基于体系结构的模型转换语义描述框架

侯金奎[1] 王 磊[1,2]
(潍坊学院计算机工程学院 潍坊 261061)[1] (山东大学计算机科学与技术学院 济南 250101)[2]

摘 要 为解决模型驱动的软件开发过程中语义特性的描述和验证等方面的问题,在对类型范畴理论进行扩充的基础上,为构件式软件模型的描述、转换以及转换过程中语义特性的保持和验证建立了一种统一的语义描述框架。范畴图表描述了体系结构模型的语义,类型态射蕴含了构件对象之间的依赖关系,类型函子用来刻画模型转换前后的映射机制。应用研究表明,该框架很好地遵循了模型驱动的软件开发理念和实质要求,为基于模型和模型转换的软件开发研究提供了新的理解和认知学习的指导架构。

关键词 计算机软件,模型驱动开发,模型转换,构件式软件,语义描述
中图法分类号 TP311.5 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2017.04.032

Formal Framework of Architecture-based Model Transformation

HOU Jin-kui[1] WANG Lei[1,2]
(School of Computer Engineering, Weifang University, Weifang 261061, China)[1]
(School of Computer Science and Technology, Shandong University, Jinan 250101, China)[2]

Abstract To resolve the problems of semantic feature description and calculation of model-driven software development, a semantic description framework for model transformation was proposed based on the extension of typed category theory. The framework can be used to formally describe component-based software model, model mapping, and semantic verification of model transformation. Category diagram is used to depict structural semantics of architecture model, in which typed morphisms are tools to describe the relations between components. Mapping mechanism between models before and after transformation is formally described by typed functors. The application research shows that the framework nicely follows the essence and requirements of model-driven development, and provides a new guide for the understanding, cognitive learning and propulsion of model-driven software development.

Keywords Computer software, Model-driven development, Model transformation, Component-based software, Semantic description

1 引言

进入新世纪以来,IT行业发展更加迅速,计算设备的能力不断提高,数量飞速增长,新技术不断涌现,爆炸式的信息增长导致软件的复杂性也日益增加,相应的软件开发和管理成本大幅增加,这已成为一个潜在的巨大危机[1]。采用模型驱动的软件开发MDSD(Model-Driven Software Development)方法是应对复杂性并提升软件开发效率和质量的一种有效手段[2]。

模型转换是模型驱动的软件开发中的一个重要环节和关键技术,它必须保证转换后的目标模型包含或保持了源模型中的语义信息和特性[2]。而语义特性的描述和验证一直是模型驱动开发相关研究中所缺乏的一项理论和技术[2-3]。已有策略[4-6]都是针对特定的应用领域和场景来研究模型转换前后需要保持的特性约束,并要求该场景下的模型转换都必须保持这些特性。但这些工作并不能为一般意义下模型转换的语义特性计算和考察提供支持。

范畴理论[7]具有强大的知识表达和推理能力,可为软件系统的层次结构提供形式化的语义描述。在本文研究的前期工作[8]中,已将范畴理论用于软构件模型及其之间关联与映射关系的描述。本文在此基础上进行进一步研究,将体系结构特征和类型范畴理论有机地结合在一起,为基于模型和模型转换的软件开发建立了一种形式化的语义描述框架,使构件模型的分析、建立、转换、精化和验证等活动都能在一致的语义基础上进行,从而给模型驱动的软件开发提供了语义特性计算能力的支持。

到稿日期:2015-11-30 返修日期:2016-03-01 本文受山东省科技发展计划项目(2011YD01042,2011YD01043),山东省星火计划项目(2011XH06017),山东省高等学校优秀中青年骨干教师国际合作培养计划(2013)资助。
侯金奎(1976—),男,博士,副教授,CCF会员,主要研究方向为软件工程、服务计算和形式化方法,E-mail:jkhou@163.com;王 磊(1982—),男,博士,讲师,主要研究方向为服务计算、数据可视化分析。

