ISSN 1000-9825, CODEN RUXUEW
Journal of Software, Vol.17, No.6, June 2006, pp.1307−1317
DOI: 10.1360/jos171307

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

基于特征模型和构件语义的概念体系结构设计*

彭 鑫+, 赵文耘, 刘奕明

(复旦大学 信息科学与工程学院 软件工程实验室,上海 200433)

Feature Model and Component Semantics Based Conceptual Architecture Design

PENG Xin+, ZHAO Wen-Yun, LIU Yi-Ming

(Software Engineering Laboratory, School of Information Science and Engineering, Fudan University, Shanghai 200433, China)

+ Corresponding author: Phn: +86-21-55885080 ext 33, Fax: +86-21-55885087, E-mail: pengxin@fudan.edu.cn

Peng X, Zhao WY, Liu YM. Feature model and component semantics based conceptual architecture design. *Journal of Software*, 2006,17(6):1307−1317. http://www.jos.org.cn/1000-9825/17/1307.htm

Abstract: DSSA (domain-specific software architecture) is essential for the development of applications in the same domain. Feature-based domain model makes the mapping from domain requirements to DSSA possible. In this paper, ontology is introduced as the representation basis for feature models, and feature ontology of a domain is also taken as the semantic description basis for business components in the domain. Then a feature model and component semantics based method for conceptual architecture design are proposed. Commonality, variability, binding time, structural relations, and dependencies are taken into account in the method. The component semantics act as the transition from feature to conceptual component design. The methods have been implemented in the ontology-based feature modeling tool and feature-based architecture development tool, and exhibit effective supports for domain development.

Key words: component; business semantics; domain model; feature; domain ontology; DSSA(domain-specific software architecture); architecture

摘 要: 特定领域软件体系结构(domain-specific software architecture,简称 DSSA)是特定领域开发中的重要资产,而基于特征的领域模型使得从领域需求到 DSSA 的映射成为可能.引入本体作为特征模型的描述基础,通过该方法得到的领域特征本体将同时作为领域内业务构件的语义描述基础存在.在此基础上,提出了基于特征模型和构件语义的概念体系结构设计方法.该方法综合考虑了特征模型中的共性、可变性、绑定时间以及结构关系、依赖关系等对 DSSA 设计的影响,同时以构件语义作为特征到概念构件设计的过渡.相关方法已经实现为基于本体的特征建模工具和基于特征模型的体系结构设计工具,为特征驱动的领域开发提供了有力的支持.

关键词: 构件;业务语义;领域模型;特征;领域本体;特定领域体系结构;体系结构

中图法分类号: TP311 **文献标识码**: A

* Supported by the National Natural Science Foundation of China under Grant No.60473061 (国家自然科学基金); the National High-Tech Research and Development Plan of China under Grant Nos.2004AA113030, 2005AA113120 (国家高技术研究发展计划(863)); the Science Technology Committee of Shanghai under Grant Nos.04DZ15022, 055115005 (上海市科委科技攻关项目)

Received 2006-01-05; Accepted 2006-03-13

软件复用的研究与实践表明,特定领域的软件复用活动相对容易取得成功[1].而领域工程是其中的关键,即可复用软件资产(包括体系结构和构件等)的生产阶段,主要包括领域分析[1]、领域设计和领域实现这 3 个活动.领域分析是领域级的需求工程活动,得到的产物是领域模型.在此基础上,领域设计阶段设计 DSSA,其中的领域通用构件将在领域实现阶段加以实现.DSSA 提供了特定领域的参考体系结构,其中包含面向特定应用的可变性和可扩展性.DSSA 是指导领域构件以及特定应用开发的领域核心资产.因此,如何在领域模型的基础上设计 DSSA 是领域工程的一个关键问题.

领域工程的一个目标是在决策空间(即领域模型)和产品空间(即体系结构模型)之间建立映射关系[2].而领域模型的组织框架则是领域需求模型向构件模型平滑过渡的前提条件[1].文献[3]认为,问题分析模型中的语法元素可以直接用于生成构件规格说明,还有许多研究关注基于分析模型的体系结构设计(见第 1 节).由此可见,基于分析模型的领域体系结构设计完全可以在相关方法支持下自动或半自动地进行.

特征建模[1,4,5]是目前最为流行的领域分析方法.本文在文献[1]中提出的特征建模方法基础上,将特征细分为业务动作、语义刻面、术语等概念,同时引入本体作为领域模型的描述方法,提出了构件端口语义的描述方法,从而以基于本体的概念语义为纽带将特征与构件语义联系起来.本文提出的方法综合考虑了领域模型中的共性、可变性、依赖以及绑定时间等因素,通过构件端口语义设计、构件聚合等具体步骤设计概念构件以及交互关系.本文从领域模型出发,使用 W3C 推荐的 Web 本体语言 OWL 定义了特征模型元模型及构件的端口语义,在此基础上提出了基于特征模型的体系结构设计方法,并结合网上购物领域设计实例说明了方法的有效性.

1 相关工作

在基于构件的软件工程中,业务构件体系结构的定义是一个巨大的挑战[3].而基于分析模型的构件和体系结构设计已经成为共识,并且有着广泛的研究基础.其中既包括基于 OO 分析方法的业务构件和体系结构设计方法 COMO[6]、O2BC[7]及文献[3,8]等,又包括基于特征的方法,如 FORM[2]以及文献[4,5]等.

COMO[6]使用 UML 作为建模工具,在业务构件设计上主要依据用例/类关系矩阵,按照"高内聚低耦合"的原则进行用例和类的聚集.O2BC[7]延续了 COMO 方法的主要思想,在构件建模方法上有所改进,例如,将用例/类关系矩阵改为实体/事件交互矩阵并更加强调耦合分析.类似的方法还有文献[8].文献[3]采取的方法是在分析模型的基础上通过几种业务类型聚合规则获得实体构件,同时使用用例模型分析获取过程构件.这些方法都是基于 OO 分析模型,没有考虑领域共性和可变性及相关的复用需求,因此无法应用于 DSSA 的设计.

