1}分别映射为子特征的可选、必选属性.在自动映射的过程中,用况和特征之间自动形成一对一的追踪关系.这样,在给定一个合法的特征模型定制结果后,根据追踪关系,从领域用况模型中去掉被删除特征对应的用况和这些用况间的关系,便自动导出软件产品的用况模型.

基于模型转换的类图导出方法[26](表 1 中方法 6)用于减量式导出软件产品的类图.为了描述领域类图,该方法提出向 UML 类图元模型中引入标记《vp》,用以描述领域类图中元素的变化性.标记《vp》的作用是:使用《vp》对类图中某个元素进行标记,就表示被标记的类图元素是可选的类图元素,该元素在定制时可以保留也可以删除.在建立起领域类图后,需要开发人员建立特征和可选类图元素间的对应关系作为追踪关系,并采用 QVT-R 语言[45]对追踪关系进行描述.在给定一个合法的特征模型定制结果后,根据 QVT-R 描述的追踪关系,从领域类图中去掉被删除特征对应的类图元素,便得到软件产品的类图.事实上,该方法将类图的自动导出问题看成是将特征模型定制结果自动转换成类图的模型转换问题.同时,将特征和可选类图元素间的追踪关系看成是模型转换的转换规则.因此,采用模型转换规则描述语言 QVT-R[45]来描述追踪关系,作为转换依据,实现软件产品类图的自动导出.

基于模板的特征模型映射方法[22](表 1 中方法 3)用于将特征模型映射为用况、类图等多种目标制品.在该方法中,开发人员需首先构建出各种目标制品的模板,也就是用于导出各种目标制品的领域制品;接着建立特征模型和模板间的追踪关系;然后,在给定一个合法的特征模型定制结果后,根据追踪关系,从模板中去掉一些模板元素,便得到目标制品.与前面所述方法 5、方法 1、方法 6 不同的是:方法 3 中的领域制品中并不包含领域制品元素的可选性信息或元素间的约束关系信息,而仅单纯地是领域中所有产品的同种目标制品的超集.例如,领域类图中不包含类图元素(类、属性、操作等)是可选还是必选、元素之间的互斥关系这些信息,而仅包含类图元素.因此,方法 5、方法 1、方法 6 这 3 种方法中的领域制品本身是可以定制的,即便没有特征模型定制结果和

追踪关系作为辅助,开发人员也可以直接在领域制品上手动定制出所需的目标制品;而方法 3 中的领域制品本身是不可定制的,定制工作只能在特征模型上进行,然后通过特征模型和领域制品间的追踪关系,传递到领域制品上,最终得到目标制品.

还有几种减量式方法与方法 3 相同,所构建的领域制品仅仅是领域中所有产品的同种目标制品的并集,不包含领域制品元素的变化性和元素间的约束关系信息.这些方法包括:

- 基于 Kutulu 的体系结构导出方法[23],用于导出软件产品的构件图;
- 基于颜色标注的代码导出方法(CIDE)[27]和基于预处理的代码导出方法(GenArch)[19−21],用于导出使用 Java 实现的软件产品的代码;
- 基于模型转换的产品导出方法[28,29],用于导出产品的序列图、类图等多种 UML 模型,所导出的 UML 模型,在后续工作[46]中被用于估算产品运行时的性能.

在本节介绍的所有减量式方法中,方法 1、方法 5、方法 6 这 3 种方法与其余方法在领域制品的建模方式上略有不同:方法 1、方法 5、方法 6 这 3 种方法将领域制品元素的可选性及领域制品元素间的约束关系直接建模在领域制品中;其余方法没有将这些信息直接建模在领域制品中,而是让特征模型来承担建模这些信息的工作,然后,通过追踪关系将这些信息从特征模型传播到领域制品中去.若要在领域制品中直接建模其元素的可选性及元素间的约束关系,需要对目标制品的元模型进行扩展,为建模上述信息提供描述机制上的支持,如方法 5 中的交互标号、方法 1 中对 include,extend 增加的基数标注等;若不在领域制品中建模这些信息,则无需对目标制品的元模型进行扩展.

3.2.2 增量式导出目标制品的方法

基于片段的用况模型导出方法(MVCSM)[30]是增量式导出产品用况模型的方法.该方法的前身[47]意在使用特征模型对领域中可复用的用况片段进行管理,后经扩展[30],用于基于特征模型自动导出产品的用况模型.在领域工程阶段,MVCSM 方法设核心用况模型为空,将用户与领域中所有产品的交互行为建模为一组对核心用况模型的修改,然后建立特征模型和修改之间的追踪关系.图 3 是该方法中追踪关系的实例.实例中:左边一列 Feature expression 是特征的逻辑描述式,用以描述特征的绑定状态信息;右边一列描述的是对核心用况模型施加的修改.Scenario 和 advice 指的都是用况片段,SC01,ADV01 等指的都是用况片段的名称,select 和 evaluate 指的都是将用况片段添加到核心用况模型中.“select scenario SC01”的意思就是向核心用况模型中添加用况片段 SC01.图 3 中追踪关系的语义是:给定一个合法的特征模型定制结果,根据定制结果对逻辑表达式进行判断,当表达式取值为真时,就执行表达式对应的修改;为假时,则不做任何操作.图 4 是 MVCSM 方法中两个用况片段的实例.用况片段以表格的形式描述用户与系统的一段交互内容,并说明被添加到核心用况模型中时的添加规则.图 4(a)所示的用况片段没有对添加规则作特殊说明,在添加时是直接添加到核心用况模型中去;图 4(b)中的用况片段添加规则是“Before:P1”,在添加时就需添加在标号为 P1 的交互之前,标号为 P1 的交互其实就是图 4(a)中所示的 Id 为 P1 的交互.

| Feature expression | Task |
|---|---|
| EShop | Select scenario SC01 |
| | Select scenario SC02 |
| Not (Shopping Cart and Bonus) | Evaluate advice ADV01 |
| (shopping cart and bonus) | Evaluate advice ADV02 |
| Update user preferences | Evaluate advice ADV03 |

Fig.3 An example of the traceability in MSVCM

图 3 MSVCM 方法中追踪关系的实例

在 MVCSM 方法中,对核心用况模型施加的修改操作类型均是纯增加操作.纯增加操作是指操作仅对核心制品做增加元素的操作,不做删除或替换元素的操作.其他增量式方法中,修改操作类型是纯增加操作的方法还包括:基于 TDL 的用况模型导出方法[44],用于增量式导出产品的用况模型;模型驱动的行为制品导出方法[34],用于增量式导出软件产品的状态机图和序列图;基于 VML 的体系结构导出方法[37,38],用于增量式导出产品的构件

图;以特征为单位的代码导出方法[35],用于增量式导出软件产品的代码;基于 XVCL 的产品导出方法[40,41],用于增量式导出软件产品的用例、构件图和代码;在基于增量的产品导出方法[39]中,用于导出产品代码的方法;基于特征模型实现建模的产品导出方法[36]和面向特征的产品开发方法(FODM)[12,42,43],用于导出产品的构件图和代码.