2　软构件与体系结构特征

为强化构件复用机制的描述能力，建立基于体系结构特征的构件模型。

2.1　体系结构特征

参考文献[9-10]，给出软件体系结构特征的相关概念如下。

定义 1(软件体系结构原子特征)　软件体系结构原子特征是描述软构件及其配置语义的单位成分，是对计算机软件和硬件相关知识的本质抽象，表现为描述软件体系结构所提供的基础概念或术语。形式上，一个软件体系结构原子特征是一个四元组 $AF=\langle ID_F, TP_F, f_{Val}, f_{Itp}\rangle$，其中：

(1) ID_F 是特征标识符；

(2) TP_F 是特征的类型；

(3) $f_{Val}:\{ID_F\}\rightarrow\{v|v\in dom(f_{Val})\}$ 是特征取值函数，给出了特征在给定环境下可能的取值；

(4) $f_{Itp}:\{ID_F\}\rightarrow\{des_i|i>0\}$ 是特征的解释函数，是用相关的概念元素对该特征的自然语言解释，并有以下两个条件成立。

(4.1) $\exists j>0$，有 $f_{Itp}(ID_{Fj})=\{des_m\}\wedge f_{Itp}(ID_{Fj})=\{des_n\}\wedge m\neq n$；

(4.2) $f_{Itp}(ID_{Fm})=f_{Itp}(ID_{Fn})\Rightarrow ID_{Fm}=ID_{Fn}$。

最后两个条件表明，在不同的语境下对同一特征允许存在多个含义一致但形式不同的自然语言释义。

软件体系结构原子特征是一个公众认可的语义域，其中的元素描述了在软件系统的相关环境中可用来定义体系结构的一些重要特性。这些特性以计算机软件系统的运行机制和概念结构为基础，涵盖硬件、软件、服务等多个方面，如过程、中断、存储、控制、消息、事件、进程、事务、通信以及代码结构等。这些特性的描述可进行组合和扩展，形成不同粒度、不同层次的复合特征，为不同抽象层的软件模型描述提供语义基础。例如，存在特征〈FID001，Integer，500，{ms}〉，它是有关数据存储的一种表示，其自然语言的释义指 500 毫秒；而另一特征〈FID002，Integer，500，{$}〉则表示 500 美元。若去除特征的解释函数，单从属性-值的对偶意义上来说，二者在机器内的存储和描述机制并无区别。

定义 2(体系结构特征空间)　体系结构特征空间是由软件体系结构特征以及这些特征之间的依赖关系构成的语义域。这里的特征之间有着严格的语义逻辑关系和明确的层次细化分类方式，形式化定义为一个三元组 $SP=\langle S_{AF}, S_{FR}, S_{CST}\rangle$，其中：

(1) $S_{AF}=\{AF_i|1\leqslant i\leqslant n\}$，$AF_i$ 为体系结构特征；

(2) $S_{FR}=\{\langle AF_i, AF_j\rangle|1\leqslant i,j\leqslant n\}$，$\langle AF_i, AF_j\rangle$ 表示特征 AF_i 与 AF_j 之间的依赖关系；

(3) S_{CST} 是由对特征及其关系的约束组成的集合。

体系结构特征空间表现为特征和特征关系的概念集合，既包括特征元素的实例，又有元素语义的变化规则。特征实例的数目体现了特征空间潜在的可复用能力。在体系结构特征空间中，基于特征之间的依赖和约束关系，复合特征可以描述为一系列配置的集合。

定义 3(复合特征)　复合特征是由体系结构原子特征或其他复合特征组成的特征子空间，它给出了这些特征以及特征间的依赖关系构成的语义域，形式化描述为 $CAF=\langle CS_{AF}, CS_{FR}\rangle$，其中 $CS_{AF}\subseteq S_{AF}$，$CS_{FR}\subseteq S_{FR}$。

2.2　软构件

在基于特征的构件描述中，构件定义了一个体系结构特征子空间，其满足语法正确性和语义完整性，是具有可复用价值的单位软件[10]。

定义 4(构件标识特征)　构件标识特征是一个六元组 $CSF=\langle ID_{CP}, SP_{CF}, SET_{Dsc}, SET_{Prd}, SET_{Req}, SET_{Act}\rangle$，其中：