随着领域工程和产品线研究的深入,特征建模[1,2,4,5]逐渐成为领域分析的主流方法.基于特征的体系结构设计也成为新的关注点.FORM[2]是一个著名的基于特征的开发方法,它将 FOAD[9]提出的特征建模方法扩展到设计阶段,关注于如何使用特征模型开发领域体系结构和构件.FORM 方法对于基于特征的领域工程和应用工程过程进行了系统的阐述,对于领域体系结构的设计也提出了一些指导性原则,但缺少一个具体的、具备可操作性的方法.文献[5]在特征建模的基础上扩展了特征依赖分析用于产品线构件设计,并提出了相应的构件设计指导方针.这些指导原则针对不同的特征依赖类型分别提出,缺少一种综合的体系结构设计方法.

本文所提出的体系结构设计方法以基于本体的特征模型为基础,结合领域模型中的共性、可变性、依赖关系以及绑定时间等信息,可以为基于特征模型的体系结构设计提供一种系统的、可自动化的方法支持.

2 基于本体的特征模型及构件语义描述

2.1 基于本体的特征模型元模型

特征是需求空间内的一阶实体[1],体现了系统具有的某种能力或特点.在基于本体的领域模型中,这些特征都被表示为概念(concept),并进一步细分为业务动作(action)、动作刻面(facet)和术语(term)等建模元素.其中:Action 是业务动作的语义主体,直接表示功能特征;Facet 是 Action 的语义刻画面,用于精确描述业务动作的细节属性;Term 是 Action 上语义刻面的术语取值,即属性的值.为了表达数值型和布尔型特征,数值类型(xsd:

number)和布尔类型(xsd:boolean)也可作为动作刻面的值域存在.除了操作特征,特征模型中还包括作为操作目标的业务对象(BusinessObject),它与 Action 之间是操作(ActOn)关系.这几类特征是特征模型的主要组成部分.

基于本体的特征元模型如图 1 所示,其中:Action,Term,BusinessObject 和 BindTime (绑定时间)被定义为本体类(owl:Class);Facet 被定义为从定义域(rdfs:domain)Action 到值域(rdfs:range)Term 的关系(owl:Property). Action 和 Term 各自按照子类关系(subClassOf)构成层次结构,分别体现了业务动作以及特征值间的特殊化关系.在此基础上,还可以定义依赖关系、绑定时间等模型元素,它们都被定义成本体关系,将在下面一一加以介绍.

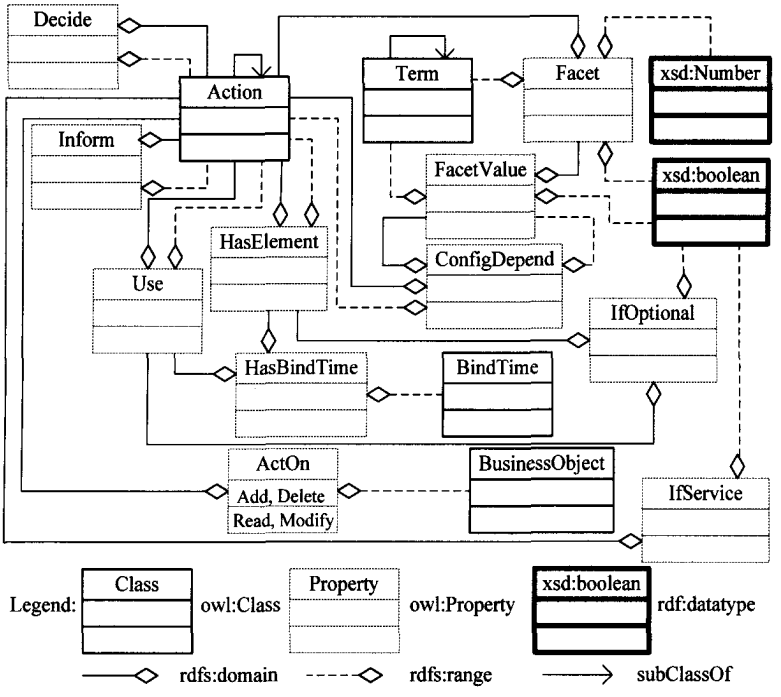

Fig.1 Ontology-Based meta-model of feature model

图 1 基于本体的特征模型元模型

subClassOf:定义在 Action 之间以及 Term 之间,表示直接的特化关系,这个关系在本体模型中可转化为本体的子类关系(rdfs: subClassOf),并保留直接的 subClassOf 关系.本文将分别用 subClassOf 和 rdfs:subClassOf 表示直接的特化关系和本体的子类关系,注意后者在 OWL 中是自反和传递的.

HasElement:另一种细化关系,表示业务动作的分解,父 Action 按照此关系关联一系列子 Action.

Use:表示为了功能的正确实现而对其他操作的依赖关系[5],一般可以理解为功能调用(invoke).

IfOptional:定义在 HasElement 或 Use 关系上的布尔值属性,表示相应的分解或使用关系是否可选.

Decide:表示一个 Action 的执行结果对另一个 Action 多个实现变体的选择具有决定作用,类似于文献[5]中的 modification.由于 Decide 与动态执行结果相关,因此,Decide 关系的目标特征绑定时间必然为引导时或运行时.

Inform:表示一个 Action 执行过程中通过消息发送对另一个的 Action 的通知关系,但并不依赖其功能.

IfService:用于标识两类不同 Action 的属性,取值为 Ture 表示该 Action 是 Service 特征.这种 Action 必须包含 HasElement 关系,并且其分解成分只是一系列相关操作的集合.例如 Edit 对于 Copy,Paste 等.而非 Service 的 Action 如果存在分解关系,则它的一次执行需要必选成分全部执行一次,且一般带有一定的流程控制.

