面向软件产品家族的变化性建模方法*

邹盛享, 张 伟, 赵海燕+, 梅 宏

(北京大学 信息科学技术学院 软件研究所,北京 100871)

Modeling Variability in Software Product Family

ZOU Sheng-Xiang, ZHANG Wei, ZHAO Hai-Yan+, MEI Hong

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

+ Corresponding author: E-mail: +86-10-62757670, E-mail: zhhy@sei.pku.edu.cn, http://162.105.203.104/belljointlab/index.jsp

Received 2004-06-03; Accepted 2004-07-29

Zou SX, Zhang W, Zhao HY, Mei H. Modeling variability in software product family. *Journal of Software*, 2005,16(1):37~49. http://www.jos.org.cn/1000-9825/16/37.htm

Abstract: To accommodate the frequent changes of user requirements and operating environments, software systems have to be much more flexible. Fortunately, modeling variability is just a realistic and efficient approach to controlling variability and implementing software reuse, which does well not only in identifying and expressing variability, but also in assisting the management of variability evolution. This paper proposes an approach to modeling variability in product family. In this approach, variabilities of system behaviors are modeled through extended use case models, while variabilities in functionality and quality are captured by feature models, and above all, both the models adopt consistent mechanisms to model variability. Summarily, the whole modeling process of the approach is discussed systematically through a real software family on mobile phone.

Key words: software product family; variability; use case model; feature model; software reuse

摘 要: 用户需求和运行环境的变化增加了软件产品开发、维护和演化的难度.另一方面,如果能对同类软件(比如软件产品家族)的变化性实施有效的控制,则可以极大地促进软件复用,提高软件生产效率和质量.对变化性建模是控制变化性的有效手段,既有助于变化性的识别和规约,又能够提供足够的机制支持变化性的演化.提出了一种面向产品家族的变化性建模方法,以变化性的管理策略为指导,从扩展的用况(use case)模型入手捕获系统行为的变化性,以特征模型来组织功能性需求和质量属性并识别其变化性,两种模型对变化性的建模采用相同的机制.还结合一个实例讨论了产品家族变化性建模的全过程.该研究对产品家族变化性的分析与建模具

* Supported by the Key Project of the National Natural Science Foundation of China under Grant No.60233010 (国家自然科学基金重点项目); the National Grand Fundamental Research 973 Program of China under Grant No.2002CB312003 (国家重点基础研究发展规划(973)); the National Natural Science Foundation for Distinguished Young Scholars of China under Grant No.60125206 (国家杰出青年科学基金); the Major Project of Science and Technology Research of Ministry of Education of China under Grant No.MAJOR0214 (国家教育部重大资助项目); the National Research Foundation for the Doctoral Program of Higher Education of China under Grant No.20010001001 (国家教育部博士点基金)

作者简介: **邹盛享**(1977－),男,湖南澧县人,硕士,主要研究领域为变化性技术,领域工程;**张伟**(1978－),男,博士生,主要研究领域为领域工程,软件构件技术,软件复用;**赵海燕**(196－),女,博士,讲师,主要研究领域为软件复用,数据挖掘;**梅宏**(1963－),男,博士,教授,博士生导师,主要研究领域为软件工程,软件复用,软件构件技术,分布对象技术.

有一定的参考作用.

关键词: 软件产品家族;变化性;用况模型;特征模型;软件复用

中图法分类号: TP311 **文献标识码**: A

软件复用被认为是解决软件危机、实现软件产业工业化生产方式的有效途径[1].软件复用活动包含两个相关的阶段:可复用软件资产的生产阶段(development for reuse)和基于可复用软件资产的应用系统开发阶段(development with reuse)[2].领域工程对应于可复用软件资产的生产阶段,即系统地识别、开发和组织领域可复用软件资产,为后期的应用系统开发提供必备的物质和技术基础.

软件产品家族方法即是应用软件复用的相关原理和技术,把整个产品家族作为同一问题空间来看待的软件开发方法.这里的产品家族是指一类共享体系结构、特征、代码、构件、中间件或者需求的软件产品[3,4].研究和实践表明,由于很好地应用了软件复用技术,软件产品家族方法极大地节省了软件产品的开发成本和时间[5,6],软件产品家族方法也因此被认为是一种提高软件开发效率和控制软件复杂性的有效措施,因而在软件产业界得到越来越多的重视[3].

产品家族方法要实现对软件复用以及对家族产品演化的支持,必须支持对变化性的有效管理.类比于软件复用的基本过程,产品家族方法过程也可分为两个基本的阶段,即:生产家族核心资产的阶段和根据家族核心资产生产单个产品的阶段.通常,对变化性的识别、设计和实现主要体现在第 1 阶段,第 2 阶段则要对变化性进行定制、配置、绑定等.在家族核心资产的生产过程中,采用怎样的模型来组织核心资产以及对变化性进行控制和管理一直是一个备受关注的问题[7,8].

Parnas 在 20 世纪 70 年代提出了模块化和信息隐藏两大原则来避免软件中不同成分之间不必要的耦合关系,使得系统中的成分可以相对独立地发生变化[9].20 世纪 90 年代,Keepence 提出了一种面向对象的家族模型,用预定义的模式来建模产品家族内的变化性,但这种模型只是在详细设计阶段的解决方案[6].另一方面,Will Tracz 在 1990 年提出了一种基于特定领域的体系结构来捕获产品家族的共性的方法,该方法能够很好地组织核心资产,但对变化性的支持却讨论不多[10].Zhang 和 Jarzabek 改进了 Bassett 的框架技术(frame technology)[11],提出了 XVCL(XML-based variant configuration language)方法来处理变化点[12,13];Becker 提出的变化性规约语言 VSL(variability specification language)[7,14]也是基于 Bassett 的框架技术.XVCL 和 VSL 都是基于 XML 的配置语言,具有很高的灵活性和抽象性,但这种纯文本化且不直观的变化性处理机制给理解带来了一定的障碍.

1990 年,Kang 提出了面向特征的领域分析方法 FODA.FODA 用特征来对需求进行模块化组织,用特征和特征之间的关系来对整个产品领域进行建模[15].在此后的领域工程或者产品家族方法研究中,包括 FORM[16],PLP[17]等,都采用了面向特征的领域需求组织方式.然而,仅仅用特征模型来建模整个家族的变化性存在一定的局限性,主要表现在特征模型不能很好地支持对软件产品外在行为的变化性建模.

