一种基于角色的特征模型构件化方法

张 俊,刘淑芬,姚志林
(吉林大学计算机科学与技术学院,吉林长春 130012)

摘 要: 为了解决领域特征模型混杂交织及其与需求模型过度耦合的问题,本文设计了一种特征模型构件化方法.该方法引入角色的概念,并以角色为中介设计了特征-角色-构件映射算法,将在领域分析过程中提取和抽象的特征映射到不同的构件模型上.通过角色的中介作用,方法实现了特征模型和需求模型的解耦,各个特征模型的可变点可以自由方便地选择和组合,从而提高了软件的构件化水平.

关键词: 特征模型;角色;模型构件化;特征-角色-构件算法
中图分类号: TP311 **文献标识码:** A **文章编号:** 0372-2112 (2011) 02-0304-05

A Feature Model Componentization Method Based on Role

ZHANG Jun, LIU Shu-fen, YAO Zhi-lin
(*College of Computer Science and Technology, Jilin University, Changchun, Jilin* 130012, *China*)

Abstract: To solve feature model's chaos and entanglement problems in certain domain and decouple the feature model and requirement model, we design a feature model componentization method. The method introduces the concept of Role, and implements an algorithm based on it called Feature-Role-Component Algorithm which maps domain features generated by requirement elicitation and analysis to different model components. The Role plays the role of intermediary, and decouples the feature and component, which enables convenient selection and composition between feature variants and enhances the componentization level of the system.

Key words: feature model; role; model componentization; feature-role-component algorithm

1 研究背景

目前软件行业面临的一个重要问题是开发过程的非标准化——过度依赖于个人的"艺术创造"而非形式化标准,这导致了产品质量的不可预测性和不可控制性,一切结果都取决于参与人员的经验和决策.相关人员经验的缺乏及软件规模的庞大不可避免地导致软件产品质量低下、缺乏适应性和可扩展性,难于维护和演化.为了实现软件过程的可控化和自动化、提高软件质量,面向对象技术、设计模式、构件技术、软件体系结构理论、软件推导及演化等多种理论和技术应运而生.其中,软件产品线(Software Product Line, SPL)理论作为高效而低成本的产品族开发范式得到了广泛关注.

软件产品线管理软件密集系统中一组共享通用的、受管的特征集合,这些特征按照指定的方式从公共核心资产中开发出来,满足特定社会细分或任务的特殊需求[1].其核心思想是在领域建模阶段通过分析领域需求创建出一套可以满足全领域软件产品需求功能和约束的特征模型,之后再根据单个软件的具体需求,通过对特征模型中的特征通用点及可变点的选择和组合——特征定制来实现具体产品的自动化生成.特征模型方法作为高度定制化与可重用软件的开发方法为软件的标准化生产提供了一种可能的及有效的解决途径.

目前,领域特征建模技术得到了广泛的关注并取得了很大的进展,但仍然存在着许多问题.当前特征模型的大部分是基于决策模型的,其基础就是基于模板的特征模型——一个涵盖全部特征的系统模型,并根据特征选择对系统模型进行裁剪得到应用模型[2].这种解决方案存在着如下问题:首先,领域建模与需求工程的内在耦合使得特征模型可变点混杂交织,建模人员需要事先分析和预测到领域内产品的所有可变点,构造这样庞大模板模型的开销甚至比直接创建一套可执行系统还要大,而且对于某些互斥的特征,将其模型整合在一起是不可能或无意义的;其次,在应用工程阶段,特征的选取产生的模型的剪裁会产生悬挂边[2]等问题,需要应用工程师手动更改;另外,这种方法基本无法实现特征模

收稿日期:2009-09-03;修回日期:2010-07-02
基金项目:国家863高技术研究发展计划(No.2009AA010314);国家自然科学基金(No.60973041)

型对于非既有特征的演化能力的支持.产生这些问题的根源在于决策基础模型的高耦合,它并没解决传统工程中功能混杂的问题而仅仅将其转移到了更高的模型层次.