HasBindTime:定义在 HasElement 和 Use 这两种关系上的绑定时间关系,表示可选或可变的成分(或 Use 对象)在什么时间完成对其父 Action(或 Use 源)的最终绑定.其中绑定时间(BindTime)分为 3 种,即组装时

(BuildTime)、引导时(LoadTime)和运行时(RunTime),分别对应于应用组装时、系统引导时和系统运行时.

ActOn:表示 Action 与 BusinessObject 之间的操作关系,按照操作方式分为创建或增加(add)、删除或移除(delete)、读取(read)和修改(modify),与 COMO[6]中的 Create,Delete,Read,Write 类似.

ConfigDepend:非直接交互情况下的配置依赖关系,表示在可变特征绑定时的相互制约关系,定义域和值域均为业务动作及刻面取值(FacetValue).其中,FacetValue 表示动作刻面下的取值假设(术语或布尔值),目的是表达与刻面取值相关的依赖.注意:Use 等依赖关系也隐含着 Action 间的配置依赖关系,这些将在绑定推理中加以处理.由于 OWL 支持以关系(property)作为值域或定义域,因此,上面有些关系定义在其他关系之上,例如,IfOptional 定义在 HasElement 和 Use 之上.领域特征模型包含两种变化性机制,即特征的可选性和特征自身的变化性[1].在基于本体的特征模型中,前者体现为 IfOptional 关系以及值域为布尔型的 Facet 取值,而后者体现在抽象的 Action 以及 Facet 取值上,它们可以在应用组装或运行时确定化为具体语义.这两种变化性的绑定都体现在 HasElement 和 Use 关系上,即变体和可选操作的绑定总是针对使用该可变操作的其他 Action 而言.

从图 1 可以看出:基于本体的特征元模型以功能特征(体现为 action)作为描述主体,但也可以描述非功能性特征,主要体现在值域为数值类型的动作刻面上.利用这种数值刻面可以描述时间约束等常见的非功能性特征.这些非功能特征将指导构件的内部设计和实现过程.为了表达基于本体的特征信息,首先定义几个相关概念.

定义 1(本体关系表示法). 对于任意的 $prop \in Property$ 以及 $subj \in prop.domain(prop$ 的定义域)和 $obj \in prop.range(prop$ 的值域),如果 $subj$ 和 obj 之间存在 $prop$ 关系,则可以表示为$(subj\ prop\ obj)$.

定义 2(语义刻面集). 对于$\forall a \in Action, a$ 的语义刻面集可定义为 $FacetSet(a)=\{f | f \in Facet \wedge (a\ rdfs{:}SubClassOf\ f.domain)\}$,其中,$f.domain$ 表示动作刻面 f 的定义域($rdfs{:}domain$).

定义 3(语义确定化). $\forall a1, a2 \in Action$ 如果满足$(a2\ rdfs{:}subClassOf\ a1)$,则对于$\forall f \in FacetSet(a1)$,都有 $f \in FacetSet(a2)$,且对于非数值型刻面 f 都有$(a2.f\ rdfs{:}subClassOf\ a1.f)$.其中,$a1.f$ 表示业务动作 $a1$ 在语义刻面 f 下的术语取值.$a2$ 与 $a1$ 的这种关系称为语义确定化,即一种对可变语义特征的绑定过程.

2.2 基于本体的特征模型实例

图 2 是基于本体的在线购物领域特征模型片断.

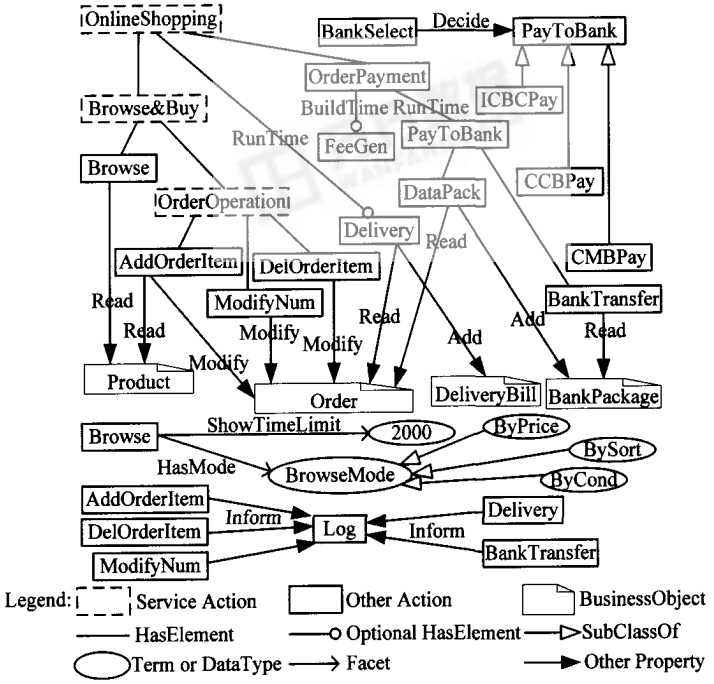

Fig.2 Segment of online shopping domain feature model

图 2 网上购物领域特征模型片断

从图 2 中可以看出,OnlineShopping(在线购物)主要包括必选的 Browse&Buy(浏览/购买)和 OrderPayment(订单支付)以及可选的 Delivery(配送).其中,Browse(浏览)操作上定义了一个数值刻面 ShowTimeLimit(结果显示时间限制)以及一个浏览模式刻面 HasMode.模型中的可变性主要体现在 Delivery 和 FeeGen(手续费生成)对于其父操作的可选性,以及 PayToBank(银行在线支付)自身的可变性上.Delivery 对于 OnlineShopping 的绑定时间是 RunTime,因为是否需要配送要在运行时根据商品类型判断.FeeGen 对于 OrderPayment 的绑定时间是 BuildTime,这意味着每个产品开发时必须决定是否需要收取手续费.PayToBank 的可变性来源于多种不同银行的支付方式,它将在运行时通过执行 BankSelect 操作来决定.图 2 中标识出了 4 个业务对象:Product(商品)、Order(订单)、DeliveryBill(配送单)和 BankPackage(银行交易数据包).其中,BankPackage 表示要发送到银行方的数据,它由 DataPack 操作创建,并由 BankTransfer 操作进行发送.