针对以上局限,Griss 等人引入了用况模型,把它与特征模型相结合来对产品家族进行建模,提出了 FeatuRSEB 方法[18].用况模型在有效地捕获系统功能性需求的同时,还能详细描述用户与系统的交互过程,而且用况模型简单易行,这些特点很好地弥补了仅用特征模型建模的不足.但 FeatuRSEB 方法中提到的用况模型还是传统的用况模型,对系统行为的变化性难以提供有效的表示机制[19,20];而且 FeatuRSEB 方法涉及到太多的视图,使建模过程过于繁琐.

根据以上分析,本文在现有特征模型和用况模型的基础上,以变化性的管理策略为指导,提出了一种组合的产品家族变化性建模方法.该方法首先从扩展的用况模型入手描述家族系统行为的变化性,然后以特征模型系统化地组织产品家族的功能性需求和质量属性并识别其变化性,同时,在这两种模型中采用相同的机制建模变化性,保证两种模型间的一致性.

本文第 1 节主要介绍有关变化性的一些基本概念.第 2 节首先介绍本文对变化性的管理策略,然后详细论述产品家族中控制变化性的两个模型:扩展的用况模型和特征模型,并给出了变化性建模的基本步骤.第 3 节介绍建模变化性的相应支持工具以及一个手机软件产品家族的建模实例.最后对全文进行总结并对下一步的工作进行展望.

1 有关变化性

本节首先介绍通常讨论变化性时所涉及的几个术语,包括软件产品线(software product line)、软件产品家族(software product family)和领域(domain),并对它们作一些辨析,然后在此基础上对变化性的定义、变化性的模式和生命周期等进行论述.

1.1 软件产品线、产品家族和领域

软件产品线是指针对特定市场区域、具有一组共性特征、从预先生产的核心资产开发而来的软件产品的集合[21].软件产品线源于工业界产品线的概念,关注于一个软件企业如何组织一组相似产品的生产.产品家族则是指一类共享体系结构属性、特征、代码、构件、中间件或者需求的软件产品[3,4],主要强调产品所在家族的共性,并不限于特定的软件企业.但有些研究对二者并不作严格区分[12].软件产品线和产品家族的共同之处在于它们都应用软件复用的相关原理与技术,都可以对应于软件复用过程的领域工程和应用工程两个阶段.

领域是指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[22].领域可以看成 4 种成分的结合体:商务范围、问题集合(问题空间)、应用系统的集合(解空间)以及共用术语组成的知识集[23].领域和软件产品线两者也是比较相近的.有研究者认为,它们的区别只在于领域是由一些概念性的术语组成,而产品线则是包含将要开发的具体的软件产品[23].也有研究者认为,二者的不同之处仅仅在于术语使用习惯的差异,学术研究人员通常使用“领域”,而软件项目经理等人员更易于接受“产品线”[24].

本文讨论的变化性建模方法并不对软件产品线、产品家族和领域这 3 个术语作严格区分,该方法同样可以应用于软件产品线和领域工程中.

1.2 变化性

在产品家族方法和领域工程中,变化性都是相对于共性而言的,共性是产品家族存在的基础,而家族内单个产品间的区别则属于变化性.从软件开发人员的角度来看,认定变化性意味着在软件生命周期中把决策延迟到某一个特定的时间阶段,即变化性的绑定时间(binding time).典型的绑定时间包括定制时刻、编译时刻、链接时刻、装载时刻和运行时刻等.

1.3 变化点(variation point)和变化点的属性

变化性通常关联在某个变化点上,每个变化点上都包含一个或一组相关的变量(variant)待确定,每个变量都是实现变化性的一种特定方式,它们指出了变化性的取值范围.在开发过程中,变量根据需要可以继续增加或者减少.根据变化点的变量是否固定可以区分变化点的两种类型:开放型(open)或封闭型(closed).如果该变化点的变量能被继续增添、删减或修改,则称该变化点是开放的,否则是封闭的.在一个产品的生命周期中,变量取值必须在其对应的绑定时间之前(包含绑定时间)被固定下来.变化点的绑定时间越晚,产品的灵活性和适应能力就越高.

1.4 变化性模式(pattern)和生命周期

变化性模式是指常见的变化性表现形式,最基本的模式有 3 种:可选(optional)、多选一(alternative)和多选多(multiple parallel).可选模式表示变化点上仅有一个变量,有待于最终决定是否绑定;多选一模式是指变化点上有一组相关的变量,而且最后只会选择其中之一绑定;多选多模式意味着在一组变量中,可以同时绑定一个以上的变量.另外还有可选的多选一(optional alternative)、可选的多选多(optional multiple parallel)等模式,它们可由基本模式组合而成.

变化性的生命周期是以体现变化性的变化点为观察对象,从该变化点处于隐含状态一直到最后的绑定状态为止所经历的几个阶段.完整的变化性的生命周期包括 6 个阶段,分别是变化性的识别、变化性的分析与设计、变化性的实现、变化性的定制、变化性的配置以及变化性的绑定.其中,前 3 个阶段的活动在家族核心资产的生产过程中完成,后 3 个阶段的活动则在单个应用系统的开发过程中完成.另外,变化性还可能会发生演化,这可能有两方面的原因,一是变化点的取值变量会发生变化,二是变化性所在的产品家族或应用环境发生变化

也能引起变化性演化.变化性的演化可能需要重新进行变化性的分析、设计和实现等活动.

2 软件产品家族的变化性建模

可以从时间和空间的两个维度来刻画变化性:时间上的变化性是指同一产品随着时间的推移在功能或质量属性上发生变化,也可以指同一产品在不同运行时刻根据不同的设置在功能或质量上发生变化;空间上的变化性,即家族中产品间的区别,是指同一时间针对不同客户或市场,产品具有的不同配置.时间上的变化性与空间上的变化性是相互独立而又密切相关的,时间上的变化性也可以通过同一产品不同版本的差异来体现.关注时间上的变化性主要是为了解决产品的演化问题,而关注空间上的变化性则主要是为了解决核心资产的复用问题.

本文中论述的变化性建模方法以变化性的管理策略为指导,以用况模型为起始,以特征模型为中心,从用况和特征两个角度来组织家族核心资产,识别变化性.两种模型对变化性采用相同的建模和约束机制,对产品间的差异(即空间上的变化性)明确标识.