在其他增量式方法中,对核心制品施加的修改操作,除了增加操作,还包括删除、替换核心制品中元素的操作.这样的方法包括:基于变化集的体系结构管理方法[32,33],用于增量式导出产品的构件图;基于增量的产品导出方法[39]中,用于导出产品类图的方法;基于逐步精化思想的产品导出方法(AHEAD)[31],用于增量式导出多种目标制品.

Id: SC01
Description: Proceed to purchase

| Id | User action | System response |
|---|---|---|
| P1 | Fill in the requested information and select the proceed option | Request the shipping method and address |
| P2 | Select one of the available shipping methods, fill in the destination address and proceed | Calculate the shipping costs |
| P3 | Confirm the purchase | Execute the order and send a request to the delivery system to dispatch the products |

(a)

Id: ADV01
Description: Buy a specific product
Before: P1

| Id | User action | System response |
|---|---|---|
| B1 | Select the buy product option | Present the selected product. The user can change the quantity of items he wants to buy. Calculate and show the amount to be paid |
| B2 | Select the confirm option | Request payment information |

(b)

Fig.4 An example of the use case fragments in MSVCM

图 4 MSVCM 方法中的用况片段实例

### 3.3 追踪关系组织方式分类结果及方法介绍

根据追踪关系组织方式对现有研究分类的结果见表 2.在所综述的 29 篇文献中,有的文献仅详述了目标制品导出过程,而没有详细叙述追踪关系的构建方法,且通过论文内容无法推断出所使用的追踪关系组织方式,如基于规则的用况模型-特征模型自动映射方法[18]、以特征为单位的代码导出方式[35]等,因而这些方法仅在表 1 中出现,而没有出现在表 2 中.

**Table 2** Classification of the approaches according to the method of organizing traceability

**表 2** 根据追踪关系组织方式对现有研究分类的结果

| 追踪关系组织方式 | 方法名称 |
|---|---|
| 反向式 | 1. 基于预处理的代码导出方法(GenArch)[19−21]<br>2. 基于模板的特征模型映射方法[22]<br>3. 基于 Kutulu 的体系结构导出方法[23]<br>4. 基于颜色标注的代码导出方法(CIDE)[27]<br>5. 基于增量的产品导出方法[39]中导出代码的方法<br>6. 基于模型转换的产品导出方法[28,29]<br>7. 基于 XVCL 的产品导出方法[40,41]<br>8. 基于变化性建模框架的产品导出方法(COVAMOF)[48] |
| 正向式 | a. 基于片段的用况模型导出方法(MSVCM)[30]<br>b. 领域用况模型建模管理方法(PLUSS)[24,25]<br>c. 基于变化集的体系结构管理方法[32,33]<br>d. 基于特征模型实现建模的产品导出方法[36]<br>e. 基于 VML 的体系结构导出方法[37,38]<br>f. 基于增量的产品导出方法[39]中导出类图的方法<br>g. 面向特征的产品开发方法(FODM)[12,42,43]<br>h. 基于 TDL 的用况模型导出方法[44] |

下面对所综述的方法进行详细介绍.

3.3.1 使用反向式组织追踪关系的方法

基于 XVCL 的产品导出方法[40,41](表 2 中方法 7)是增量式导出目标制品的方法,该方法专门设计了一种文本标记语言 XVCL,用于在核心制品和修改中直接标注出特征模型和这些可复用制品间的追踪关系.图 5 是该方法的一个实例.图 5(a)是核心代码的片段,图 5(b)是一个对核心代码的修改.标记〈x-frame〉用以对修改进行命名,修改的名字为 InformDispatcher,其对核心代码的修改内容为增加函数 SendInfo.〈option value〉用于标记追踪关系中特征的绑定状态,〈adapt x-frame〉用于标记在当前位置施加的修改的名称.〈option value〉和〈adapt x-frame〉配套使用,表示当特征 SEPARATED 被选中时,就将名为 InformDispatcher 的修改施加在当前位置,即,将函数 SendInfo 的代码插入到当前位置.可以看到:在该方法中,追踪关系直接标注在核心代码和修改中,追踪关系完全依照核心代码和修改(可复用制品)中代码的结构进行分布.所以,该方法使用的是反向式追踪关系组织方式.此外,XVCL 标记语言还被应用在基于增量的产品导出方法中[39],用以记录特征和代码间的追踪关系.因此,在基于增量的产品导出方法中[39],特征模型和代码间的追踪关系是采用反向式进行组织的.同时,在基于增量的产品导出方法中[39]中,特征模型和设计模型间的追踪关系是采用正向式进行组织的.所以,基于增量的产品导出方法[39]同时出现在表 2 的反向式和正向式两栏中.

```
import java.util.*;
public class CADCreateTask {
   private Caller aCaller;
   private Task aTask;
〈select option="CT-DISP"〉
   〈option value="SEPARATED"〉
      〈adapt x-frame="InformDispatcher"/〉
   〈/option〉
…
```

(a)

```
〈x-frame name="InformDispatcher",
language="iava"〉
   public bool SendInfo(){
   …
}

〈/x-frame〉
```

(b)

Fig.5 An example of XVCL

图 5 XVCL 方法的一个实例

基于预处理的代码导出方法(GenArch)[19−21](表 2 中方法 1)是减量式导出产品代码的方法,该方法仅适用于导出以 Java 语言编写实现的软件产品.与 XVCL 类似,GenArch 也采用了标记语言描述特征和代码间的追踪关系.GenArch 方法使用如"@Feature(name="F1")的标记对面向整个领域的代码中的每个 class,interface 进行标注."@Feature(name="F1")表示的是:如果特征 F1 被选中,那么当前被标注的 class 或 interface 被保留;反之,则被删除.在 GenArch 方法中,追踪关系是直接标注于领域代码(可复用制品)当中的.所以,该方法使用的是反向式追踪关系组织方式.

在方法 7、方法 1 这两种方法中,追踪关系均以文本形式进行描述.还有几种反向式方法,其中的追踪关系采用图形化的形式进行表示.

基于颜色标注的代码导出方法(CIDE)[27]使用颜色来标注追踪关系,该方法提出:为每个特征分配一种颜色,并使用特征的颜色来对与特征存在追踪关系的领域代码进行染色标注.在该方法中,特征模型和领域代码之间的追踪关系,也是直接标记在了代码(可复用制品)上.因此,该方法使用的是反向式追踪关系组织方式.