(1) ID_{CP} 为构件的标志；

(2) $SP_{CF}=\langle CS_{AF}, CS_{FR}\rangle$ 为体系结构特征子空间，CS_{AF} 包含了构件中所有的体系结构特征，CS_{FR} 包含了这些特征间的依赖关系；

(3) $SET_{Dsc}\subset CS_{AF}$ 是关于构件的功能、属性及其它信息描述相关的体系结构特征的集合；

(4) $SET_{Prd}\subset CS_{AF}$ 是构件对外部环境所能提供的特征构成的集合；

(5) $SET_{Req}\subset CS_{AF}$ 是构件对外部环境所需求的特征构成的集合；

(6) $SET_{Act}\subset CS_{AF}$ 是与构件的动作描述相关的特征构成的集合。

构件覆盖的体系结构特征的数目反映了构件的粒度。原子构件的粒度最小，通常实现为体系结构特征空间内的原子特征，其复用度最高，但因粒度最小，故复用效率最低[9]。复合构件由原子构件组合而成，其粒度比原子构件大，复用效率高，包含了较多的结构特征，通常复用成本也较高。

定义 5(构件规范)　构件规范是一个八元组 $SCP=\langle CSF, N_{CP}, DS_{Info}, DS_{Prd}, DS_{Req}, DS_{Act}, DS_{Bhv}, f_{AP}\rangle$，其中：

(1) $CSF=\langle ID_{CP}, SP_{CF}, SET_{Dsc}, SET_{Prd}, SET_{Req}, SET_{Act}\rangle$ 是构件标识特征。

(2) N_{CP} 是构件的名字。

(3) DS_{Info} 是关于构件的功能、性能、属性、运行、分类以及资源位置等描述信息构成的集合。这些信息的语义由集合 SET_{Dsc} 中的体系结构特征或特征的组合来提供。

(4) DS_{Prd} 是由描述构件服务端口的信息构成的集合，这些信息描述了该构件可向其他构件提供的服务及方式。服务端口信息的语义由集合 SET_{Prd} 中的体系结构特征或特征的组合来提供。

(5) DS_{Req} 是由描述构件请求端口的信息构成的集合，这些信息描述了该构件请求其他构件提供的服务及方式。请求端口信息的语义由集合 SET_{Req} 中的体系结构特征或特征的组合来提供。

(6) DS_{Act} 是由构件的行为动作构成的集合。构件通过端口和操作与外部环境交互，完成消息的发送和接收，这些收发动作触发构件内部行为，引起状态的变迁。构件动作分外部

可见动作和内部动作，其语义由集合 SET_{Act} 中的体系结构特征或特征的组合来提供。

(7) DS_{Bhv} 是构件的外部可见行为，由 DS_{Act} 中的动作序列来刻画。

(8) $f_{AP}:DS_{Act}\rightarrow DS_{Prd}\cup DS_{Req}$ 描述了构件的行为动作与端口的操作及消息之间的关联和对应关系。

构件模型以软件体系结构特征为语义基础，整个语义实体是对构件属性和功能的抽象与描述，构件端口反映了构件之间的通信规范及语法约束。这样的语义便于被机器理解，从而为约束规则的自动推导和构件的自动化组装奠定了基础。

2.3 体系结构配置

体系结构特征之间的依赖关系决定了构件之间的关系及其实现机制。一般意义上，构件之间的关系可分为关联、组合与依赖 3 类，特征之间的依赖关系分为 4 种[9]：特征完整性依赖、特征取值依赖、特征多值依赖、特征语义依赖。其中，特征完整性依赖反映了特征间的“供-需”关系，是构件组合关系的语义基础；特征取值依赖和特征多值依赖刻画的是一种“值-值”依赖关系，即一组特征的取值受另一组特征取值的影响，它是构件之间依赖关系描述的语义基础；特征语义依赖是指特征之间的语义约束关系（如时序约束、功能耦合、事务逻辑等），反映到构件层面上就是构件之间的各种关联关系。