要解决这些问题,首先要解决决策(特征)模型混杂的问题,实现各个特征的独立性和可插拔性.为此,本文设计了一套特征模型构件化方法,该方法引入角色概念,通过对领域知识的分析提取,将特征模型分解为可复用的、功能简单的角色集合,并以角色为中介设计了一套特征-角色-模型构件映射算法,该算法完成特征模型的构件化转化,实现了特征模型与系统需求模型的解耦,解决了领域特征模型功能混杂交织的问题,大大提高了单个特征间自由选择与组合的能力,同时在一定程度上解决了特征模型演化问题,进一步提高了软件的构件化水平.

本文的组织结构如下:第一部分对文章的研究背景做一个简明扼要的介绍;第二部分给出了国内外在特征模型相关领域的研究成果并对其优缺点加以分析;第三部分给出了特征模型表达方式,引入角色的概念.第四部分给出了特征到角色的操作化过程,并以角色为中介设计实现了特征-角色-模型构件算法,阐述特征与模型构件的解耦和映射方法;第五部分对全文加以总结并给出今后的研究方向.

2　相关研究

软件产品线作为实现软件工业化生产的可能途径得到了业界的广泛关注和研究.Kang 等人首先在文献[3]中提出了特征模型的概念以及 FODA 方法,并在文献[4]中扩展 FODA 提出 FORM 方法,将软件开发划分为领域工程及应用工程两个阶段.Czarnecki 等在文献[2]中提出将产品线表示为特征模型和模型模板.模型模板是涵盖所有可用模板实例模型元素的模型集合,产品配置通过存在条件和元表达式对模型模板进行剪裁得到符合特定产品要求的模型.这种方法很好的表达了特征模型的公共特征点和可变特征点,并为特征及其实现提供了良好的可追踪性.但是由于模型模板的全领域特性使得其规模庞大,并且要求设计和维护人员对领域知识有全面而细致的了解,能够一次性完成领域所有公共及可变特征的建模工作,因此设计和维护工作量极大且极易出错.对应大型应用来说并不实用.Nicolas Guelfi 等人在文献[5]中提出基于模型转换的需求分析方式,通过使用 FIDJI 分析模型以及 UML、OCL、文本、用例图等工具实现需求到模型的映射.但该方法专注领域在需求的提取而非组合,因此基本上没有对模型组合的支持,无法实现功能的自由组合.

国内方面,文献[6]提出了一种基于领域模型和构件组合的开发框架 FDMCC,该框架通过领域模型描述系统公共特性和可变性,并在可变点上集成构件实现产品族.然而该方法并没有给出领域模型的抽取方式,使得从需求获取到模型建立的过程产生缺失.文献[7]提出了一种基于特征的构件模型,实现了构件与特征的映射,同时文献[8]也提出了一种基于领域特征本体进行构件的语义描述和组装工作.但他们方法的出发点都是为了实现构件的设计而非软件产品线特征模型,其关注点是构件实现了哪些特征而非对于选定的特征需要哪些构件,从而无法实现特征的自由选择和组合.

3　特征建模

3.1　特征模型

特征是单个需求的内聚集合,是对用户可见的软件系统特性.特征模型可以理解为由领域专家通过对领域的分析和抽象建立的特征集合以及这些集合元素间的关联和约束.其中特征关联描述了特征间的关系,可以细分为特化、分解等形式;约束则体现的特征间的静态绑定关系,为应用工程阶段需求定制的验证提供了支持.基本的约束关系包括必要 requires(即选择 A 特征则必须同时选择 B 特征)和互斥 excludes(即选择 A 特征就不能同时选择 B 特征).

如图 1 所示,整个特征模型被分解为飞行计划定制、飞行计划管理、雷达管制、空域管理及人机界面等特征.而飞行计划管理又被分解为航迹生成、冲突检测与冲突处理等子特征.因此可以说特征模型的创建过程就是不同抽象层次的特征不断分解的过程.