基于模板的特征模型映射方法[22]是减量式导出目标制品的方法,在该方法中,追踪关系通过标记在领域制品元素上特征的逻辑表达式来描述.在自动导出目标制品的过程中,根据给定的特征模型定制结果,对领域目标制品元素上特征的逻辑表达式的真值进行计算.如果真值为真,那么就保留所标注的领域制品元素;反之,则去掉制品元素.通过从领域制品中去掉逻辑表达式为假的元素,便得到目标制品.图 6 是使用该方法描述特征模型与用况间追踪关系的一个实例,在该实例中:领域用况描述的是顾客在电商网站结账时与软件系统的交互内容,领域用况内的交互采用活动图的形式进行描述;特征逻辑表达式中的特征使用数字表示,没有标识数字的领域用况元素是必选元素,这些元素在当前用况被选中时就会被保留下来.在图 6 所示的实例中,如果特征模型定制

结果为选中特征 1~特征 3 这 3 个特征,去掉特征 4、特征 5 两个特征,那么在自动导出产品用况时,就从图 6 所示的领域用况中去掉标记 4、5 的活动,并对活动间关系作一些后续处理,就得到特征模型定制结果对应的用况.

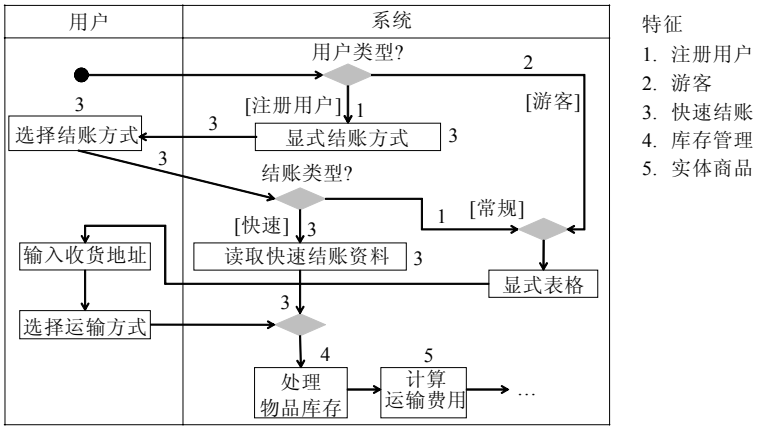


Fig.6 An example of the approach to mapping features to other models based on templates
图 6 基于模板的特征模型映射方法的一个实例

在基于 Kutulu 的体系结构导出方法中[23],特征和领域构件图之间的追踪关系采用 Kutulu 语言进行描述. Kutulu 是一种图形化的追踪关系描述语言,在支持该方法的工具中,使用 Kutulu 描述追踪关系,就是在构件图视图中新建一个带有特征名字的矩形图元,然后将该矩形图元和构件图元素的图元使用直线连接,表明矩形图元中的特征和该体系结构元素存在追踪关系.

在基于模型转换的产品导出方法中[28,29],领域目标制品均采用支持 UML 的图形化建模工具进行建模.在构建好领域制品后,再使用 UML 中的注释图元将领域制品元素对应的特征信息标注在元素上,记录下特征模型和领域制品间的追踪关系.

基于变化性建模框架的产品导出方法(COVAMOF)[48]是包括变化性建模、软件产品自动导出在内的一套软件复用方法.在建立追踪关系时,该方法并没有直接建立特征模型和可复用制品间的追踪关系,而是通过中间模型,间接地建立特征模型和可复用制品间的追踪关系.具体来说,该方法首先提出了一种类似于特征模型的变化性建模机制[49],并使用这种机制分别对需求、设计、实现这 3 个层次中的可复用软件制品进行共性变化性建模,为需求、设计、实现这 3 个层次各自构建出一个变化性模型.同时,该变化性建模机制还提供了一种名为 realization(实现)的关系,专门用于建模设计层变化性模型到需求层变化性模型、实现层变化性模型到设计层变化性模型之间的追踪关系.这样,在对需求层次的变化性模型进行定制之后,根据 realization 关系,可导出设计、实现层次的变化性模型定制结果,但这时还没有得到软件产品的设计、实现制品.根据 COVAMOF 方法的描述,该方法所使用的可复用的设计和实现制品都以文本的形式进行描述,并都已在制品中事先人工标注了制品的变化性信息,而设计和实现层次的变化性模型是根据人工标注的制品变化性信息自动抽取出来的.由此可以看出:事实上,标注在可复用设计和实现制品中的变化性信息,就是设计、实现层次的变化性模型和制品间的追踪关系,该追踪关系可用于从设计、实现层次的变化性模型定制结果最终导出软件产品的设计和实现制品.另外,由于需求层次的变化性模型实际上就是特征模型,所以 COVAMOF 方法并没有直接建立特征模型和可复用制品间的追踪关系,而是建立了特征模型和可复用制品的变化性模型间的追踪关系(realization 关系)以及可复用制品的变化性模型和可复用制品间的追踪关系.在该方法的支持工具中,需求、设计、实现层次的变化性模型及它们之间的追踪关系(realization)均采用图形化方式进行描述;设计、实现层次的变化性模型和可复用的设计、实现制品间的追踪关系,是以文本形式标注在制品当中的.因此,COVAMOF 方法同时采用了图形和文本的形式描述追踪关系.

3.3.2 使用正向式组织追踪关系的方法

领域用况模型建模管理方法(PLUSS)[24,25](表 2 中方法 b)是减量式导出软件产品用况模型的方法,该方法已在第 3.2.1 节中介绍过.在该方法中,每个领域用况交互都与一个特征建立追踪关系.对于上述追踪关系,该方法没有明确说明描述方式,但从上面提及的文献内容可以推断出,上述追踪关系是可以采用表格的形式进行描述的,即:在图 2 中表格的左侧增加一列,新列中的每一行用于记录与该行交互对应的特征.事实上,表格中的每一行其实就是一条追踪关系.由于每条追踪关系中的特征信息各不相同,所以该方法中追踪关系的组织方式,相当于根据特征绑定状态的异同对追踪关系进行聚集和分离.因此,该方法是正向式组织追踪关系的方法.

基于片段的用况模型导出方法(MSVCM)[30](表 2 中方法 a)是增量式导出产品用况模型的方法,该方法已在第 3.2.2 节中介绍过.MSVCM 使用一种简单的语言描述特征和用况片段间的追踪关系,但并未对所使用的语言进行说明,仅仅解释了使用该语言描述的追踪关系实例的语义.第 3.2.2 节中的图 3 是该方法中追踪关系的实例,可以看出,在该方法中,特征绑定状态相同的追踪关系是聚集在一起的.所以,该方法是正向式组织追踪关系的方法.