依据定义 4 和定义 5，构件的设计过程可看作：遵循一定的规范化设计原则[9]，将体系结构特征空间 SP 划分为一系列的子空间 $\{SP_1,SP_2,\cdots,SP_n\}$，每一个子空间 SP_i 都映射为一个独立的构件；再按照特征之间的语义依赖关系建立构件之间的组合和交互机制，这样即可逐步完成整个体系结构配置的设计。每个构件都能向外部环境提供或多或少的一组服务，按照业务逻辑，通过端口实现组装和连接，构成一个结构配置，完成一个或多个领域模型的功能目标，解释为特定应用领域中相应的业务处理；还可通过对这个配置中的构件进行增删、组装以及调整，重构业务模型。系统的运行环境以及需求会不断发生变化，这要求软件必须具备一定的适应能力。构件式的软件系统可通过构件的粒度调整来实现配置的演化，如加入新的功能构件、构件升级、淘汰过时的构件，必要时还可实现构件的迁移等。

3 体系结构模型及转换的语义描述

本文建立的形式化描述框架以类型范畴理论为基础，有关范畴理论的知识可参阅文献[7,11]，此处不再赘述。

3.1 构件对象

从概念上看，范畴是由一个对象集合和一个关联两个对象的箭头集合构成的[7]。当其用于构造软件规范时，对象是以某种逻辑表达的规范，箭头定义了这些规范之间的关系。本文以定义 5 中的软构件规范的实例作为范畴对象（称为构件对象），同时作为对范畴理论的扩充，规定这里的对象是有类型区分的（即构件类型）。

定义 6（构件对象） 构件对象是一个三元组 $OBJ_{CMP}=\langle ID_{CMP},SCP,TP_{CMP}\rangle$，其中：

(1) ID_{CMP} 是构件对象在系统中的唯一标志；

(2) $SCP=\langle CSF,N_{CP},DS_{Info},DS_{Prd},DS_{Req},DS_{Act},DS_{Bhv},f_{AP}\rangle$ 是构件的规范描述；

(3) TP_{CMP} 是构件对象在特定应用领域下的类型说明。

从本质上来说，构件的类型是由其所包含的体系结构特征所决定的，但在不同的应用领域有不同的划分类别，这属于特定应用领域下的不同释义。如在电子商务领域，可按照构件在应用系统中所处的层次将其划分为资源类、业务支撑类、商务应用类、应用适配类以及展现类等。

3.2 构件态射

在构件式软件系统中，构件之间的关联关系与通信的实现方式（即构件之间的交互机制）在很大程度上直接决定着系统的运行效率，当然也影响着软件的开发效率。构件间的交互机制表现为构件端口之间的交互及语义约束，是由构件包含的体系结构特征间的依赖关系所决定的。本文用范畴理论中的态射来表示构件对象之间的关系。

定义 7（构件态射） 给定两个构件对象 $OBJ_{CMP1}=\langle ID_{CMP1},SCP_1,TP_{CMP1}\rangle$ 和 $OBJ_{CMP2}=\langle ID_{CMP2},SCP_2,TP_{CMP2}\rangle$，其中，$SCP_1=\langle CSF_1,N_{CP1},DS_{Info1},DS_{Prd1},DS_{Req1},DS_{Act1},DS_{Bhv1},f_{AP1}\rangle$，$SCP_2=\langle CSF_2,N_{CP2},DS_{Info2},DS_{Prd2},DS_{Req2},DS_{Act2},DS_{Bhv2},f_{AP2}\rangle$。从 OBJ_{CMP1} 到 OBJ_{CMP2} 的态射是一个三元组 $\gamma=\langle OBJ_{CMP1},OBJ_{CMP2},TP_{MPS}\rangle$（为表示直观，通常描述为 $\gamma:OBJ_{CMP1}\rightarrow OBJ_{CMP2}$）形式，包括如下内容。