注意:PayToBank 的 3 个变体 ICBCPay,CCBPay 和 CMBPay 都与 PayToBank 一样包含各自所特有的两个子操作 DataPack 和 BankTransfer,且均包含特定格式的 BankPackage.为了保持简洁,这些都没有在图中体现.

2.3 构件语义描述

领域本体中的业务动作依照语义确定化关系形成业务动作的语义层次.一个业务动作在对另一个业务动作进行语义确定化的同时还会定义新的语义刻面.对于业务构件来说,端口语义总是对应于一个业务动作,并因构件实现方式的不同而具有相应的业务特征.端口语义是确定性与可变性的结合,即端口语义在限定一定语义范围的同时又隐含着此范围内的可变性.由此,我们可以在图 1 的本体元模型基础上定义构件的端口语义.

定义 4(构件端口语义). 构件任何端口 P 的端口语义 $PortSemantics(P)$是一个二元组$\langle a,fv\rangle$,其中 $a\in Action$,fv 是 a 的语义刻面集 $FacetSet(a)$到 Term 的取值函数,且$\forall f\in FacetSet(a)$都有$(fv(f)\ rdfs{:}subClassOf\ a.f)$.

构件的服务端口和请求端口语义均符合端口语义定义,分别表示构件能够对外提供的服务以及要求外界(其他构件)提供的服务.而端口语义之间的关系则表示为语义协议,其中可以体现特征模型中相关的依赖关系.端口语义是对所对应的操作特征语义的进一步确定化,即某些刻面取值的具体化,体现了特定的应用需求或实现决策.例如,图 2 中的 Browse 在实现中可以特化为只支持按条件查找(ByCond)的浏览方式并加以实现.

定义 5(概念构件). 概念构件 Comp 表示为三元组$\langle$SPSet,RPSet,Protocol$\rangle$,其中,SPSet 是服务端口语义的集合;RPSet 是请求端口语义的集合;而 Protocol 是构件的语义协议,体现了各服务/请求端口语义之间的约束关系.

3 领域体系结构设计方法

DSSA 侧重于描述应用系统中的构件组成及接口,说明了系统功能如何在构件间分配[10].因此,确定构件的功能语义、类型(公共构件或抽象构件)以及交互关系等,即确定概念体系结构,是 DSSA 设计的首要任务.

定义 6(概念体系结构). 概念体系结构在问题域和需求的结构(体现为特征模型)基础上,以概念构件为体系结构元素描述系统的抽象视图,定义了各业务构件的功能分配以及交互语义以及体系结构的可变性策略.

文献[4]通过概念体系结构、逻辑体系结构和部署体系结构 3 种视图来定义体系结构,每个视图关注软件体系结构的不同方面.逻辑体系结构建模按照非功能特征对概念体系结构进行精化、调整和转换,而部署体系结构建模关注功能如何分布于计算节点以及如何交互,以满足相关的非功能特征[4].随着工业构件标准(例如 COM,EJB 等)的成熟,当前的信息系统开发主要关注业务逻辑的实现,而在事务处理及构件部署等方面更多地则是以相关标准(例如 J2EE)为依据依托中间件平台完成,因此,本文的方法将主要关注于概念体系结构设计.

3.1 基于特征模型的DSSA设计原则

对于成功的软件复用而言,系统地发现并利用相关软件系统间的共性是一个基本的技术需求[2].文献[5]进一步指出:共性不仅存在于产品间(体现为共性特征),而且存在于特征间(体现为特征的共性依赖).因此,将可复用性最大化是首要原则,而这又依赖于 DSSA 中共性与可变性的良好分离.可变性的绑定时间同样可能对体系结构产生影响,例如,文献[5]在构件设计方针中就对运行时绑定和构建时绑定进行了区分.

文献[2]提出的一个重要原则是:特征模型中的逻辑边界应该对应于体系结构模型中的物理边界,否则,相关

的特征就需要进一步进行分解或细化.这一原则强调了特征与 DSSA 中构件建模元素上的对应关系.其次,"高内聚低耦合"也是一个提及较多的构件设计准则,例如,文献[2]认为具有强数据或控制依赖的特征应该分配到一起以减少通信量,其次还有文献[3,6–8]等.总的来说,基于特征模型的 DSSA 设计原则主要包括:

- 满足特征模型中依赖关系等约束信息,同时以适当的方式支持可变特征的绑定;
- 分离共性和可变性,提高构件的可复用性,将共性最大化;
- 满足特征模型中可变特征的不同绑定时间要求;
- 尽量降低构件的复用成本并提高复用效率,复用成本主要体现为复用者对构件的定制成本,而复用效率主要体现为构件的粒度和功能;
- 保持体系结构模型与特征模型中元素边界的一致性,即在 DSSA 中体现出清晰的逻辑边界.

3.2 方法步骤

功能性特征主要用来标识所需构件,而非功能性特征用来划分构件或者选择构件间的连接器类型[2].基于特征模型的 DSSA 设计的首要任务是以适当的方式将功能特征分配到各个构件上.在本文基于构件语义的方法中,概念体系结构设计体现为构件语义交互关系的设计.在保证语义完整性的前提下,设计过程将以第 3.1 节中提出的设计原则为指导.基于特征模型的 DSSA 设计方法步骤如下:

(1) 在特征模型基础上,按照一个操作特征(action)对应一个特征构件的原则生成初始体系结构;

(2) 在初始体系结构基础上对特征构件进行聚合,并对服务和请求端口进行调整;

(3) 对 IfService 属性为 True 的特征构件进行削减,同时对各种业务对象(BusinessObject)作出处理,例如,独立为实体构件、分配到相应的操作构件中或对应到外部资源上,从而得到完整的领域概念体系结构.