2.1 变化性的管理策略

本文的变化性管理策略是指针对变化性生命周期的各个阶段提出相应的对策来实现对变化性的控制:

(a) 在变化性的识别阶段,需要把变化点识别出来.本文主要采用在用况建模过程中识别具有变化性的用况,以及在特征建模时识别具有变化性的特征的手段识别变化点;

(b) 在变化性的分析与设计阶段,首先需要识别变化性的模式,再确定变化点的各个变量值以及它们之间的关系,最后还要决定变化点的绑定时间等.我们的两种模型采用相同的变化性建模机制:在图形表达上,用图元变体表达变化性的可选模式,用维度(dimension)-值(value)[25]组织其他变化性模式,并用命题逻辑来表达它们之间的相互依赖、互斥关系.

(c) 在变化性的实现阶段,要利用各种变化性的实现技术,在详细设计甚至是编码阶段完成对变化性的控制.比如 Michialis[26]提出的聚合(aggregation)/代理(delegation)、继承(inheritance)、重载(overloading)、框架技术(frames)、反射技术(reflecion)、设计模式(design pattern),或者是 Mikael 提出的可选的体系结构构件(optional architecture component)、常量条件(condition on constant)、变量条件(condition on variable)等技术[27].

(d) 在变化性的定制阶段,依据变化性在产品家族中所受的限制关系来对变化性进行定制.定制变化性伴随着对产品家族模型的剪裁而发生,一般是对变化点的变量值进行删减.

(e) 在变化性配置阶段,根据单个应用系统的特性,并结合变化性的定制结果来对已实现的变化性进行配置或作部分调整和修改,以符合此应用系统的需要.

(f) 在变化性的绑定阶段,需要对变化点的变量值进行绑定.但对于空间上的变化性来说,变化性的绑定其实发生在变化性的定制阶段.

与以上完整的生命周期的 6 个阶段相对应,可以对变化性划分 7 种状态,其对应关系如图 1 所示,该图是对文献[28]中的变化性生命周期示意图的完善.变化性的状态分别是:① 隐含的:变化性未被识别的状态,如果一个变化性在某个开发阶段被引入,意味着在前面的阶段当中,这个变化性都是隐含的;② 已识别的:通过共性变化性分析活动,识别出该变化性;③ 已设计的:经过确定变化性的模式、绑定时间等,该变化性被设计和明确表达出来;④ 已实现的:采用某种实现技术,该变化点已经被实现了;⑤ 已定制的:定制后还存在的变化性是指同一个产品内部的变化性;⑥ 已配置的:此时已实现的变化性是针对于特定应用系统的;⑦ 已绑定的:体现该变化性的变化点的变量取值已经被绑定.

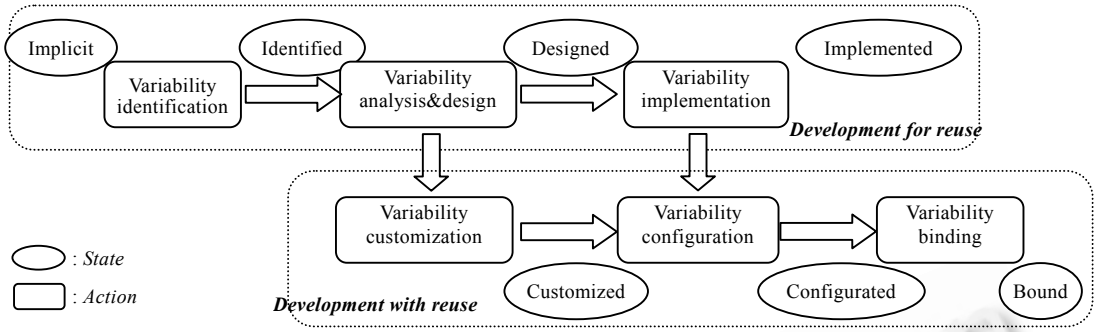

Fig.1 The lifecycle of variability

图 1 变化性生命周期

2.2 扩展的用况建模

用况模型最早由 Jacobson 在 1992 年提出[29],该模型以活动者和用况作为基本建模元素来刻画用户需求,着眼于描述软件用户与软件系统间的交互.用况可以用流程图、顺序图、Petri 网或程序设计语言来表示.但通常,示意图和简单文本是描述用况的实用形式.示意图能够直观地表现活动者与各个用况间的关系;而文本则对用况包含的每个场景进行了详细的描述.

一般来说,用户对希望怎样与系统交互以及希望得到怎样的系统反馈都有比较清楚的认识,因此通过与用户的交流建立系统的用况模型并不困难.正由于建立用况模型如此直观自然,所以一经提出就被各种研究方法所接受,并被广泛应用到软件开发过程中.

在变化性建模中引入用况模型正是考虑到用况模型的以下特点能够弥补特征模型的不足:

(a) 与特征模型相比,用况模型相对直观,可以作为整个建模过程的起点.用况模型能够促进用户与开发者之间的有效沟通,方便开发者捕获用户需求.

(b) 特征模型采用层次化的方式来捕获系统具有的功能和非功能性需求,而用况模型中的用况则详细说明了系统的功能如何被使用.用况模型相对于特征模型采用了一种不同的视角来描述软件.

(c) 用况模型能够捕获软件系统的外在行为.用况模型关注用户与系统之间的交互,这种动态信息与特征模型描述的静态结构结合起来能够更全面地描述软件系统.

(d) 用况模型更易于详细表述产品新功能的添加,支持对产品演化的描述.

下面将首先介绍变化性在用况模型中的具体体现,然后介绍本文对传统用况模型的扩展.

2.2.1 用况模型中变化性的具体体现

在产品家族中,不同的应用系统由于其针对的商业目标或用户群的不同,使得用户与系统之间的交互存在差异性.具体来说,变化性在产品家族用况模型中主要体现在以下 3 个方面:

- 活动者的不确定性,产品家族内的某个产品因为服务对象不一样,可能会有特别的活动者;
- 用况的不确定性,如果某个应用系统有特别的功能,那么使用这项功能的人机交互的典型方式即用况自然就是特别的;
- 用况间关系的不确定性,由于用况自身的不确定性,致使在产品家族的用况模型中用况间的关系也存在变化性.