(1) $\gamma_{SCP}:SCP_1\rightarrow SCP_2$ 是构件规范描述之间的映射。

(1.1) $\gamma_{CSF}:CSF_1\rightarrow CSF_2$ 描述了构件标识特征之间的映射关系；

(1.2) $\gamma_{Info}:DS_{Info1}\rightarrow DS_{Info2}$ 是属性、功能等描述信息间的映射；

(1.3) $\gamma_{Prd}:DS_{Prd1}\rightarrow DS_{Prd2}$ 是服务端口间的映射；

(1.4) $\gamma_{Req}:DS_{Req1}\rightarrow DS_{Req2}$ 是请求端口间的映射；

(1.5) $\gamma_{Act}:DS_{Act1}\rightarrow DS_{Act2}$ 是动作之间的映射；

(1.6) $\gamma_{Bhv}:DS_{Bhv1}\rightarrow DS_{Bhv2}$ 是外部行为描述间的映射。

(2) $\gamma_{TP}:TP_{CMP1}\rightarrow TP_{CMP2}$ 是类型说明之间的映射关系。

(3) TP_{MPS} 说明了态射的类型。

根据 2.3 节中的分析，通常构件之间的关系有关联、组合与依赖 3 种，那么构件态射的类型也就相应地分为关联态射、组合态射和依赖态射。当然，在不同的应用领域，构件之间可能存在更多的关系机制需要描述，这就需要开发人员自定义面向特定应用的新型态射。例如可以新定义一种 Mapping 态射，用来描述模型驱动开发中不同抽象层次的构件规范之间的精化关系。

3.3 构件组合

范畴理论中余极限（Colimit）[7]的概念可用来描述构件之间的层次组合关系，这种关系本质上是软件体系结构特征间的结构依赖关系在更高抽象层次上的反映。

定义 8（余极限构件） 给定两个构件对象 $OBJ_{CMP1}=\langle ID_{CMP1},SCP_1,TP_{CMP1}\rangle$ 和 $OBJ_{CMP2}=\langle ID_{CMP2},SCP_2,TP_{CMP2}\rangle$，其中，$SCP_1=\langle CSF_1,N_{CP1},DS_{Info1},DS_{Prd1},DS_{Req1},DS_{Act1},DS_{Bhv1},f_{AP1}\rangle$，$SCP_2=\langle CSF_2,N_{CP2},DS_{Info2},DS_{Prd2},$

DS_{Req2}，DS_{Act2}，DS_{Bhv2}，$f_{AP2}\rangle$。OBJ_{CMP1}和OBJ_{CMP2}的余极限构件表示由二者组合形成的复合构件，用$OBJ_{CMP}=\langle ID_{CMP}, SCP, TP_{CMP}\rangle$和两个组合态射$\gamma_1: OBJ_{CMP1}\rightarrow OBJ_{CMP}$与$\gamma_2: OBJ_{CMP2}\rightarrow OBJ_{CMP}$给出，其中：

(1)ID_{CMP}是组合后构件的唯一标识。

(2)$SCP=\langle CSF, N_{CP}, DS_{Info}, DS_{Prd}, DS_{Req1}, DS_{Act}, DS_{Bhv}, f_{AP}\rangle$是组合后的规范描述。

(2.1)CSF是组合后的标识特征；

(2.2)N_{CP}是组合后的名字；

(2.3)$DS_{Info}=DS_{Info1}\cup DS_{Info2}$；

(2.4)$DS_{Prd}=DS_{Prd1}\cup DS_{Prd2}-Interaction(DS_{Prd1}, DS_{Prd2})$，其中$Interaction(DS_{Prd1}, DS_{Prd2})$表示$OBJ_{CMP1}$和$OBJ_{CMP2}$之间存有交互关系的内部端口组成的集合；

(2.5)$DS_{Req}=DS_{Req1}\cup DS_{Req2}-Interaction(DS_{Prd1}, DS_{Prd2})$；

(2.6)$DS_{Act}=DS_{Act1}\cup DS_{Act2}$；

(2.7)$DS_{Bhv}=DS_{Bhv1}\parallel DS_{Bhv2}/\{e\mid e\in Inner(DS_{Act1}, DS_{Act2})\}$，其中$Inner(DS_{Act1}, DS_{Act2})$表示要去掉组合后的内部动作；