以上各步骤中,初始体系结构(步骤 1)设计的主要目标是满足功能完整性;而构件聚合(步骤 2)则主要考虑构件的可复用性、开发成本、可维护性及复用效率.各步骤详细说明及相关算法将在第 3.3 节中介绍.

3.3 相关步骤及算法

为了后续算法的表述方便,首先需要定义一些特征集合符号.此外,DSSA 可以通过参数化及继承等机制来支持可变性和可扩发展性.因此,还需要定义相关的构件类型.对于$\forall sa \in Action$,相关集合定义如下:

子动作集(其中 *subClassOf* 表示非自反的直接特化关系):$SubSet(sa)=\{a|a \in Action \wedge (a\ subClassOf\ sa)\}$;

整体/部分依赖集:$ElementSet(sa)=\{a|a \in Action \wedge (sa\ HasElement\ a)\}$;

使用依赖集:$UseSet(sa)=\{a|a \in Action \wedge (sa\ Use\ a)\}$;

判断依赖集:$DecideSet(sa)=\{a|a \in Action \wedge (a\ Decide\ sa)\}$;

操作通知集:$InformSet(sa)=\{a|a \in Action \wedge (sa\ Inform\ a)\}$.

定义 7(抽象构件、可选构件和实现构件). 作为占位符出现在体系结构中的构件称为抽象构件,其对应的具体构件需要延迟到应用开发时才能确定.与之对应的是实现构件,一般在领域开发阶段加以实现.而可选构件则是指其他构件与它的交互关系是可选的,如果这种关系都是可选的,则该构件针对整个系统也是可选的.

定义 8(构件交互关系). 概念构件设计中识别两类构件交互,即调用(invoke)和通知(inform).

3.3.1 初始体系结构生成

初始体系结构可以在特征模型基础上,按照一个 Action 对应一个特征构件的原则直接生成.生成过程中还需要按照绑定时间等信息确定特征构件的类型(抽象构件、可选构件或实现构件).实现算法如下:

算法 1. 初始体系结构生成算法.

(1) 将每一个 Action 对应到一个特征构件,然后对于所有的 $sa \in Action$,按照步骤 2 和步骤 3 确定 *sa* 对整个系统的绑定时间以及对应的特征构件类型,再按照步骤 4~步骤 7 建立特征构件之间的关系;

(2) 如果集合 $insSet=\{ins|ins=(a\ HasElement\ sa) \vee ins=(a\ Use\ sa), a \in Action\} \neq \phi$ 且 $\forall ins \in insSet$ 都有 $(ins\ IfOptional\ True)$,则 *sa* 对应的特征构件是可选构件,否则再按照步骤 3 判断是实现构件还是抽象构件;

(3) 对集合 $\{bindtime|\exists ins=(a\ HasElement/Use\ sa) \wedge (ins\ HasBindTime\ bindtime)\}$ 中全部的绑定时间,取最晚

的那种绑定时间作为 *sa* 对系统的绑定时间,如果是 *BuildTime*,则对应的特征构件是抽象构件,否则是实现构件(包括绑定时间集合为空);

(4) 如果 *sa* 对应的是抽象构件,则从体系结构中删去所有满足 $a \neq sa \wedge (a\ rdfs{:}\ subClassOf\ sa)$条件的 *a* 所对应的特征构件,而 *sa* 对应的特征构件则作为占位符存在;

(5) 如果 *sa* 对应的是抽象构件或者 $SubSet(sa)=\varnothing$,则在 *sa* 和所有的 $ta \in ElementSet(sa) \cup UseSet(sa)$对应的特征构件之间建立调用关系,如果相应的特征关系(*HasElement* 或 *Use*)是可选的,则该交互关系也是可选的;

(6) 如果 *sa* 对应的不是抽象构件并且 $SubSet(sa) \neq \varnothing$,则在 *sa* 和所有的 $ta \in SubSet(sa) \cup DecideSet(sa)$对应的特征构件之间建立调用关系,并将该构件标志为代理构件;

(7) 在 *sa* 和所有的 $ta \in InformSet(sa)$之间建立通知关系.

在此算法中,构件类型判断主要以关系的可选性和绑定时间为依据.如果存在多种绑定时间,则以最晚的时间为准.在对自身可变的 Action 处理上考虑了总体绑定时间,如果是组装时,则设计为抽象构件;否则,设计为代理构件.这种方法与文献[5]中单一的工厂模式实现方式相比更加灵活,减少了 DSSA 中不必要的复杂性.这样得到的特征构件包括 3 种类型:完成原子功能的实体构件(对应于叶子 Action)、实现下层操作交互流程控制的框架构件(对应于拥有分解关系的 Action)以及提供自身变体选择的代理构件(对应于包含变体的 Action).

3.3.2 特征构件聚合

构件(而不是端口)是独立开发并参与复用的基本单位.完成全部端口设计后,DSSA 中的功能完整性已经可以满足,但考虑到构件的开发成本以及复用的效率,还需要将仅有一个服务端口的基本构件进行聚合,从而得到完整的领域概念体系结构.降低构件开发成本要求减少独立构件的数量并提高单个构件的内聚度,而降低复用成本则要求构件粒度尽量大.产品线资产的设计必须考虑如何使可变特征的选取对其特征的影响尽可能小,因此,特征的可变性必须局限在一个或少数几个构件中[5].在此过程中,还需要保持构件的可复用性,避免因为构件聚合而降低原有构件的可复用性.合并后,参与合并的两个构件间的交互变成内部交互关系.

特征构件聚合过程将以聚合度计算为判断标准,因此,将分别介绍特征构件聚合过程和特征构件聚合度评估算法.聚合过程按照特征功能关系自上而下进行,顶层特征集合 TopSet 的定义如下:

$$TopSet=\{sa \mid sa \in Action \wedge (\nexists\ a \in Action.\ (sa\ subClassOf\ a) \vee (a\ HasElement\ sa) \vee (a\ Use\ sa) \vee (sa\ Decide\ a))\}.$$

特征构件聚合过程以算法 1 产生的初始体系结构作为处理对象,其中特征构件间的交互关系来源于 HasElement,Use,Decide,subClassOf,Inform 这些特征关系,算法 2 的聚合过程中将使用这些信息.由于抽象构件仅代表体系结构中的占位符,因此,算法 2 的聚合过程不考虑抽象构件以及它们与其他构件的交互关系.