2.2.2 对传统用况模型的扩展

传统的用况模型对变化性的支持机制有限[19,20].为了把用况模型引入产品家族建模中,特别是针对以上变化性,有必要做一些扩展.本文引入以下几种支持变化性的变体和关系:

- 活动者可选变体.表示这个活动者只出现在部分系统中,只是产品家族中部分应用系统的活动者.
- 用况可选变体.表示这个用况只发生在产品家族中部分成员的边界上.
- 维度-值关系.用来表示某个用况(这个用况称为基用况)与一组相关用况之间的关系.即在基用况上存在

一个变化点,这个变化点的取值存在于一组相关用况上.本文把这组用况称为基用况的值(value),而称基用况为维度(dimension).维度-值关系并不对维度上的每个值之间的关系有任何约束;对于约束,可通过扩展互斥关系等变化性约束机制来表示.维度-值关系和变化性约束机制的结合能够准确地表达多选一模式、多选多模式等多种变化性模式.

维度-值机制为变化性的隔离、抽象和封装提供了很好的支持.维度是其所有值的一个抽象.这种抽象隔离了变化性对其他部分的影响.每一个值则封装了不同的变化性,而且当新的变化性产生时,可以把它作为一个值添加到维度中,从而也实现了对变化性演化的支持.

- 互斥关系.表示两个用况间有某种冲突或者不能同时发生,增加这种关系主要是为了确保对家族用况模型剪裁时,不把有冲突的两个用况同时选进单个软件系统模型里.例如,对于手机来说,在国内现有网络条件下,发送彩信是基于 GPRS 网络服务,发送彩 e 是基于 CDMA 网络服务.而 GPRS 网络和 CDMA 网络分别由不同的通信运营商提供,所以在对手机软件产品家族建模时,发送彩信用况和发送彩 e 用况间的关系就可以用互斥关系来表达,在模型剪裁时只能二选其一.互斥关系是一种对变化性的约束.

通过增加以上新元素得到的用况扩展元模型如图 2 所示,这是对 UML 用况元模型[30]的扩展(图 2 中灰色图元为新增的建模元素).Thomas 在 2002 年也对 UML 用况元模型进行了扩展[19],但其扩展机制不能表示活动者的变化性;而且由于没有认识到用况间存在互斥关系,在表达多选一模式和多选多模式时相对困难.而且,由于其扩展机制和特征模型的变化性建模机制不一致,致使这两种模型不能很好地结合.

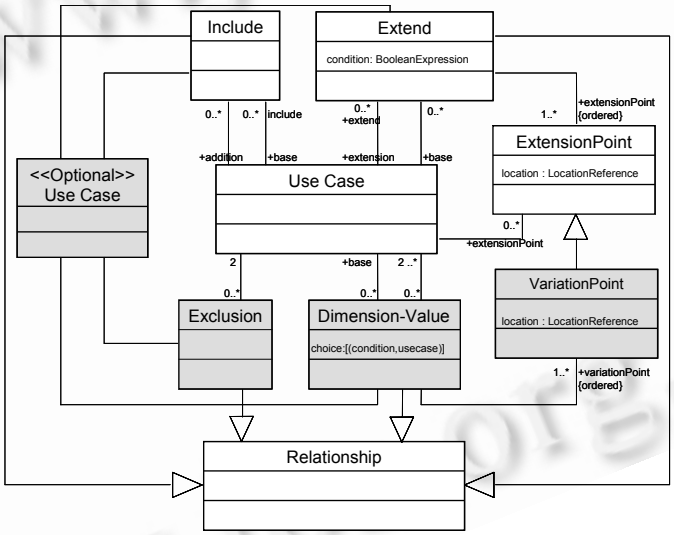

Fig.2 The extended use-case metamodel

图 2 用况扩展元模型

根据扩展的用况元模型,可得到用况的新模版.见表 1,这是基于 Cockburn 在文献[31]中建议的模版作出的改进版.其中 IsOptional,Dimension,Exclusion 属性都是引入用以支持变化性的.IsOptional 用来标识用况是否可选,Dimension 用来记录维度-值关系,而 Exclusion 用来记录互斥关系.为保证用况模型与特征模型之间的可追踪性,还特别引入 FeatureRelated 来记录与一个用况相联系的特征.

2.3 特征模型

1982 年,Davis 在文献[32]中提出以特征(feature)为基本单元描述软件需求.在 FODA 方法[15]中,这种面向特征的需求规约组织方式被引入到领域工程和产品家族的研究中.与传统软件需求规约的组织结构相比,特征模型作为一种新型的需求组织方式具有结构简单、支持复用、便于交流、易于图形化建模等优点.因而在后来的各种方法中,包括 FORM,FeatuRSEB,PLP 等,都把特征作为领域需求空间的一阶实体,以特征模型为中心组织领域模型.

Table 1 The extended template of use-cases
表 1 扩展后的用况模版

Name	名字
Description	简短解释
Use cases list	参与用况列表
Pre_Condition	前置条件
Post_Condition	后置条件
Procedure	过程序列
…	…
IsOptional	是否可选
Feature related	相关的特征
Relation	关系
Inclusion	包含关系
Extension	扩展关系
Generalization	泛化关系
Dimension	维度值关系
Exclusion	互斥关系
…	…

但遗憾的是,以上各种方法都欠缺对特征模型的组织框架的深入研究,对特征模型中各种结构元素的语义也没有作出明确的说明.这些情况导致了特征模型在表现形式上的冗余性和混乱性.为了解决这些方面的不足,我们在总结以往领域工程和产品家族研究成果的基础上,提出了一种适用于领域工程和产品家族的特征模型.我们从特征模型的基本组织结构、变化性的表现方式和限制机制等若干方面对特征模型的组织框架和剪裁机制进行了统一、抽象的描述,并在此基础上给出了特征模型的具体形式以及建模过程,形成了一种面向特征的领域建模方法(FODM)[33].下面本文着重介绍特征模型的组织结构和具体形式以及特征模型的变化性表示及约束机制.

2.3.1 组织结构和具体形式

一个特征描述了一个相对独立的需求实体.我们提出的特征模型在关注单个特征的同时,也注重发现特征之间存在的语义关系.特征模型的基本组织结构提供了一种把各个独立的特征组织在一起的方式.与大部分研究方法的特征模型相同,我们采用层次式的方式来组织特征,各层之中的特征通过整体-部分关系(whole-part association,简称 WPA)连接,为形象起见,本文也将每个整体-部分关系中的整体特征称为父特征,其部分特征称为子特征.