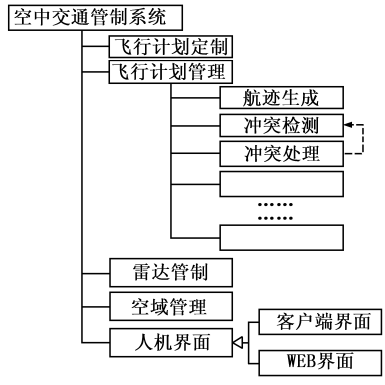


图1　空中交通管制系统的特征模型

在人机界面特征上可以看到特化关联.通过不同的技术实现不同的人机交互方式.然而在我们的方法中,并不十分强调这种关系,我们可以后移该过程,通过模型驱动技术的 PIM 到 PSM 的转换过程加以实现[9~11].

在约束方面,对冲突处理特征的选择必须要求冲突检测特征被选择,因此特征模型中存在约束 requires($f_{\text{冲突处理}}$,$f_{\text{冲突检测}}$).在本文设计的方法中,由于可以实

现特征间的自由选择和组合,因此特征间约束对方法的设计几乎没有影响.

3.2 角色

为了实现特征模型向构件模型的转换,我们首先引入角色的概念:

定义 1 角色可以定义为一个三元组 R = (ID, Des, OPS),其中 ID 是角色的标识;Des 是角色的一个描述,可以是形式化也可以是非形式化的;OPS 是角色定义的操作集合.根据定义 1,角色可以用来表示特征的协议或职责,它描述了特征要完成的某一特定任务.角色是特征与构件接口连接的媒介,通过角色,特征可以映射到构件模型上.引入角色后,特征模型的元模型可以简化地如图 2 所示.

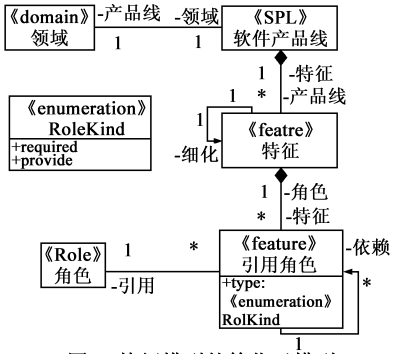


图2 特征模型的简化元模型

角色不针对某个特定特征,它存储在角色库中,供全产品线特征查询引用.为了兼顾角色的复用以及在特征中的具体作用,我 们另外引入了引用角色的概念.引用角色特定于某个特征,是对角色库中角色的引用,

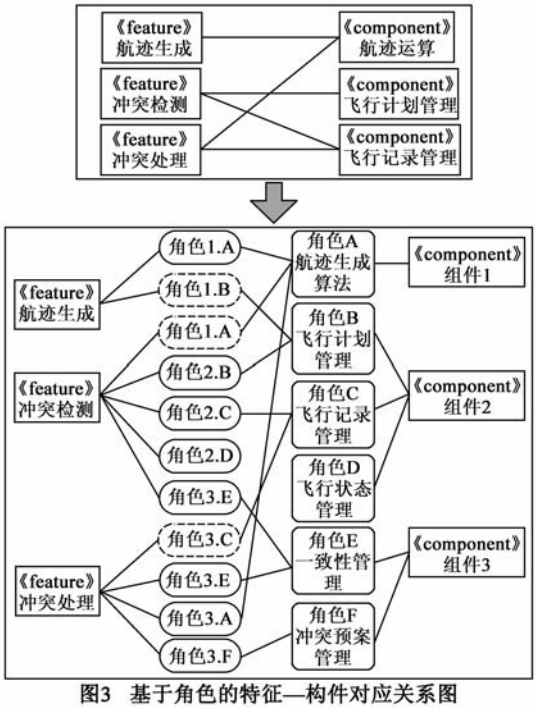


图3 基于角色的特征—构件对应关系图

每个引用角色有一个 type 属性来表明其在特征中的作用,type 的可能取值包括:required——该角色需要其他角色为其特供服务;provided——角色为其他角色提供服务.