(2.8)$f_{AP}=f_{AP1}\cup f_{AP2}-\{\{e\mid e\in Inner(DS_{Act1}, DS_{Act2})\}\rightarrow DS_{Prd}\cup DS_{Req}\}$，即不包含内部动作与内部端口的操作和消息间的关联。

(3)TP_{CMP}是复合构件的类型说明。

3.4　软件体系结构模型

类型范畴[11]是针对知识的抽象描述与处理，在一般范畴理论的基础上强化了知识构造和推理的能力。在构件式软件系统的模型描述中，体系结构配置刻画了构件以及构件间的依赖、交互及架构关系。本文将构件式软件系统的体系结构模型定义为一个类型范畴。

定义 9(体系结构模型)　构件式软件体系结构模型是一个类型范畴，形式化描述为一个四元组$Model_{Arc}=\langle S_{CMP}, S_{CRL}, S_{Type}, S_{Rule}\rangle$，其中：

(1)$S_{CMP}=\{OBJ_{CMP}\mid OBJ_{CMP}=\langle ID_{CMP}, SCP, TP_{CMP}\rangle\}$是模型中构件对象的集合。

(2)S_{Type}是态射的类型的集合。

(3)$S_{CRL}=\{\gamma_n\mid \gamma_n=\langle OBJ_{CMPi}, OBJ_{CMPj}, TP_{MPS}\rangle\}$是模型中构件态射的集合，其中$OBJ_{CMPi}\in S_{CMP}$，$OBJ_{CMPj}\in S_{CMP}$，$TP_{MPS}\in S_{Type}$。

(4)$S_{Rule}: S_{Type}\times S_{Type}\rightarrow S_{Type}$是态射类型之间复合规则的集合。

(5)$u\in S_{Type}$表示单位态射类型，对S_{CMP}中的任一OBJ_{CMP}，有$\gamma_{Id_OBJ_{CMP}}=\langle OBJ_{CMP}, OBJ_{CMP}, u\rangle\in S_{CRL}$。

(6)部分映射关系集合$S_{Rule}: S_{Type}\times S_{Type}\rightarrow S_{Type}$，把任一$(r_m, r_n)\in dom(S_{Rule})$对应到类型$r_w=r_m\times r_n$，表示$r_i$和$r_j$的类型复合，并且满足下述规则：

(6.1)若$\exists\langle OBJ_{CMPi}, OBJ_{CMPj}, r_m\rangle$和$\langle OBJ_{CMPj}, OBJ_{CMPk}, r_n\rangle$，则有$(r_m, r_n)\in dom(S_{Rule})$；

(6.2)对所有$r\in S_{Type}$，有$u\times r=r\times u=r$。

(7)对$OBJ_{CMPi}, OBJ_{CMPj}, OBJ_{CMPk}\in S_{CMP}$，$r_m, r_n\in S_{Type}$，若$\exists\gamma_m=\langle OBJ_{CMPi}, OBJ_{CMPj}, r_m\rangle$和$\gamma_n=\langle OBJ_{CMPj}, OBJ_{CMPk}, r_n\rangle$，则称$\langle OBJ_{CMPi}, OBJ_{CMPj}, r_m\rangle\times\langle OBJ_{CMPj}, OBJ_{CMPk}, r_n\rangle\rightarrow\langle OBJ_{CMPi}, OBJ_{CMPk}, r_m\times r_n\rangle$为复合态射，记作$\gamma_n\circ\gamma_m$。

(8)对所有的$\gamma_m=\langle OBJ_{CMPi}, OBJ_{CMPj}, r_m\rangle$，$\gamma_n=\langle OBJ_{CMPj}, OBJ_{CMPk}, r_n\rangle$和$\gamma_k=\langle OBJ_{CMPk}, OBJ_{CMPl}, r_t\rangle$，总有$\gamma_k\circ(\gamma_n\circ\gamma_m)=\langle OBJ_{CMPi}, OBJ_{CMPl}, (r_m\times r_n)\times r_t\rangle=\langle OBJ_{CMPi}, OBJ_{CMPl}, r_m\times(r_n\times r_t)\rangle=(\gamma_k\circ\gamma_n)\circ\gamma_m$。