需求工程把软件需求分为业务需求、用户需求和功能需求 3 个不同的层次[34],这 3 个层次上的需求都可能是特征存在的地方.我们把业务需求、用户需求和功能需求中具有的特征分别称为服务特征、用况特征和功能特征.而一个功能在执行过程中表现出的行为特点,既可能是产品家族中所有系统的共性,也可能是只有单个系统才具有的独特之处,我们也把它作为特征的一种表现形式称为行为特点特征.此外,还有一些特征来源于系统的质量和性能要求,我们称为质量特征.

而在用况和特征之间,存在以下关系:① 一个用况可能与多个特征相关联.一个特征也可能与多个用况相关联.② 用况本身也可能成为特征的来源,但能否产生特征要依据领域专家等参与者的经验和判断.③ 特征的关注对象是系统本身,用以表明系统有哪些特点,而用况则详细说明系统的功能如何被使用,观察的角度在系统使用者上.④ 每个来自于用况的特征往往也可以看做是一个行为特点特征,或者功能层特征,或者服务层特征.

基于以上分析得到了如图 3 所示的特征模型的具体形式.该特征模型除了记录系统具有的服务、功能、行为特点、用况等特征外,还显示记录了系统具有的质量特征.服务、功能、行为特点 3 种特征通过 WPA 形成层次式结构;而任意特征之间都可能存在依赖或互斥等关系,表明它们之间的相互关联或影响.

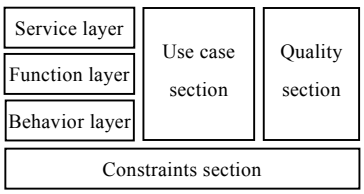

Fig.3 A concrete form of feature models
图 3 特征模型的具体形式

2.3.2 变化性表示机制

显然,上面提到的 5 种具体形式的特征都可能存在变化性.特征模型提供了两种表示变化性的机制:

① 特征的可选性.特征的可选性依附于 WPA 上,表现为部分特征相对于整体特征的可选性.

② 特征自身的变化性.特征自身的变化性是指一个特征由于封装了不同的细节而体现出的具有不同行为特点的特征.与扩展的用况模型相同,我们用维度-值的概念来描述特征及其具有的变化性.在特征模型中,我们把一个自身具有变化性的特征称为一个维度,把它封装了不同细节的变化性特征称为该维度上的一个值.

同样,在特征模型中,维度-值机制也没有对同一维度上不同值之间的关系做出强制的规定,而维度-值机制和变化性约束机制的各种组合,可比 FeatuRSEB 等方法中的变化点-变量机制表达更丰富的变化性语义.

2.3.3 变化性约束机制

变化性的约束机制用来建模变化性特征之间的约束关系,主要用在家族模型剪裁和变化性绑定的时刻.利用变化性特征之间的约束关系,可以检验经过剪裁或绑定后的特征模型是否满足约束条件.

在生成单个应用系统的特征模型时,需要对家族模型做剪裁,主要任务包括决定是否保留可选特征以及对维度上的值作一些删减.剪裁家族模型只是去除了产品间的差异性,剪裁后的单个应用系统特征模型同样还可能存在变化性,这些变化性将一直保留到绑定时刻,比如系统运行时刻.而在变化性的绑定阶段,变化性特征之间仍然存在约束关系.

本文采用的变化性约束机制是基于命题逻辑的.从变化性特征剪裁时是否保留和最终是否绑定的角度来看,在家族模型里一个变化性特征从被识别出开始,它在生命周期内存在 4 种状态:保留状态与未保留状态,绑定状态与未绑定状态.由此,可以用谓词作用于特征来表达该特征所处的状态:以 T 表示谓词:"...在剪裁后处于保留状态",则对于特征 f,命题"特征 f 在剪裁后处于保留状态"可表示为 $T(f)$,T 的取值为

$$T(f)\begin{cases}\text{Ture}, & f\text{处于保留状}\\ \text{false}, & f\text{处于未保留状}\end{cases}$$

同样,以谓词 B 表示"...在变化性绑定后处于绑定状态",对于特征 f,命题"特征 f 在绑定后处于绑定状态"可表示为 $B(f)$,B 的取值为

$$B(f)\begin{cases}\text{Ture}, & f\text{处于保留状}\\ \text{false}, & f\text{处于未保留状}\end{cases}$$

而对于任一特征来说,如果最后能被绑定则必然意味着剪裁时也被保留下来了,用关系式可表示为 $(\forall f)(B(f)\rightarrow T(f))$.

变化性常见的模式比如多选一模式和多选多模式通常在变化性绑定时用来表达对值之间的约束关系,可以用逻辑关系式明确定义:

$$Alternative(F1,F2,...,Fn)=_{def}(\exists i\forall j)(1\le i\le n)(1\le j\le n)(i\ne j)(B(Fi)\wedge\neg B(Fj)),$$

$$MultipleParallel(F1,F2,...,Fn)=_{def}(\exists i)(1\le i\le n)B(Fi).$$

而在特征模型中,依赖(require)和互斥(exclude)关系是特征间最常见的两种关系.两特征间的依赖关系表示一个特征的存在必须以另一个特征的存在为前提,比如在 WPA 关系中,其子特征都依赖于父特征;两特征间的互斥关系表示这两个特征由于某种冲突而不能同时共存.可以用命题逻辑来表述这两种约束关系,如果用 P 表示"在特征模型中存在特征 P",用 Q 表示"在特征模型中存在特征 Q",则特征 P 和特征 Q 之间的依赖或互斥约束可分别写作:

P *Require* Q 可写为 $P\rightarrow Q$;

P *Exclude* Q 可写为 $\neg(P\wedge Q)$.

而为了区分在裁剪时刻和绑定时刻特征间存在的约束关系,就需要用上面提到的两个谓词 T 和 B,如对特征 P 和特征 Q,它们在模型剪裁时要满足的依赖或互斥关系可以表述为

裁剪时刻 P *Require* Q 可写为 $T(P)\rightarrow T(Q)$;

裁剪时刻 P *Exclude* Q 可写为 $\neg(T(P)\wedge T(Q))$.