基于 VML 的体系结构导出方法[37,38](表 2 中方法 e)是增量式导出软件产品构件图的方法,在该方法中,特征模型和对核心构件图的修改之间的追踪关系使用 VML 语言进行描述.图 7 是一个使用 VML 语言描述追踪关系的实例,图 7 中的实例描述的是特征 Keypad 与修改之间的追踪关系.追踪关系的内容是:在 Keypad 特征被选中时,就通过 IAccess,IRegister 接口连接 KeypadReader 和 LockControlMng 两个构件等等.可以看到,在该方法的追踪关系中,特征绑定状态相同的追踪关系是聚集在一起的.所以,该方法是正向式组织追踪关系的方法.

```
// Variability description of the Control Lock SPL architecture
00 Concern LockControl {
01  VariationPoint AuthenticationDevice {
02   Kind: alternative ;
03   Variant Keypad {
04     SELECT:
05       connect(KeypadReader,LockControlMng) using interface(IAccess);
06       connect(KeypadReader,LockControlMng) using interface(IRegister);
07       connect(LockControlMng,KeypadAuth) using interface(IVerify);
08       connect(KeypadAuth,LockControlMng) using interface(IRegister);
09     UNSELECT:
10       remove(KeypadReader);
11       remove(KeypadAuth);}
12       ...
```

Fig.7 An example of the traceability specified in VML

图 7 使用 VML 描述追踪关系的实例

基于 TDL 的用况模型导出方法[44](表 2 中方法 h)是增量式导出产品用况模型的方法,该方法提出一种语言 TDL,用于描述特征模型和对核心用况模型的修改之间的追踪关系.TDL 语言的形式与 VML 语言比较相似,将与同一特征绑定状态相关的修改聚集在一起描述,以不同特征绑定状态来区分不同的追踪关系条目.所以,该方法也是正向式追踪关系组织方法.

方法 b、方法 a、方法 e、方法 h 均采用文本的形式描述特征模型和可复用制品间的追踪关系,以下几种方法采用图形化的方式描述特征模型和可复用制品间的追踪关系.

基于变化集的体系结构管理方法[32,33]提供了工具 EASEL 来建模追踪关系,该方法中的变化集指的是,为实现一个被选中特征的功能所需对核心体系结构施加的修改.在 EASEL 的支持下,特征和变化集之间的对应关系以视图对应的形式进行描述,即:在特征模型视图中选择一个特征,便可在变化集视图中查看、编辑被选择特征对应的变化集内容.图 8 是 EASEL 工具的截图.图 8 左侧的 ArchStructure 视图就是变化集视图,右侧的 Variability 视图就是特征模型视图.截图中的内容是:如果特征 Baseline 被选中,那么就从核心体系结构中删除构件 CD Reader,删除构件 CD Reader 与构件 Sound Source 之间的交互,删除构件 CD Reader 与构件 Sound Source 交互时使用的接口,增加构件 CD Reader/Writer,为构件 CD Reader/Writer 增加两个接口,并通过这两个接口建立 CD Reader/Writer 和构件 Sound Source 之间的交互.EASEL 在对追踪关系进行组织时,将相同特征绑定状态的追踪

关系聚集在一起.因此,该方法采用的是正向式追踪关系组织方式.与该方法相似,基于特征模型实现建模的产品导出方法[36]、FODM(面向特征的产品开发方法)[12,42,43]和基于增量的产品导出方法[39]中类图的导出方法,也使用视图对应的形式进行追踪关系的描述,即:在特征模型视图中选择一个特征,便可在另一视图中对被选中特征对应的修改进行查看和编辑.因此,这 3 种方法也是正向式追踪关系组织方法.

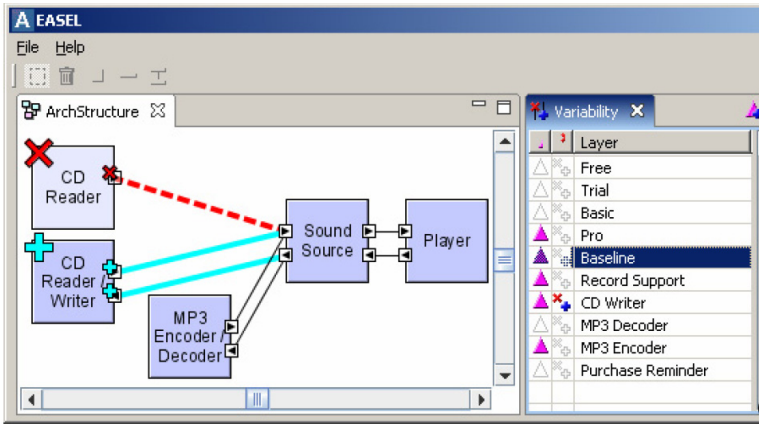


Fig.8 A snapshot of EASEL

图 8 EASEL 工具的截图

需要指出的是,正向式和反向式这两种追踪关系的组织方式并不是互斥的.由于二者是针对于相同内容的不同组织方式,两种方式是可以互相转化的.文献[50]介绍的工作就是将采用反向式组织的追踪关系转换成正向式组织的工作.

还有一些方法,如基于模型转换的类图导出方法[26]、基于多模型的体系结构导出方法[51,52],均采用模型转换描述语言(如 QVT-R[45],ATL[53]等)来描述追踪关系.这些方法并未指明追踪关系的组织方式.事实上,以模型转换描述语言描述的追踪关系,可以采用正向、反向式中的任意一种方式进行组织.

## 4 问题与展望

本节针对现有基于特征模型自动导出软件产品研究中存在的问题与不足,提出可能的研究设想.

1. 导出不同种类目标制品需使用不同方法,缺乏较为通用的目标制品导出方法.

从对相关研究的介绍中可以看到,导出不同种类的目标制品可能需要使用到不同的方法.这会给开发人员带来很大的学习负担,也加大了已导出目标制品的维护负担.同时,对于已经存在的方法,有时也无法直接使用.例如,开发人员想要实现用况模型的自动导出,而当前项目中所使用的用况模型元模型,和已有任一方法中所使用的用况模型元模型均不完全相同,开发人员必须对已有方法进行调整再加以使用,这又额外增加了开发人员的负担.引起上述问题的原因在于:现有方法大多仅针对特定种类目标制品的特定元模型进行设计,而没有从更高的层次上来看待和解决导出问题,致使方法仅适用于某种目标制品的某种特定形式.其实,对于采用相同目标制品导出方式的方法来说,目标制品元模型和追踪关系描述方式的异同,其实仅是表面的异同,而方法的本质却是一致的.例如,采用增量式目标制品导出方式,那么对于任何种类的任何形式的目标制品,实际上都是进行如下几步工作.