算法 2. 特征构件聚合过程.

(1) 将所有特征构件置为未处理状态,以 TopSet 所对应的特征构件集合作为队列 queue 的初始元素集;

(2) 如果 queue 为空,算法结束;否则取出 queue 的队首特征构件 head;

(3) 考虑集合{comp|head 与 comp 有必选的 HasElement 交互}与 head 之间的整体聚合度(见算法 3),如果高于阈值,则将其中的特征构件与 head 所在的构件合并;

(4) 对于所有 head 与之有 Use 交互的构件 comp,如果二者满足以下 3 个条件则将它们合并,并将该交互关系变为内部交互:a) 它们的交互关系是必选的;b) comp 不是代理构件;c) 其他构件对 comp 没有交互关系;

(5) 将 head 置为已处理,并将所有 head 所调用或通知的未处理特征构件一一加入到 queue 中,转步骤(2).

注意:此算法中的考察对象是与特征一一对应的特征构件,如步骤(3)中必选成分构件集与 head 合并后还可能加入 queue(步骤(5))继续对外执行合并.这些逐层进行的聚合过程最终可能产生一些大粒度构件.该算法主要考虑 HasElement 和 Use 两种交互关系带来的特征构件合并,抽象构件与其构件的区分则是聚合的首要原则.其中 HasElement 关系相关的聚合将以全部必选成分的整体聚合度为标准(见算法 3),而 Use 关系的聚合则有较高的专有性要求.为了保持变体选择的独立性和外部可控性,代理构件不参与聚合.构件合并后相关的服务端口和请求端口将相应地进行合并,其中每一个服务端口对应一个操作特征.

算法 2 中,步骤 3 的合并是构件聚合的主要形式,其目标是提高作为独立开发和复用基本单位的构件内聚

度、构件复用效率以及系统整体结构的清晰性.而相关构件的整体相关性是评价标准.对于特征构件 *comp* 以及它的必选 HasElement 交互集合 $elemSet=\{comp_1,comp_2,\ldots,comp_n\}$,定义它们的整体聚合度如下:

算法 3. 特征构件聚合度算法.

(1) 设对 *elemSet* 的构件交互集合 $set1=\{c|$存在 $comp_i(1\le i\le n)$满足 c 对 $comp_i$ 有 invoke 或 inform 关系$\}$,如果 $x=|set1|>0$,则第 1 部分聚合度 $p_1=|\{(c_1,c_2)|c_1\in set1,c_2\in elemSet$ 且 c_1 对 c_2 有交互关系$\}|$,否则令 $p_1=n,x=1$;

(2) 设 *elemSet* 中构件的操作对象集合 $Set2=\{o|$存在 $comp_i(1\le i\le n)$满足 $comp_i$ 对 o 有操作关系$\}$,如果 $y=|set2|>0$,则第 2 部分聚合度 $p_2=|\{(c,o)|c\in elemSet,o\in set2$ 且 c 对 o 有操作关系$\}|$,否则令 $p_2=n,y=1$;

(3) 设 *elemSet* 对其他构件的交互集合 $set3=\{c|$存在 $comp_i(1\le i\le n)$满足 $comp_i$ 对 c 有 invoke 或 inform 关系$\}$,如果 $z=|set3|>0$,则第 3 部分聚合度 $p_3=|\{(c_1,c_2)|c_1\in elemSet,c_2\in set3$ 且 c_1 对 c_2 有交互关系$\}|$,否则令 $p_3=n,z=1$;

(4) *comp* 与 *elemSet* 的整体聚合度为 $p=(p_1/x+p_2/y+p_3/z)/n$.

3.3.3 Service 特征及业务对象处理

完成特征构件聚合后,还需要在特征模型和概念体系结构基础上对 Service 特征和业务对象作出相应处理.

算法 4. Service 特征及业务对象处理.

(1) 对于所有满足 $sa\in Action\wedge(sa\ IfService$ True)的特征 *sa* 执行步骤 2,对于所有业务对象 *obj* 执行步骤 3,步骤 4;

(2) 如果 *sa* 所对应的特征构件 *comp* 在聚合过程中没有与其他构件合并,则将 *comp* 从概念体系结构中删除,并将所有指向 *comp* 的交互关系复制后转向所有满足(*sa HasElement a*)的特征 *a* 上;

(3) 对 *obj* 进行设计分析,如果该业务对象表现为系统外部对象(例如,操作系统单元、文件),则在体系结构中标注为外部资源,否则在体系结构中创建相应的实体构件与之对应,并建立操作构件与实体构件间的交互关系;

(4) 如果 *obj* 对应的外部资源或实体构件仅与一个操作构件相关,则将其分配到该构件中作为内部成员.

此算法对特征模型中的 Service 特征和业务对象进行处理.Service 特征作为下层操作特征的组织单位存在,本身并不存在对应的计算语义(如框架构件的流程控制),因此完成聚合分析后需要移除,相关交互由其成员构件承担(根据 IfService 定义这样的分解一定存在).业务对象的处理则将完成系统中主要的实体构件设计,但首先要分析业务对象的系统形态,因为它们可能对应于外部资源,例如图 2 中的 BankPackage 就体现为 XML 文件.

3.4 构件及连接器实现设计

概念体系结构为构件间的功能分配和交互边界.在此基础上,可以根据特征依赖关系确定构件的语义协议,即各端口间的交互及语义约束,并根据系统部署和运行要求为各个构件选择特定的构件模型(例如 EJB,Web Service 等),这些信息将指导构件的内部设计和开发.连接器作为构件间的交互通道,将以满足构件交互适配和性能要求为设计目标.非功能特征也会影响体系结构设计.它们一般不能被精确定位到特定构件,但会影响系统的整体结构和行为[4].非功能性需求在特征模型可以表示为数值刻面,它们将体现在构件的端口语义中,这样,连接器设计时就能根据交互两端的性能需求作出相应的设计决策.