如图 3 所示,通过角色和引用角色的中介,特征与构件间多对多的关系可以表示为两个一对多的关系,即:一个特征可以表示为多个角色间的相互作用;而构件则可以理解为多个角色的实现——即角色构成构件对外的接口.通过这种关系分解,我们可以将特征间复杂的关系解耦为一个个单独的构件,并通过角色的中介组合特征,从而形成系统模型.

4 特征-角色-构件映射算法

4.1 特征操作化

为了实现映射,在角色分析阶段,领域专家首先要进行特征操作化分解——给出特征包含的引用角色,引用角色的类型及其依赖(调用)关系,如图 4 所示.引用角色间箭头表示了其依赖关系.

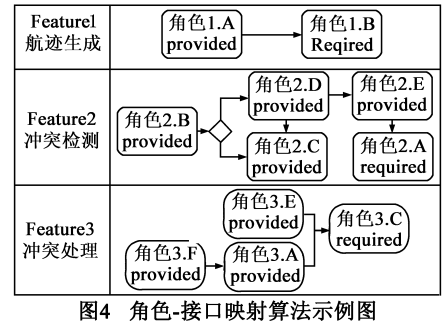


图4 角色-接口映射算法示例图

为了叙述和表示的方便,在以下算法陈述中以角色代码(ABCDEF)代表实际角色名.

4.2 特征-角色-构件映射算法

下面给出角色到构件接口的映射算法:

(1)计算所有角色的角色特征集

定义 2 角色特征集 $RFS(r)$ 表示对于角色 r,其引用角色所在特征的集合,即:

$$RFS(r) = \bigcup \{f_i \mid \exists\, i \cdot f_i = r.ref_i.ftr\} \tag{1}$$

其中 $r.ref_i$ 表示角色 r 的第 i 个引用角色,$r.ref_i.ftr$ 表示包含引用角色的特征.

定义 3 主动角色特征集 $PRFS(r)$ 表示对于角色 r,其类型为 *provided* 的引用角色所在特征的集合.

$$PRFS(r) = \bigcup \{f_i \mid \exists\, i \cdot f_i = r.ref_i.ftr \wedge r.ref_i.type = provided\} \tag{2}$$

(2)根据角色特征集生成构件

定义 4 构件是系统组成单元,由构件接口和实现组成,它被定义为一个七元组 Comp = {CID, PIntf, RIntf, PRole, RRole, Impl, Init},其中 CID 是构件标识,PIntf 表示构件的 provided 接口集合,RIntf 表示接口的 required 接

口集合,PRole 表示构件 provided 角色的集合,RRole 表示构件的 required 角色集合,Impl 是构件的实现,Init 表示构件的初始状态.

在该步骤中,算法的主要任务是根据主动角色特征集确定产品线构件——即为构件生成 Role 集合.过程 createComponetSet 根据 PRFS 集合值归类所有角色,并为每类角色创建一个构件,其算法描述如下:

(*a*)初始化角色集合 RS = R,CRS = ∅ .其中,RS 为角色库全部角色的集合,CRS 用于存放 PRFS 值相同的角色.

(*b*)如果 RS 集为∅ ,过程结束;否则,转到步骤 *c*.

(*c*)取出 RS 集的第一个角色 *r*,将其加入到 CRS 集合中,遍历 RS 集的其它角色 r_i,如果 PRFS(r_i) = PRFS(*r*) 则将 r_i 添加到集合 CRS 中,并在集合 RS 中删除角色 r_i.

(*d*)创建一个新的构件模型 *c*,并将 CRS 赋值给 *c* 的 PRole 集合.

(*e*)在 RS 中删除角色 *r*,跳转到步骤 *b*.

根据算法,我们可以得到:

$$CS = \{ Comp_{|PRole=\{A\}|}, Comp_{|PRole=\{B,C,D\}|}, Comp_{|PRole=\{E,F\}|} \}$$

(3)确定构件的 required 及 provided 接口

在步骤(2)中,由于使用了主动角色特征集,所以生成构件的角色集只包含 provided 角色集合而缺少 required 角色集合.因此,在步骤(3)中需要进行角色修正以及构件接口的生成工作.