(1) 开发人员确定目标制品的元模型.这一步要确定目标制品中包含哪些元素,每种元素的属性是什么,元素间关系如何;

(2) 开发人员构建核心制品,或设核心制品为空;

(3) 开发人员构建特征模型和修改之间的追踪关系.这一步中,开发人员要为不同的特征绑定状态构建

该绑定状态下需要对核心制品施加的修改,并采用一种描述机制记录特征绑定状态与修改的对应关系.在增量式方法中,修改所包含的操作总共分为 3 类:增加、删除和替换.有的增量式方法仅包含增加操作,有的增量式方法还酌情包含了删除和替换操作.无论增量式方法中包含了几种修改操作,同种修改操作在不同方法中的语义实际上都基本相同.增加操作用于说明:增加的目标制品元素是什么、元素的各属性取值以及所增加元素与已存在的其他元素之间的关系如何.删除操作用于说明:被删除的元素是什么.替换操作用于说明:被替换的元素或元素属性取值是什么、替换后新的元素或元素属性取值是什么;

(4) 在前 3 步工作完成后,就可以根据给定的特征模型定制结果,根据第(3)步中构建的追踪关系,将与定制结果中特征绑定状态对应的修改施加在核心制品上,导出目标制品.这一步的导出过程可以在工具支持下自动完成.

由上述过程可以看到:目标制品元模型和追踪关系描述方式其实不会影响导出方法的步骤,而仅会给步骤带来细节差异.因此,可以为导出不同种类目标制品提供一个通用的方法框架.框架中明确给出实现目标制品自动导出的步骤(如上述步骤(1)~步骤(4),但更加详细),还可以对追踪关系描述方式进行一定程度的统一,给出追踪关系描述机制的参考样式以及如何根据目标制品的特点对描述机制进行定制.这样,在导出不同种类目标制品时,开发人员可根据所需导出目标制品的特点,根据框架的内容配置出所需的方法,实现目标制品的自动导出.由于所有方法均是通过框架配置得出,方法之间的相似性也得到保证.

2. 在增量式导出目标制品的过程中,修改的施加顺序不同可能造成目标制品有所不同.

对于软件产品的自动导出方式,如果采用减量式的导出方式,随着特征模型规模的扩大,构建和维护面向整个领域的可复用领域制品会变得越来越困难;相比之下,增量式比减量式更具优势.但是,增量式导出方式也有其固有的问题,即:对核心制品施加的修改集合相同,但修改的施加顺序不同,可能得到不同的目标制品.这个问题在软件产品自动导出的研究中并没有引起太多重视,Istoan 等人[34]对这个问题有所提及;Clarke 等人[54]对这个问题进行了形式化描述,但其并未给出检测和解决方法.

要解决上述问题,可以从追踪关系入手,通过对追踪关系内容的检查来避免上述问题的发生.具体来说:

- 首先需要将追踪关系形式化地描述出来,描述出在何种应用条件下(特征绑定状态下),将修改施加在核心制品上以导出目标制品;
- 在有了形式化描述的追踪关系后,对追踪关系进行检查,检查的准则是:查找是否存在两个或多个这样的修改,它们的应用条件可同时为真,它们对目标制品中同一元素进行不同的修改操作,且没有固定的使用顺序.若不存在这样的修改,则在导出目标制品的过程中,即便修改的施加顺序不同,导出的目标制品也均是相同的;若存在这样的修改,则将这个情况反馈给开发人员,由开发人员决定是否需要对追踪关系和可复用软件制品进行修改.

3. 追踪关系的形式化描述机制有待改进.

在现有的基于特征模型自动导出软件产品的方法中,追踪关系的描述机制有很多种.这些机制在形式上体现出如下两种特点:(1) 追踪关系直接标注在可复用制品中,与可复用制品混合为一体,如基于模板的特征模型映射方法[22]、基于 XVCL 的产品导出方法[40,41];(2) 追踪关系与可复用制品相互独立,如基于 VML 的体系结构导出方法[37,38]、几种使用模型转换语言描述追踪关系的方法[51,52].如果将追踪关系直接标注于可复用制品中,追踪关系通常会和可复用制品的内容相互交错,给开发人员高效维护追踪关系和可复用制品带来阻碍.例如,开发人员要对可复用制品进行维护,而在阅读理解可复用制品和对制品进行修改时,追踪关系会在一定程度上分散开发人员的注意力,需要开发人员额外花费精力屏蔽掉当前不关注的追踪关系信息.而若将追踪关系与可复用制品独立记录,则可避免上述问题.目前,现有研究中使用的追踪关系独立记录机制仅有 VML 语言和几种模型转换描述语言.其中,VML 仅能描述特征模型和体系结构间的追踪关系;模型转换语言虽可用于描述特征模型和其他多种可复用软件制品间的追踪关系,但使用模型转换语言需要开发人员事先掌握模型转换的相关知识内容,这给追踪关系的描述工作带来很大负担.所以,若要将追踪关系独立于可复用制品进行记录,仍然需要

根据可复用制品本身的特点,有针对性地提出简洁、易用的追踪关系描述机制.

要设计新的描述机制,还需要考虑到的一点是:为实现软件产品的自动导出,为所设计的追踪关系描述机制提供支持工具也是必须完成的工作.通常,为图形化描述机制开发支持工具会消耗较长的时间,提供文本形式的追踪关系描述语言作为描述机制并为其提供编译环境是比较经济的做法.对于追踪关系语言的设计,可以参考VML 语言的形式,将追踪关系描述为一组追踪规则.每条追踪规则分为两个部分,分别是使用条件和操作序列.使用条件是一个特征的逻辑描述式,描述的是实施操作序列中操作的先决条件.操作序列描述的是对可复用制品的操作内容,在设计操作语句时,首先确定下来可复用制品的形式以及所需施加在其上的操作,然后为描述可复用制品和操作设计所需表达语句即可.当给定一个合法的特征模型定制结果后,对于每一条追踪规则,判断其使用条件的真假,仅保留使用条件为真的追踪规则.然后,根据不同追踪规则中的操作内容,自动地计算出所保留的追踪规则的使用顺序.最后,根据计算出来的顺序逐条使用追踪规则,实施对可复用制品的操作,便可导出目标制品.上述语言设计思路对于增量式、减量式目标制品导出方式均适用.我们已对上述语言设计的思路在增量式目标制品导出方式中进行了实践,实现了基于特征模型增量式自动导出产品用况模型的过程[44].

4. 在导出目标制品的过程中未将非功能性需求纳入考量.

在基于特征模型自动导出软件产品时,特征模型定制结果是软件产品的功能性需求模型,相同的功能性需求可能对应多种不同的设计和实现策略.因此,一个特征模型定制结果可能对应多个软件产品.这些产品虽然实现的是相同的功能性需求,但性能不同,所能满足的非功能性需求也不同.所以,将非功能性需求的考量纳入到产品导出过程中,在功能性需求相同时,根据不同的非功能性需求导出相应的软件产品是十分必要的.但目前,对于产品导出过程中非功能性需求的考量,仅有文献[55–57]中的研究对这个问题进行了一些初步探索,但仍未形成系统的解决方案.因此,如何在产品自动导出过程中处理非功能性需求对所导出产品的影响,仍然是值得深入讨论的问题.