如果这两个特征是在绑定时刻有依赖或互斥的关系,则可以表述为

绑定时刻 P *Require* Q 可写为 $B(P)\rightarrow B(Q)$;

绑定时刻 P *Exclude* Q 可写为$\neg(B(P)\wedge B(Q))$.

总之,通过逻辑运算符和谓词 T 和 B,家族特征模型内特征之间的约束关系都可表现为一组逻辑关系式,这为模型剪裁和变化性绑定时的关系语义检查提供了有力的支持.

2.4 变化性建模过程

在前面所介绍的用况模型及特征模型的基础上,本节给出变化性建模的具体过程.图 4 描述了该过程包含的主要活动以及活动之间的相互关系.该过程首先从构建产品家族的单个典型产品的用况模型入手,以此为基础对家族的变化性进行分析,进行面向家族的用况建模和特征建模,同时维护用况模型和特征模型之间的可追踪性.在整个建模过程中,还要注意把分析活动中获取的领域术语记录到术语词典中.

构建典型用况模型.这一子过程在整个变化性建模过程中是可选的,但如果建模人员对整个产品家族并不是很熟悉,则有必要先进行此项子过程.建立典型产品的用况模型不用考虑其家族特性,按普通的用况模型建模方法即可.实施这一子过程有利于促进建模者理解整个产品家族,从而为下一阶段的建模打下坚实的基础.

构建家族用况模型.构建家族用况模型是以典型用况模型为基础,添加产品家族中其他系统里出现的活动者和用况,并根据每个活动者和每个用况在产品家族中的特性,分别用不同的方式标识出其共性和变化性,然后正确地表达它们之间的语义关系.整个子过程大体上可分为 4 个步骤来进行:

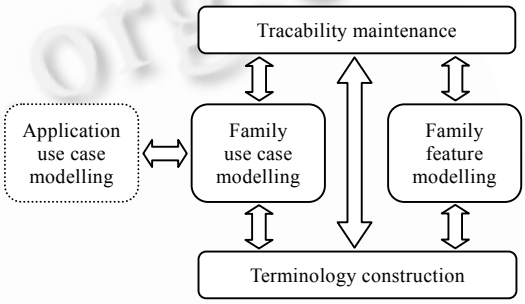

Fig.4 Variability modeling process

图 4 变化性建模过程

第 1 步.分析典型用况模型中的活动者和用况在整个领域中的特性,对于每个产品都会出现的活动者和用况,直接提取到领域用况模型中来;对于只是该产品才出现的活动者和用例,则用活动者变体和用况变体标识出来.

第 2 步.考虑整个领域中的其他产品系统,补充可选的活动者和用况,并用变体表示.特别是对于每一变体,要详细说明该变体存在的条件.

第 3 步.描述每个可选用况的主场景,找出主场景执行的前置条件,识别出部分条件不满足时可能导致的次场景.

第 4 步.用基本的泛化关系、包含关系、扩展关系或扩充的变化性的维度-值关系、互斥关系来刻画用况间的关联.

构建家族特征模型.依据前面介绍的特征的 5 种具体形式,构建家族特征模型要考虑产品家族内所有应用系统的服务、功能、行为特点等特征,以及与其相关的用况特征和质量需求特征,并注意分析每个特征的共性或变化性.家族特征建模过程主要包括六大活动,即交互过程分析活动、服务分析活动、功能分析活动、行为特点分析活动、质量需求分析活动和共性变化性分析活动.其中交互过程分析是特征建模的起点,它的主要任务是发现用户和系统交互过程中存在的特征.这就要求先进行用况建模,在用况建模结束后,特征建模要依据用况建模的结果,分析交互行为,从中抽象出用况特征,然后从用况模型和用况特征中归类总结出应用系统的各种服务,继续进行下面的特征建模活动.具体过程指南可参见文献[33].

维护可追踪性.本文用两个模型来对产品家族中的变化性进行建模,如何在这两个模型中保持可追踪性就变得尤为重要.维护两个模型间的可追踪性也是进一步完善这两个模型的一个过程,同时也能够对将来可能的演化提供必要的支持.这项过程的主要任务包括对用况模型中的每一个用况建立与之相关的特征列表、对特征模型中的每一个特征建立与之相关的用况列表以及根据追踪情况对两个模型作出适当的调整等.

构建术语词典.术语词典定义了产品家族内产品开发过程中所用到的术语,为软件开发的参与人员理解产品家族以及相互交流提供了一致的术语空间.术语词典是产品家族核心资产的一个重要成分.在整个变化性建

模过程中,需要注意把提炼出的术语添加到术语词典中.

3 支持工具和实例研究

本节首先介绍前文中论述的变化性建模方法相应的支持工具,然后以手机应用软件产品家族为实例对象,示例前述变化性建模方法的实施过程.

3.1 变化性建模工具

我们的建模工具包括用况建模和特征建模两个工具.两个工具均用 Java 语言开发,基于通用的图形用户界面,支持用户所见即所得的编辑操作;在模型的存储上采用基于 XML 的语义文本.

图 5 是特征建模工具的界面.其顶部是菜单栏和常用工具栏,中部左侧是导航栏,中部右侧是模型元素编辑区域,底部是系统信息的显示窗口.在编辑区域通过点击模型元素的右键菜单,弹出其属性编辑界面.工具还提供模型检查、术语维护、文档发布等功能,可以通过在菜单栏中点击相应的菜单项来触发这些功能.

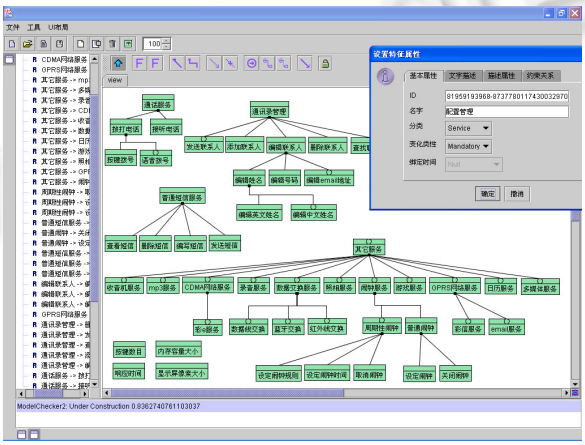

Fig.5 The feature model of mobile phone software family (part)

图 5 手机应用软件的家族特征模型(部分)

3.2 手机应用软件产品家族的变化性建模

手机应用软件是手机产品的专用软件系统.在手机用户越来越庞大的现代社会,手机产品种类也越来越多.但无论其外观差异如何巨大,作为同一个产品家族的产品,手机内部应用软件系统的基本功能是比较稳定的.对其变化性进行建模,便于比较和区分不同系统的功能特点,也有助于组合不同的变化性开发新的手机应用软件,推出新产品.