根据分析,我们可以得到对于构件 *c*,其 provided 接口集可以通过公式(3)获得:

$$PIntf(c) = \{ op \mid \exists\, i \cdot op \in PRole_i(c).OPS \wedge |RFS(PRole_i(c)| > 1 \} \quad (3)$$

其中 *r*. *OPS* 表示某个角色 *r* 的操作集合,而 | *RFS*(*r*)|则表示角色 *r* 角色特征集的维度.

构件 RRole 集合则表示为:

$$RRole(c) = \{ r \mid r = PRole_i(c).ref_j.dep_k \wedge r.type = required \} \quad (4)$$

其中,Ref_j. *dep* 表示引用角色的依赖角色.

构件的 *Required* 接口集则可以表示为:

$$RIntf(c) = \{ op \mid \exists\, i \cdot op \in RRole_i(c).OPS \} \quad (5)$$

根据算法以上公式得到图 5 所示构件:

(4)构建特征与构件的映射

在步骤(4)中,根据特征引用角色类型以及构件接口构建特征(复合)构件.算法表示如下:

首先,根据公式(6)获得特征对应(复合)构件的简单构件集 FCS:

$$FCS(f) = \{ c \mid \exists\, i \cdot f.role_i \in PRole(c) \} \quad (6)$$

然后,根据公式(6)(7)(8)(9)分别获得(复合)构件的 Provided 接口集,Required 接口集,Provided 角色集以及 Required 角色集:

$$PIntf(c_f) = \bigcup PIntf(FCS_i(f)) \quad (7)$$

$$RIntf(c_f) = \bigcup RIntf(FCS_i(f)) - \bigcup PIntf(FCS_i(f)) \quad (8)$$

$$PRole(c_f) = \bigcup PRole(FCS_i(f)) \quad (9)$$

$$RRole(c_f) = \bigcup RRole(FCS_i(f)) \quad (10)$$

最后,根据以上公式求得的集合创建(复合)构件.算法得到特征的构件模型如图 6 所示:

经过以上算法,特征被映射到构件或复合构件上,并可以通过接口与其它特征构件方便的进行组合,以满足产品族不同产品的需求.

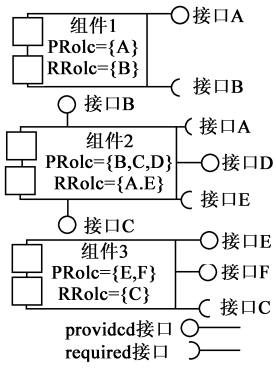


图5　系统模型构件图

在完成特征—角色—构件映射后,系统设计人员可以根据角色描述的功能细化构件模型,从而完成系统建模工作.由于模型构件是基于接口的构件,因此在特征的组合方面没有任何的限制与特殊要求,只要构件接口匹配,特征就可以自由的组合,从而实现了需求工程和领域工程的解耦,提高了系统的可扩展性及演化特性.

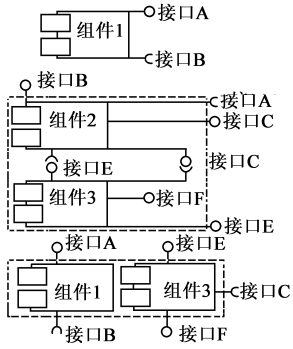


图6　特征构件映射图

5　总结与展望

本文对软件产品线技术和模型驱动技术进行了深入的探讨,针对目前特征不能实现灵活选择和组合的问题,提出了一套基于角色特征模型的软件设计框架.通过角色的中介作用,特征描述与构件实现得以解耦并建立映射,通过这种映射,可以方便地实现特征组合以及系统模型的构件化.