要在目标制品导出过程中考虑非功能性需求因素,最简单的做法是:在给定一个合法的特征模型定制结果后,导出实现该定制结果的所有产品,对产品的非功能性属性逐一地进行测量查看,在其中选出满足非功能性需求的产品.上述做法的问题在于:当可导出的产品数量很大时,这种寻找所需产品的做法效率较低.在可导出产品数量很大时,一种可行的做法是:获取单个特征对应的不同实现制品的非功能性属性值,在给定特征模型定制结果后,根据定制结果中所包含的特征集合,估算出不同实现产品的非功能性属性值.在第 2 种做法中,需要解决的问题是:如何获取单个特征对应的实现制品的非功能性属性值以及根据何种计算规则利用所获取的数据进行估算.对于这两个问题,需要根据非功能性属性的特点,分别设计解决方案.

5. 针对特定领域的软件复用研究与自适应系统研究可进一步结合.

针对特定领域的软件复用通过将变化性引入到软件产品的开发中,使开发人员通过定制便可快速导出分别满足不同需求的产品.在上述过程中,开发中引入的变化性增加了开发所得产品的灵活性.这个以变化性换取灵活性的思想,在自适应系统领域也得到了应用.动态软件产品线[58,59]就是将针对特定领域的软件复用方法应用到自适应系统的实现中去的研究.应用方式具体来说,就是[60]:

- 使用特征模型建模自适应系统在不同环境中所需满足的所有需求,然后构建特征模型和软件设计、实现制品间的追踪关系.这样,当自适应系统运行在某一环境状态中时,系统所需满足的需求就是特征模型表示的所有需求的一个子集,即,一个特征模型定制结果.此时,自适应系统所需运行状态可以根据特征模型定制结果和追踪关系自动配置出来;
- 当系统运行环境发生改变并导致系统需求发生变化时,系统的需求便从一个特征模型定制结果转变为另一个特征模型定制结果.这时,只要根据新的特征模型定制结果和追踪关系,将自适应系统自动切换到新特征模型定制结果对应的系统配置状态,便实现了系统状态的自动切换.

将针对特定领域的软件复用方法应用在自适应系统的研究中,可以为解决自适应系统研究中的一些问题开辟新的视角.但目前,动态软件产品线的相关研究成果并不多见.如何将特定领域软件复用方法和自适应系统研究二者进行更好的结合,仍然是值得深入讨论的问题.

对于二者的结合,除了可以如文献[60]中提到的方法,将产品自动导出方法应用于自适应系统的实现,还可以从其他方面进行考虑,如将特定领域软件复用中建模方法应用于自适应系统建模.例如,可以将特征模型同时用于建模自适应系统运行环境和不同环境中系统所需满足的需求,再将运行环境特征模型与表示需求的特征模型以特征间约束关系建立关联,描述特定环境中系统的需求是什么,以此完成自适应系统需求与环境之间关系的建模.其他方面,针对本节问题 4 的研究成果也可以应用于自适应系统研究中,用于自适应系统根据非功能性需求,如当前运行环境中的资源限制,找出自动调整自身行为的策略.其实,特定领域软件复用研究和自适应系统研究均需在开发过程中考虑产品的可变性.二者的区别在于:特定领域软件复用研究中,产品的定制方案确定时间多为编译前;而自适应系统研究中,产品定制方案确定时间为运行时.因此,在上述两种研究中,其中一种研究的成果经过调整,就有可能应用在另一种研究中.

## 5 结 论

本文对现有基于特征模型自动导出软件产品的研究进行了综述.首先,提出了一个自动导出软件产品方法的分类框架;然后,使用该框架对现有基于特征模型自动导出软件产品的方法进行了分析;最后,基于对现有方法的分析,指出现有研究中存在的问题和不足,提出了解决这些问题和不足的设想,还给出了与软件产品自动导出相关的未来的研究方向.

**致谢** 在此,我们向对本文的工作给予支持和建议的老师和同学表示感谢.

**References**:

[1] Li KQ, Chen ZL, Mei H, Yang FQ. An outline of domain engineering. Computer Science, 1999,26(5):21−25 (in Chinese with English abstract).

[2] Deelstra S, Sinnema M, Bosch J. Experiences in software product families: Problems and issues during product derivation. In: Proc. of the 3rd Int'l Software Product Line Conf. Boston: Springer-Verlag, 2004. 165−182. [doi: 10.1007/978-3-540-28630-1_10]

[3] Deelstra S, Sinnema M, Bosch J. Product derivation in software product families: A case study. Journal of Systems and Software, 2005,74(2):173−194. [doi: 10.1016/j.jss.2003.11.012]

[4] Rabiser R, Grünbacher P, Dhungana D. Requirements for product derivation support: Results from a systematic literature review and an expert survey. Information and Software Technology, 2010,52(3):324−346. [doi: 10.1016/j.infsof.2009.11.001]

[5] Czarnecki K, Helsen S. Feature-Based survey of model transformation approaches. IBM Systems Journal, 2006,45(3):621−645. [doi: 10.1147/sj.453.0621]

[6] Yue T, Briand LC, Labiche Y. Facilitating the transition from use case models to analysis models: Approach and experiments. ACM Trans. on Software Engineering and Methodology, 2013,22(1):5−34. [doi: 10.1145/2430536.2430539]

[7] Gotel O, Cleland-Huang J, Hayes JH, Zisman A, Egyed A, Grünbacher P, Antoniol G. The quest for ubiquity: A roadmap for software and systems traceability research. In: Proc. of the 20th Int'l Requirements Engineering Conf. Chicago: IEEE, 2012. 71−80. [doi: 10.1109/RE.2012.6345841]

[8] Winkler S, Pilgrim J. A survey of traceability in requirements engineering and model-driven development. Software and Systems Modeling, 2010,9(4):529−565. [doi: 10.1007/s10270-009-0145-0]

[9] Kang KC, Cohen SG, Hess JA, Novak WE, Peterson AS. Feature-Oriented domain analysis (FODA) feasibility study. Technical Report, CMU/SEI-90-TR-21-ESD-90/TR-222, Pittsburgh: Carnegie-Mellon University Software Engineering Institute, 1990.

[10] Kang KC, Kim S, Lee J, Kim K, Kim GJ, Shin E. FORM: A feature-oriented reuse method with domain specific reference architecures. Annals of Software Engineering, 1998,5(1):143−168.

[11] Griss ML, Favaro J, d'Alessandro M. Integrating feature modeling with the RSEB. In: Proc. of the 5th Int'l Conf. on Software Reuse. Victoria: IEEE, 1998. 76−85. [doi: 10.1109/ICSR.1998.685732]

[12] Zhang W. Research on feature-oriented domain modeling [Ph.D. Thesis]. Peking: Peking University, 2006 (in Chinese with English abstract).