整个变化性建模过程从构建单个用况模型开始.我们首先选取一个典型的手机应用软件,针对它构建典型的用况模型;然后以这个用况模型为基础来构建家族用况模型.经过分析,手机用户作为最基本的活动者,而数据交换设备是与手机进行数据交换的,比如其他的手机,或者电脑等,因为不是所有的手机都有交换数据的功能,所以数据交换设备作为可选的活动者.同样,语音拨号、发送电子邮件、手机照相等用况不具有产品家族共性,都用可选用况图标来标识.对于每个活动者,用况都要详细描述其属性;而对于变体,还要特别标注其存在或发生的条件.最后,需描述各用况之间的关系,比如数据传输与数据线传输、红外线传输、蓝牙传输,用况间用维度-值关系来表达.同样,拨出电话与按键拨号、语音拨号间也用维度-值关系来表达.而且注意到,语音拨号与按键拨号在运行时刻是互斥的关系,这也是变化性常见的多选一模式.图 6 所示为手机应用软件的部分家族用况模型视图.

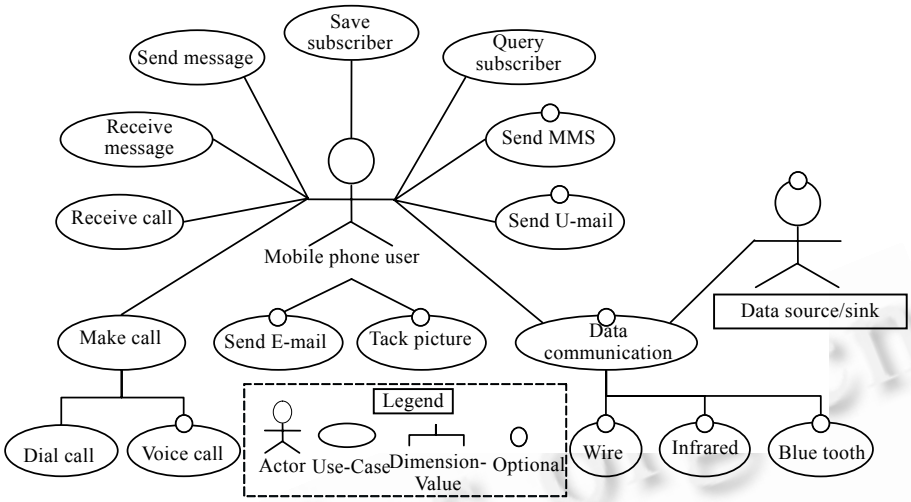

Fig.6 The use-case model of mobile phone software family (part)

图 6 手机应用软件的家族用况模型(部分)

接下来,按特征模型的建模指导过程来创建家族特征模型.特征建模从交互过程分析活动开始:经过分析,接入通话、拨出电话、发送短消息、存储联系人、数据传输等用况都可以抽象出相应的特征;再根据用况模型和这些特征并结合手机软件的特点,可以总结出手机软件有通话服务、通讯录管理服务、话机配置服务和短信服务四大基本服务,另外还有闹钟、日历、录音、游戏、数据交换等其他可选服务;而对于每一服务特征,可以将它细化为不同的功能特征,比如通话服务可以分为拨打电话和接听电话两大功能;而对于某些功能性特征,又可按其行为特点进一步细分.比如拨打电话就可按拨打方式细化为按键拨号和语音拨号;另外还有一类服务质量特征,比如响应时间、按键数目、内存容量大小以及话机显示屏像素大小等特征;特征建模还要进行共性和变化性分析活动,其主要工作是识别特征的共性或变化性、分析特征之间可能存在的约束关系,在标识出变化性特征后,要特别考察变化性特征与其他特征间的约束关系,比如 GPRS 网络服务与 CDMA 网络服务是剪裁时刻互斥的,而剪裁时刻每个可选子特征都依赖于其父特征,如周期性闹钟特征就依赖于闹钟服务特征,此外还有绑定时刻的约束关系,比如拨打电话特征要在语音拨号和按键拨号特征之间进行单选.手机应用软件家族特征模型(部分)如图 5 所示,其约束关系如下:

T(GPRS 网络服务) $\rightarrow \neg T$(CDMA 网络服务),

T(语音拨号) $\rightarrow T$(录音服务),T(彩 e 服务) $\rightarrow T$(CDMA 网络服务),T(彩信服务) $\rightarrow T$(GPRS 网络服务),

T(周期性闹钟) $\rightarrow T$(闹钟服务),T(数据交换服务) $\rightarrow$ (T(数据线交换)$\vee T$(蓝牙交换)$\vee T$(红外线交换)),

T(数据线交换) $\rightarrow T$(数据交换服务),T(蓝牙交换) $\rightarrow T$(数据交换服务),

T(红外线交换) $\rightarrow T$(数据交换服务),T(email 服务) $\rightarrow T$(GPRS 网络服务),

B(拨打电话) $\rightarrow$ *Alternative*(按键拨号,语音拨号),B(编辑姓名) $\rightarrow$ *MultipleParallel*(编辑英文姓名,编辑中文姓名),B(闹钟服务) $\rightarrow$ *Alternative*(周期性闹钟,普通闹钟),B(数据交换服务) $\rightarrow$ *Alternative*(数据线交换,蓝牙交换,红外线交换),…

整个变化型建模的最后要建立两个模型间的可追踪性,并维护两模型的一致性,这一过程同时也需要对两种模型进行检查,修正错误和完善建模.而在上述整个变化性建模过程中,需要注意把遇到的术语进行提炼、整理,然后记录到术语词典中.

4 结束语

本文在分析已有各种变化性建模方法和控制技术的基础上,提出了一种以变化性的管理策略为指导、以扩展的用况模型和特征模型为表现形式的变化性建模方法.本文还论述了变化性建模的基本过程,并结合相应的

支持工具,以手机应用软件为实例展示了该过程的主要步骤和结果.