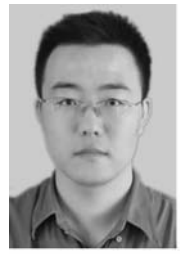
角色概念的提出大大提高了特征与模型的构件化水平,但也可能造成构件过分细化,可能无逻辑意义的问题.一种可能的解决途径是对角色引入语义,通过对角色的语义分析,组合过分细化的构件形成更大的有逻辑意义的复合构件.另外,如何实现角色的演化从而满足产品线不断变化的需求也是今后要研究的方向.

参考文献:

[1] P Clements, L Northrop. Software Product Line: Practices and

Patterns [M]. Boston, MA: Addison-Wesley, 2001.
[2] K Czarnecki, M AntKiewicz. Mapping features to models: a template approach based on superimposed variants [A]. Proceedings of the 4th International Conference on Generative Programming and Component Engineering [C]. Berlin: Springer-Velag, 2005. 422 – 437.
[3] K C Kang, S G Cohen, J A Hess, W E Novak, A S Peterson. Feature-Oriented Domain Analysis (FODA) Feasibility Study [R]. Technical Report, CMU/SEI-90-TR-21, Pittsburgh, Pennsylvania: Software Engineering Institute, 1990.
[4] K C Kang, S Kim, J Lee, K Kim, E Shin, M Hug. FORM: a feature-oriented reuse method with domain-specific reference architectures [J]. Annals of Software Engineering, 1998, 5(1): 143 – 168.
[5] N Guelfi, G Perrouin. A flexible requirements analysis approach for software product lines [A]. In: Proceedings of the 13th International Working Conference on Requirements Engineering: Foundation for Software Quality, LNCS-4542 [C]. Berlin: Springer-Velag, 2007. 78 – 92.
[6] 王晓燕,刘淑芬,张俊.一种基于领域模型和构件组合的软件开发框架[J].电子学报,2009,37(3):540 – 545.
Wang Xiao-yan, Liu Shu-fen, Zhang Jun. A framework based on domain model and component composition [J]. Acta Electronica Sinica, 2009, 37(10): 540 – 545. (in Chinese)
[7] 王忠杰,徐晓飞,战德臣.基于特征的构件模型及其规范化设计过程[J].软件学报,2006,17(1):39 – 47.
Wang Zhong-jie, Xu Xiao-fei, Zhan De-chen. Feature-based component model and normalized design process [J]. Journal of Sofware, 2006, 17(1): 39 – 74. (in Chinese)
[8] 彭鑫,赵文耘,钱永乐.基于领域特征本体的构件语义描述和组装[J].电子学报,2006,34 (12A):2473 – 2477.
Peng Xin, Zhao Wen-geng, Qian Yong-le. Semantic representation and composition of business components based on domain feature ontology [J]. Acta Electronica Sinica, 2006, 34(12A): 2473 – 2477. (in Chinese)
[9] D Muthing, C Atkinson. Model-driven product line architectures [A]. Proceedings of the 2nd Software Product Line Conference (SPLC2) [C]. Berlin: Springer-Velag, 2002. 110 – 129.
[10] øHaugen, B Møller-Pedersen, J Oldevik, A Solberg. An MDA-based framework for model-driven product derivation [A]. In Proceedings of the 8th IASTED International conference on Software Engineering and Applications [C]. Cambridge: ACTA Press, 2004. 709 – 714.
[11] O Avila-García, A E García, E V S Rebull. Using software product lines to manage model families in model-driven engineering [A]. Proceedings of the 22nd Annual ACM Symposium on Applied Computing-Model Transformation Track [C]. New York: ACM Press, 2007. 1006 – 1011.

作者简介:



张 俊 男,1981 年出生于吉林省吉林市,博士研究生.研究方向:模型驱动架构和系统架构.
E-mail: mail_of_zhangjun@163.com

刘淑芬 女,1950 年出生,教授、博士生导师,研究方向:计算机支持协同工作、软件体系结构、基于模型驱动的软件编程方法、网络管理技术.
E-mail: liusf@mail.jlu.edu.cn

姚志林(通信作者) 男,1973 年生于吉林长春,博士、讲师,研究方向:工作流技术,模型驱动架构及 SOA.
E-mail: yaozl@jlu.edu.cn