[13] Gomaa H, Shin ME. Automated software product line engineering and product derivation. In: Proc. of the 40th Annual Hawaii Int'l Conf. on System Sciences. Waikoloa: IEEE, 2007. 285−292. [doi: 10.1109/HICSS.2007.95]

[14] Botterweck G, Lee K, Thiel S. Automating product derivation in software product line engineering. In: Münch J, Liggesmeyer P, eds. Proc. of the Software Engineering Conf. Kaiserslautern: GI-Edition, 2009. 177−182.

[15] Krueger CW. The 3-tiered methodology: Pragmatic insights from new generation software product lines. In: Proc. of the 11th Int'l Software Product Line Conf. Kyoto: IEEE, 2007. 97−106. [doi: 10.1109/SPLINE.2007.34]

[16] Krueger CW, Clements P. Systems and software product line engineering with BigLever software gears. In: Proc. of the 16th Int'l Software Product Line Conf. Salvador: ACM Press, 2012. 256−259. [doi: 10.1145/2364412.2364458]

[17] Chaves FJE. Dynamic and automated product derivation for consumer electronics software applications. IEEE Trans. on Consumer Electronics, 2013,59(4):883−891. [doi: 10.1109/TCE.2013.6689703]

[18] Braganca A, Machado RJ. Automating mappings between use case diagrams and feature models for software product lines. In: Proc. of the 11th Int'l Software Product Line Conf. Kyoto: IEEE, 2007. 3−12. [doi: 10.1109/SPLINE.2007.17]

[19] Cirilo E, Kulesza U, de Lucena CJP. A product derivation tool based on model-driven techniques and annotations. Journal of Universal Computer Science, 2008,14(8):1344−1367.

[20] Cirilo E, Kulesza U, Coelho R, de Lucena CJP, Staa A. Integrating component and product lines technologies. In: Proc. of the 5th Int'l Conf. on Software Reuse. Beijing: Springer-Verlag, 2008. 130−141. [doi: 10.1007/978-3-540-68073-4_12]

[21] Cirilo E, Nunes I, Kulesza U, Lucena CJP. Automating the product derivation process of multi-agent systems product lines. Journal of Systems and Software, 2012,85(2):258−276. [doi: 10.1016/j.jss.2011.04.066]

[22] Czarnecki K, Antkiewicz M. Mapping features to models: A template approach based on superimposed variants. In: Proc. of the 4th Int'l Generative Programming and Component Engineering. Olavi Saal: Springer-Verlag, 2005. 422−437. [doi: 10.1007/1156 1347_28]

[23] Dayibas O, Oguztuzun H. Kutulu: A domain-specific language for feature-driven product derivation. In: Proc. of the 36th Computer Software and Applications Conf. (COMPSAC). Swissotel Grand Efes Izmir: IEEE, 2012. 105−110. [doi: 10.1109/COMPSAC.2012. 20]

[24] Eriksson M, Börstler J, Borg K. The PLUSS approach-domain modeling with features, use cases and use case realizations. In: Proc. of the 8th Software Product Lines. Rennes: Springer-Verlag, 2005. 33−44. [doi: 10.1007/11554844_5]

[25] Eriksson M, Borg K, Börstler J. Use cases for systems engineering—An approach and empirical evaluation. Journal of Systems Engineering, 2008,11(1):39−60. [doi: 10.1002/sys.20087]

[26] Haugen Ø, Møller-Pedersen B, Oldevik J, Solberg A. An MDA®-based framework for model-driven product derivation. In: Hamza MH, ed. Proc. of the IASTED Conf. on Software Engineering and Applications. MIT Cambridge: IASTED/ACTA Press, 2004. 709−714.

[27] Kästner C, Apel S, Kuhlemann M. Granularity in software product lines. In: Proc. of the 30th Int'l Conf. on Software engineering. Leipzig: ACM Press, 2008. 311−320. [doi: 10.1145/1368088.1368131]

[28] Tawhid R, Petriu DC. Integrating performance analysis in the model driven development of software product lines. In: Proc. of the 11th Model Driven Engineering Languages and Systems. Toulouse: Springer-Verlag, 2008. 490−504. [doi: 10.1007/978-3-540-87875-9_35]

[29] Tawhid R, Petriu DC. Product model derivation by model transformation in software product lines. In: Proc. of the 14th Int'l Symp. on Object/Component/Service-Oriented Real-Time Distributed Computing Workshops (ISORCW). Newport Beach: IEEE, 2011. 72−79. [doi: 10.1109/ISORCW.2011.18]

[30] Bonifácio R, Borba P. Modeling scenario variability as crosscutting mechanisms. In Proc. of the 8th ACM Int'l Conf. on Aspect-Oriented Software Development. Virginia: ACM Press, 2009. 125−136. [doi: 10.1145/1509239.1509258]

[31] Don B, Sarvela JN, Rauschmayer A. Scaling step-wise refinement. IEEE Trans. on Software Engineering, 2004,30(6):355−371. [doi: 10.1109/TSE.2004.23]

[32] Hendrickson SA, van der Hoek A. Modeling product line architectures through change sets and relationships. In: Proc. of the 29th Int'l Conf. on Software Engineering. IEEE, 2007. 189−198. [doi: 10.1109/ICSE.2007.56]

[33] Hendrickson SA, Subramanian S, van der Hoek A. Multi-Tiered design rationale for change set based product line architectures. In: Proc. of the 3rd Int'l Workshop on Dharing and Reusing Architectural Knowledge. Leipzig: ACM Press, 2008. 41−44. [doi: 10.1145/1370062.1370073]

[34] Istoan P, Biri N, Klein J. Issues in model-driven behavioural product derivation. In: Proc. of the 5th Workshop on Variability Modeling of Software-Intensive Systems. Pisa: ACM Press, 2011. 69−78. [doi: 10.1145/1944892.1944900]

[35] Jansen AGJ, Smedinga R, van Gurp J, Bosch J. First class feature abstractions for product derivation. IET Software, 2004,151(4): 187−197. [doi: 10.1049/ip-sen:20040922]

[36] Peng X, Shen L, Zhao W. Feature implementation modeling based product derivation in software product line. In: Proc. of the 5th Int'l Conf. on Software Reuse. Beijing: Springer-Verlag, 2008. 142−153. [doi: 10.1007/978-3-540-68073-4_13]

[37] Sánchez P, Loughran N, Fuentes L, Garcia A. Engineering languages for specifying product-derivation processes in software product lines. In: Proc. of the 1st Int'l Conf. on Software Language Engineering. Toulouse: Springer-Verlag, 2009. 188−207. [doi: 10.1007/978-3-642-00434-6_13]

[38] Zschaler S, Sánchez P, Santos J, Alférez M, Rashid A, Fuentes L, Moreira A, Araújo J, Kulesza U. VML*——A family of languages for variability management in software product lines. In: Proc. of the 2nd Int'l Conf. on Software Language Engineering. Denver: Springer-Verlag, 2010. 82−102. [doi: 10.1007/978-3-642-12107-4_7]

[39] Schaefer I, Worret A, Poetzsch-Heffter A. A model-based framework for automated product derivation. In: Proc. of the 1st Int'l Workshop on Model-Driven Approaches in Software Product Line Engineering. 2009. 14−21.