下一步的研究工作将进一步关注产品家族中的变化性在实现阶段和演化过程中所遇到的问题,以期得到针对软件产品家族变化性控制的较完整的解决方案.对现有支持工具的完善和改进也是将来的任务之一.

References:

[1] Mili H, Mili F, Mili A. Reusing software: Issues and research directions. IEEE Trans. on Software Engineering, 1995,21(6): 528−562.

[2] Karlsson EA. Software Reuse: A Holistic Approach. Chichester: John Wiley and Sons Ltd, 1995.

[3] Bosch J. Design and Use of Software Architectures. Addison-Wesley, 2000.

[4] Jazayeri M, Ran A, Van Der Linden F. Software Architecture for Product Families: Principles and Practice. New York: Addison-Wesley, 2000.

[5] James W. Investment analysis of software assets for product lines. Technique Report, CMU/SEI-96-TR-010, ADA 315653, Pittsburgh: Software Engineering Institute, Carnegie Mellon University, 1996.

[6] Keepence B, Mannion M. Using patterns to model variability in product families. IEEE Software, 1999,16(4):102−108.

[7] Becker M. Towards a general model of variability in product families. In: Bosch J, ed. Proc. of the 1st Workshop on Software Variability Management. Groningen, 2003.

[8] Geyer L, Becke M. On the influence of variabilities on the application-engineering process of a product family. In: Chastek G, ed. Proc. of the 2nd Software Product Line Conf. Lecture Notes in Computer Science, Vol. 2379, Heidelberg: Springer-Verlag, 2002. 1−14.

[9] Parnas D. On the criteria to be used in decomposing systems into modules. Communications of the ACM, 1972,15(12):1053−1058.

[10] Tracz W. A domain-specific software architecture engineering process outline. ACM Software Eng. Notes, 1990,18(2):40−49.

[11] Bassett PG. Framing Software Reuse-Lessons from the Real World. Indianapolis: Prentice Hall PTR, 1996. 3−69.

[12] Zhang HY, Jarzabek S. An XVCL approach to handling variants: A KWIC product line example. In: Rivepiboon W, ed. Proc. of the 10th Asia-Pacific Software Engineering Conf. Chiang Mai: IEEE Computer Society, 2003. 116−125.

[13] Jarzabek S, Zhang H. XML-Based method and tool for handling variant requirements in domain models. In: Titsworth FM, ed. Proc. of the 5th IEEE Int'l Symp. on Requirements Engineering, RE 01. Toronto: IEEE Computer Society, 2001. 166−173.

[14] Becker M. XML-Enhanced product family engineering. In: Tsai JP, ed. Proc. of the 6th Biennial World Conf. on Integrated Design and Process Technology (IDPT 2002). Pasadena: Society for Design and Process Science, 2002.

[15] Kang KC, Cohen SG, Hess JA, Novak WE, Peterson AS. Feature-Oriented domain analysis feasibility study. Technique Report, SEI-90-TR-21, Pittsburgh: Software Engineering Institute, Carnegie Mellon University, 1990.

[16] Kang KC, Kim S, Lee J, Kim K, Shin E, Huh M. FORM: A feature-oriented reuse method with domain-specific architecture. Annals of Software Engineering, 1998,5:143−168.

[17] Chastek G, Donohoe P, Kang KC, Thiel S. Product line analysis: A practical introduction. Technique Report, SEI-2001-TR-001, Software Engineering Institute, Carnegie Mellon University, 2001.

[18] Griss ML, Favaro J, d'Alessandro M. Integrating feature modeling with the RSEB. In: Proc. of the 5th Int'l Conf. on Software Reuse. IEEE Computer Society, 1998. 76−85.

[19] van der MaBen T, Lichter H. Modeling variability by UML use case diagrams. In: Geppert B, Schmid K, eds. Proc. of the Int'l Workshop on Requirements Engineering for Product Lines (REPL 02). Essen: Avaya Inc., 2002. 19−25.

[20] John I, Muthig D. Tailoring use cases for product line modeling. In: Geppert B, Schmid K, eds. Proc. of the Int'l Workshop on Requirements Engineering for Product Lines (REPL 02). Essen: Avaya Inc., 2002. 26−32.

[21] Clements PC, Northrop L. Software Product Lines—Practices and Patterns. New York: Addison-Wesley, 2001. 1−50.

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

[23] Schmid K. Scoping software product lines. In: Donohoe P, ed. Software Product Lines, Experience and Research Directions. Kluwer Academic Publisher, 2000. 513−532.

[24] Czarnecki K, Eisenecker UW. Generative Programming: Methods, Tools, and Applications. New York: Addison-Wesley, 2000. 82−130.

[25] Geyer L. Feature modeling using design spaces. In: Knauber P, Pohl K, eds. Proc. of the 1st German Workshop on Software Product Lines. Kaiserslautern: Fraunhofer IESE, 2000. 35−39.

[26] Anastasopoulos M, Gacek C. Implementing product line variabilities. ACM SIGSOFT Software Engineering Notes Canada, 2001,26(3):109−117.

[27] Svahnberg M, van Gurp J, Bosch J. A taxonomy of variability realization techniques. Technical Report, Blekinge Institute of Technology, 2002.

[28] Chen ZL. Research on domain application variability control mechanism and technology [Ph.D. Thesis]. Beijing: Peking University, 2002 (in Chinese with English abstract).

[29] Jacobson I, Christeron M, Overgaard G. Object Oriented Software Engineering: A Use Case Driven Approach. Massachusetts: Addison-Wesley, 1992. 123−159.

[30] OMG Unified Modeling Language. Version1.4, 2001. http://www.uml.org

[31] Cockburn A. Writing Effective Use Cases. Addison-Wesley, 2000. 21−138.

[32] Davis AM. The design of a family of application-oriented requirements languages. Computer, 1982,15(5):21−28.

[33] Mei H, Zhang W, Gu F. A feature oriented approach to modeling and reusing requirements of software product lines. In: Titsworth F, ed. Proc. of the COMSAC 2003. Dallas: IEEE Computer Society, 2003. 250−255.

[34] Wiegers KE. Software Requirements. 2nd ed., New York: Microsoft Press, 1999. 3−22.

附中文参考文献:

[4] 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.

[28] 陈兆良.领域应用变化性控制机理与技术研究[博士学位论文].北京:北京大学,2002.