(9)对任一$\gamma_m=\langle OBJ_{CMPi}, OBJ_{CMPj}, r_m\rangle$，有$\gamma_m\circ\gamma_{Id_OBJ_{CMPi}}=\gamma_{Id_OBJ_{CMPj}}\circ\gamma_m=\gamma_m$成立。

3.5　模型精化和模型转换

模型驱动的软件开发是对实际问题进行建模，并转换、精化模型，直至生成可执行代码的过程。体系结构模型转换的过程伴随着体系结构特征的逐步精化。精化关系把不同层次和粒度的特征连接起来，从而形成一种层级结构。

定义 10(体系结构特征精化)　给定不同抽象层次的两个体系结构特征空间$SP_1=\langle S_{AF1}, S_{FR1}, S_{CST1}\rangle$和$SP_2=\langle S_{AF2}, S_{FR2}, S_{CST2}\rangle$，称$SP_2$是$SP_1$的精化，必须满足下列条件：

(1)存在S_{AF2}关于S_{AF1}的一个抽象函数$\Gamma: SP_2\rightarrow SP_1$，满足：

(1.1)Γ是全函数；

(1.2)$\Gamma(S_{AF2})\subseteq S_{AF1}$。

(2)$\forall AF_i, AF_j\in S_{AF2}$，若$\exists\langle AF_i, AF_j\rangle\in S_{FR2}$，则必有$\langle\Gamma(AF_i), \Gamma(AF_j)\rangle\in S_{FR1}$，并且

(2.1)若存在针对$\langle AF_i, AF_j\rangle$的前置约束条件$pre(\langle AF_i, AF_j\rangle)\in S_{CST2}$，则必有$pre(\langle\Gamma(AF_i), \Gamma(AF_j)\rangle)\in S_{CST1}$，且$pre(\langle\Gamma(AF_i), \Gamma(AF_j)\rangle)\Rightarrow(pre(\langle AF_i, AF_j\rangle))$；

(2.2)若还存在针对$\langle AF_i, AF_j\rangle$的后置约束条件$post(\langle AF_i, AF_j\rangle)\in S_{CST2}$，则必有$post(\langle\Gamma(AF_i), \Gamma(AF_j)\rangle)\in S_{CST1}$，且$pre(\langle\Gamma(AF_i), \Gamma(AF_j)\rangle)\wedge post(\langle AF_i, AF_j\rangle)\Rightarrow post(\langle\Gamma(AF_i), \Gamma(AF_j)\rangle)$。

上述条件说明：在抽象函数Γ的解释下，对于精化后的特征空间SP_2中的每一个特征和特征间的依赖关系，都能在SP_1中找到映像和依据；精化后的特征依赖的前置约束条件要比精化前的弱，后置条件要比精化前的强。

范畴理论中的函子(Functor)[7]可用来描述软件体系结构模型转换前后的映射关系，因为这里的模型是类型范畴，所以这个函子就称作类型函子[11]。

定义 11(体系结构映射函子)　给定两个由类型范畴描述的体系结构模型$Model_{Arc1}=\langle S_{CMP1}, S_{CRL1}, S_{Type1}, S_{Rule1}\rangle$和$Model_{Arc2}=\langle S_{CMP2}, S_{CRL2}, S_{Type2}, S_{Rule2}\rangle$，$Model_{Arc1}$到$Model_{Arc2}$的类型函子称为体系结构映射函子，记作$\Psi: Model_{Arc1}\rightarrow Model_{Arc2}$，定义如下。

(1)对S_{CMP1}中的每个构件对象OBJ_{CMP}，Ψ关联S_{CMP2}中的一个对象$\Psi(OBJ_{CMP})$。

(2)Ψ是从S_{Type1}到S_{Type2}的同态映射，并满足以下特性：

(2.1)对S_{Type1}中的每个类型TP_{MPS}，Ψ关联S_{Type2}中的一个类型$\Psi(TP_{MPS})$；

(2.2)对单位态射类型$u_1\in S_{Type1}$，有$\Psi(u_1)=u_2$，$u_2\in S_{Type2}$；

(2.3)$\Psi(r_m \times r_n)=\Psi(r_m)\times\Psi(r_n)$,这里 $r_m, r_n \in S_{Type1}$,$\Psi(r_m), \Psi(r_n) \in S_{Type2}$。