完成 DSSA 设计后,实现构件和标识出的可选或可变构件的实现变体都可以在领域实现过程中加以实现.由于框架是 DSSA 的部分实现[10],因此,领域工程活动结束后得到的是大粒度的可复用产品——领域框架.其中的可选构件和抽象构件将在应用工程中进行定制,并对标识出的特定应用构件进行开发实现.

4 实例研究及评估

这部分将以图 2 中的网上购物领域为例,说明基于特征模型和构件语义的体系结构设计方法并进行评估.

4.1 初始体系结构设计以及构件聚合

在图 2 的特征模型基础上,通过执行算法 1 很容易获得初始体系结构,其中,FeeGen 由于其总体绑定时间为 BuildTime,因此被确定为抽象构件.同时,由于对其父特征的可选性,FeeGen 和 Delivery 构件被识别为可选.而

PayToBank 将作为对 3 种银行交易方式的代理构件存在,并请求 BankSelect 构件作为实现变体的选择依据.

在初始体系结构基础上执行构件聚合以及 Service 构件和业务对象处理后(算法 2 和算法 3)得到如图 3 所示的概念体系结构.其中:实心方块代表服务端口,空心方块代表请求端口,各服务端口上所对应的特征语义也进行了标注,而聚合过程中被合并的操作特征则标注在构件内部.聚合主要发生在 OrderOperation 和 OrderPayment 上.图 2 中,OrderOperation 的 3 个子特征 AddOrderItem,DelOrderItem 和 ModifyNum 都与业务对象 Order 存在 Modify 关系,因此具有较高的聚合度.而特征构件 OrderPayment 与 PayToBank(Proxy)则同样由于其专有的调用关系而表现出较高的聚合度,合并后二者间的交互端口变为内部成分,而后者的请求端口则体现为合并后构件的请求端口.图 2 中的 4 个业务对象中,BankPackage 被标志为外部数据包,因此,3 种银行的 BankPackage 被标识为外部资源,并分别被分配到相应的支付构件中.DeliveryBill 同样由于仅与 Delivery 构件有操作关系而被分配到 Delivery 中成为内部对象.Product 和 Order 由于在多个构件间共享,因此独立为实体构件.它们和 DeliveryBill 一样作为实体信息的封装对象存在,对业务对象提供对象数据操作服务,并负责完成与数据库之间的映射操作.这样得到的体系结构将具有良好的分层特性,即以操作构件实现业务逻辑,以实体对象完成数据层的封装.