[40] Jarzabek S, Bassett P, Zhang H, Zhang W. XVCL: XML-Based variant configuration language. In: Proc. of the 25th Int'l Conf. on Software Engineering. Portland: IEEE, 2003. 810−811. [doi: 10.1109/ICSE.2003.1201298]

[41] Zhang H, Jarzabek S. XVCL: A mechanism for handling variants in software product lines. Science of Computer Programming, 2004,53(3):381−407. [doi: 10.1016/j.scico.2003.04.007]

[42] Zhang W, Mei H, Zhao H, Yang J. Transformation from CIM to PIM: A feature-oriented component-based approach. In: Proc. of the 8th Model Driven Engineering Languages and Systems. Montego Bay: Springer-Verlag, 2005. 248−263. [doi: 10.1007/11557432_18]

[43] Fan B. Design and implementation of a feature-oriented domain engineering supporting tool [MS. Thesis]. Beijng: Peking University, 2012 (in Chinese with English abstract).

[44] Yu W, Zhang W, Zhao H, Jin Z. TDL: A transformation description language from feature model to use case for automated use case derivation. In: Proc. of the 18th Int'l Software Product Line Conf., Vol.1. Florence: ACM Press, 2014. 187−196. [doi: 10.1145/2648511.2648531]

[45] OMG meta object facility (MOF) 2.0 query/view/transformation. Version 1.1, 2011. http://www.omg.org/spec/QVT/1.1/PDF/

[46] Tawhid R, Petriu DC. Automatic derivation of a product performance model from a software product line model. In: Proc. of the 15th Int'l Software Product Line Conf. Munich: IEEE, 2011. 80−89. [doi: 10.1109/SPLC.2011.27]

[47] Bonifácio R, Borba P. Towards a crosscutting approach for variability management. In: Thiel S, Pohl K, eds. Proc. of the 12th Int'l Software Product Line Conf. Limerick: Lero Int. Science Centre, 2008. 353−360.

[48] Sinnema M, Deelstra S, Hoekstra P. The COVAMOF derivation process. In: Proc. of the 9th Int'l Conf. on Software Reuse. Turin: Springer-Verlag, 2006. 101−114. [doi: 10.1007/11763864_8]

[49] Sinnema M, Deelstra S, Nijhuis J, Bosch J. COVAMOF: A framework for modeling variability in software product families. In: Proc. of the 8th Software Product Line Conf. Boston: Springer-Verlag, 2004. 197−213. [doi: 10.1007/978-3-540-28630-1_12]

[50] Duran-Limon HA, Castillo-Barrera FE, Lopez-Herrejon RE. Towards an ontology-based approach for deriving product architectures. In: Proc. of the 15th Int'l Software Product Line Conf. Munich: ACM Press, 2011. 29−33. [doi: 10.1145/2019136.2019169]

[51] González-Huerta J, Abrahão S, Insfran E. Automatic derivation of AADL product architectures in software product line development. In: Delange J, Feiler PH, eds. Proc. of the 1st Architecture Centric Virtual Integration Workshop. Valencia: CEUR, 2014. 69−78.

[52] González-Huerta J, Insfran E, Abrahão S, McGregor JD. Architecture derivation in product line development through model transformations. In: Proc. of the Information System Development. Springer-Verlag, 2014. 371−384. [doi: 10.1007/978-3-319-07215-9_30]

[53] Jouault F, Kurtev I. Transforming models with ATL. In: Proc. of the Satellite Events at the MoDELS 2005 Conf. Montego Bay: Springer-Verlag, 2006. 128−138. [doi: 10.1007/11663430_14]

[54] Clarke D, Helvensteijn M, Schaefer I. Abstract delta modeling. In: Proc. of the 9th Int'l Conf. on Generative Programming and Component Engineering. Eindhoven: ACM Press, 2010. 13−22. [doi: 10.1145/1868294.1868298]

[55] González-Huerta J, Insfrán E, Abrahão S. Defining and validating a multimodel approach for product architecture derivation and improvement. In: Proc. of the 16th Int'l Conf. on Model-Driven Engineering Languages and Systems. Miami: Springer-Verlag, 2013. 388−404. [doi: 10.1007/978-3-642-41533-3_24]

[56] González-Huerta J, Insfran E, Abrahão S, McGregor JD. Non-Functional requirements in model-driven software product line engineering. In: Proc. of the 4th Int'l Workshop on Nonfunctional System Properties in Domain Specific Modeling Languages. Innsbruck: ACM Press, 2012. Article 6. [doi: 10.1145/2420942.2420948]

[57] Siegmund N, Rosenmuller M, Kuhlemann M, Kastner C, Saake G. Measuring non-functional properties in software product line for product derivation. In: Proc. of the 15th Asia-Pacific Software Engineering Conf. Beijing: IEEE, 2008. 187−194. [doi: 10.1109/APSEC.2008.45]

[58] Hallsteinsen S, Hinchey M, Park S, Schmid K. Dynamic software product lines. Journal of Computer, 2008,41(4):93−95. [doi: 10.1109/MC.2008.123]

[59] Hallsteinsen S, Hinchey M, Park S, Schmid K. Dynamic software product lines. In: Proc. of the Systems and Software Variability Management. Springer-Verlag, 2013. 253−260. [doi: 10.1007/978-3-642-36583-6_16]

[60] Capilla R. Variability realization techniques and product derivation. In: Proc. of the Systems and Software Variability Management. Springer-Verlag, 2013. 87−99. [doi: 10.1007/978-3-642-36583-6_6]

**附中文参考文献**:

[1] 李克勤,陈兆良,梅宏,杨芙清.领域工程概述.计算机科学,1999,26(5):21−25.

[12] 张伟.面向特征的领域建模技术研究[博士学位论文].北京:北京大学,2006.

[43] 樊波.一个面向特征的领域工程支持工具的设计与实现[硕士学位论文].北京:北京大学,2012.

**于文静**(1987－),女,吉林长春人,博士生,主要研究领域为需求工程,软件复用.

**张伟**(1978－),男,博士,副教授,CCF 会员,主要研究领域为需求工程,软件开发方法.

**赵海燕**(1966－),女,博士,副教授,CCF 高级会员,主要研究领域为需求工程,软件复用.

**金芝**(1962－),女,博士,教授,博士生导师,CCF 会士,主要研究领域为软件需求工程,知识工程.