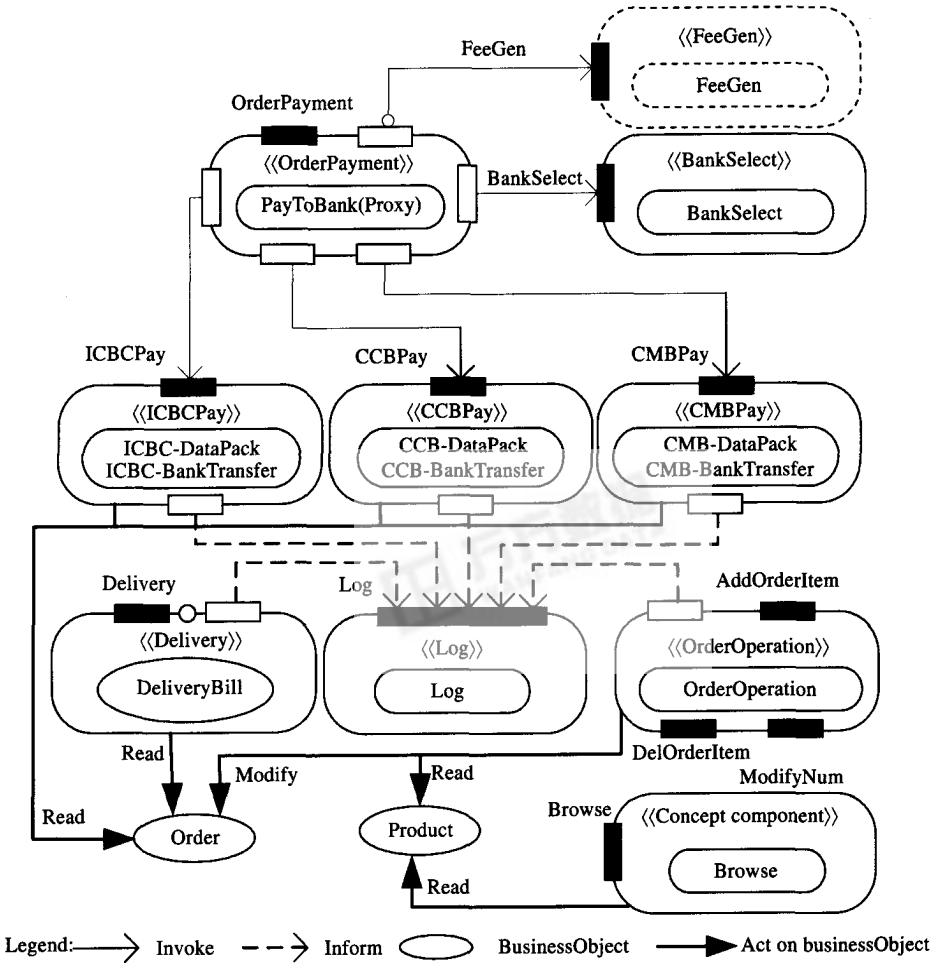

Fig.3 Conceptual DSSA segment of online shopping domain

图 3 网上购物领域概念体系结构片断

4.2 方法评估

基于本体、特征驱动的概念体系结构设计以构件的特征语义设计作为特征模型到体系结构模型的过渡手

段.在这个过程中,基于本体的特征模型充当了需求模型和领域语义描述基础的双重角色.特征本体对于相关概念间的关系作出了明确定义,因此,基于特征本体的构件语义描述使得体系结构的分析验证成为可能.基于本体的特征建模可以利用本体的推理能力完成特征模型的一致性检查.当架构师对 DSSA 进行修改或面向特定应用进行定制后,还可以使用类似的本体推理方法检查体系结构模型与特征模型的一致性.目前,相关的特征建模和体系结构设计方法已经实现为相应的特征建模工具 OntoFeature 和体系结构设计工具 OntoArch.其中,OntoFeature 支持 OWL 模型的自动生成;而 OntoArch 以该 OWL 文件作为输入并自动生成概念体系结构.Jena 被用作 OWL 处理工具,并通过定义与特征依赖以及绑定相关的 Jena 推理规则实现相关推理过程.OntoArch 还提供了体系结构编辑、版本管理支持、体系结构描述语言(ADL)自动生成以及基于推理的 DSSA 定制指导和体系结构一致性检查.这样,架构师可以在工具的支持下对自动生成的概念体系结构进行编辑和定制.

一些著名的 SA 分析方法,例如 SEI 的 ATAM(architecture tradeoff analysis method)[11],关注于 SA 正确性基础上的各种质量属性(例如可修改性、性能等)的决策权衡.ATAM 的目标是按照质量需求评价体系结构设计[11].由于 ATAM 方法必须在已经建立好的体系结构基础上进行评估,因此无法用于特征驱动的领域开发的正向阶段,也没有考虑领域开发特性.但 ATAM 方法对于质量属性的评估可以用于 DSSA 设计完成后的质量评估,这也是目前我们的方法所欠缺的,我们将在今后的工作中对此进行深入研究.

5 结 论

基于特征的领域模型以特征作为需求空间的建模元素,使需求向体系结构的映射成为可能.在基于构件的领域工程中,DSSA 设计必须以可复用性为首要目标.特征的结构关系、配置依赖以及其他依赖关系对于开发可复用、可适应性的产品线资产有着十分重要的影响[5].本文提出的方法综合考虑了这些因素对于体系结构以及构件的可复用性、复用成本及效率的影响,同时以本体作为特征模型和构件端口语义的描述基础,从而为领域模型到构件语义的平滑过渡创造了条件.本文的方法主要针对概念体系结构的设计,而实际的体系结构设计过程还需要考虑实现技术及平台等因素.第 3 方构件的引入同样也会影响体系结构的设计.我们将在今后的工作中进一步对这些方面进行研究,使得相关方法和工具能够更好地辅助领域工程活动的开展.

References:

[1] Zhang W, Mei H. A feature-oriented domain model and its modeling process. Journal of Software, 2003,14(8):1345−1356 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/14/1345.htm

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

[3] Teixeira HV, Braga RM, Werner CM. Model-Based generation of business component architectures. In: Steinmetz R, Mauthe A, eds. Proc. of the 30th EUROMICRO Conf. (EUROMICRO 2004). Rennes: IEEE Computer Society Press, 2004. 176−183.

[4] Liu DY, Mei H. From requirements to software architecture: A feature oriented mapping approach. Acta Scientiarum Naturalium Universitatis Pekinensis, 2004,40(3):372−378 (in Chinese with English abstract).

[5] Lee K, Kang KC. Feature dependency analysis for product line component design. In: Bosch J, Krueger C, eds. Proc. of the 8th Int'l Conf. on Software Reuse. LNCS 3107, Springer-Verlag, 2004. 69−85.

[6] Lee SD, Yang YJ, Cho ES, Kim SD, Rhew SY. COMO: A UML-based component development methodology. In: Proc. of the 6th Asia Pacific Software Engineering Conf. Takamatsu: IEEE Computer Society Press, 1998. 54−63.

[7] Ganesan R, Sengupta S. O2BC: A technique for the design of component-based applications. In: Li QY, Riehle R, Pour G, Meyer B, eds. Proc. of the 39th Int'l Conf. and Exhibition on Technology of Object-Oriented Languages and Systems. Santa Barbara: IEEE Computer Society Press, 2001. 46−55.

[8] Xu W, Yin BL, Li ZY. Research on the business component design of enterprise information system. Journal of Software, 2003,14(7):1213−1220 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/14/1213.htm

[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. Pittsburgh: Software Engineering Institute, Carnegie Mellon University, 1990. 1−52.

[10] Hu WH, Zhao W, Zhang SK, Wang LF. Study of application framework meta-model based on component technology. Journal of Software, 2004,15(1):1−8 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/15/1.htm

[11] Kazman R, Klein M, Clements P. Method for architecture evaluation. Technical Report, CMU/SEI-2000-TR-004, Pittsburgh: Software Engineering Institute, Carnegie Mellon University, 2000. 1−83.

附中文参考文献:

[1] 张伟,梅宏.一种面向特征的领域模型及其建模过程.软件学报,2003,14(8):1345−1356. http://www.jos.org.cn/1000-9825/14/1345.htm

[4] 刘冬云,梅宏.从需求到软件体系结构:一种面向特征的映射方法.北京大学学报(自然科学版),2004,40(3):372−378.

[8] 徐玮,尹宝林,李昭原.企业信息系统业务构件设计研究.软件学报,2003,14(7):1213−1220. http://www.jos.org.cn/1000-9825/14/1213.htm

[10] 胡文蕙,赵文,张世琨,王立福.基于构件技术的应用框架元模型的研究.软件学报,2004,15(1):1−8. http://www.jos.org.cn/1000-9825/15/1.htm

彭鑫(1979−),男,湖北黄冈人,博士生,主要研究领域为软件再工程,软件体系结构,构件技术.

刘奕明(1978−),男,博士生,主要研究领域为领域工程,软件体系结构.

赵文耘(1964−),男,教授,博士生导师,CCF高级会员,主要研究领域为软件工程,电子商务.