(3)Ψ 是从 S_{CRL1} 到 S_{CRL2} 的同态映射,并满足以下特性:

(3.1)对 S_{CRL1} 中的每个态射 $\gamma=\langle OBJ_{CMPi}, OBJ_{CMPj}, TP_{MPS}\rangle$,$\Psi$ 关联 S_{CRL2} 中的态射 $\Psi(\gamma)=\langle \Psi(OBJ_{CMPi}), \Psi(OBJ_{CMPj}), \Psi(TP_{MPS})\rangle$;

(3.2)对 S_{CMP1} 的每个 OBJ_{CMP},$\Psi(\gamma_{Id}\_OBJ_{CMP})=\gamma_{Id}\_\Psi(OBJ_{CMP})$;

(3.3)$\Psi(f\circ g)=\Psi(f)\circ\Psi(g)$,操作$\circ$代表态射合成。

类型函子描述了体系结构配置转换(或精化)前后的映射关系。在模型驱动开发过程中的每一个抽象描述层都有一个用类型范畴表示的体系结构模型,一系列的类型范畴通过体系结构映射函子连接成一个体系结构空间。从高层的抽象模型 M_0 开始,通过逐步地规范精化和配置转换,经历 $M_0 \rightarrow M_1 \rightarrow \cdots \rightarrow M_n$,得到模型的具体实现 M_n。这说明以模型转换为核心的模型驱动开发过程与迭代地对问题模式进行选择、实例化、组合与优化的过程是一致的[12]。在这种意义上,一系列的类型函子就描述了整个问题的求解过程。

4 实例研究

本节以一个飞机航班在线订票的综合服务系统作为实例来阐述所提描述框架的应用。实际的业务流程比较复杂,这里作了简化处理。

图 1 描述了这个订票系统的结构,其主要由 6 个子系统配置而成:客户端用户窗口(UserWindow)、服务中心(ServiceCenter)、机票服务系统(FlightService)、宾馆服务系统(HotelService)、车辆接驳服务系统(VechileService)、网上支付系统(Payment)。用户通过 UserWindow 部件提供的交互界面与 ServiceCenter 交互,再经 ServiceCenter 使用 FlightService,HotelService 和 VechileService 提供的各种服务。当用户确认了某一服务选项后,ServiceCenter 会请求使用构件 Payment 完成用户在线支付相应款项的操作。其中,构件 Payment 会连接外部的银行服务系统,FlightService 需要调用航空公司的数据信息服务并与之实时互动。

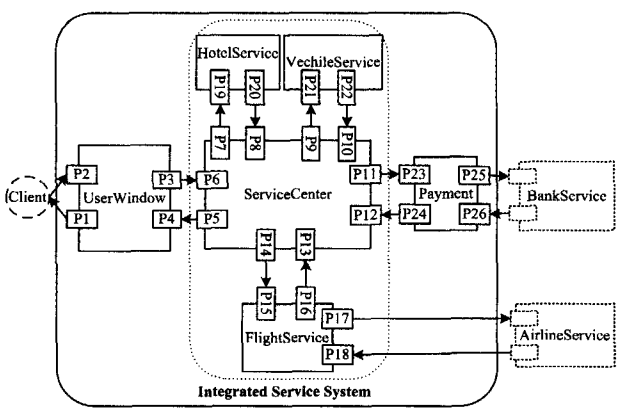

图 1 机票预订综合服务系统的体系结构

基于本文第 2 节提出的形式化描述框架,上述系统结构的类型范畴图如图 2 中第 1 层所示,其中 $c_1, c_2, \cdots$ 表示组合态射;$f_1, f_2, \cdots$ 表示交互和依赖态射。构件 Book 是依据定义 8 对 ServiceCenter,FlightService,HotelService,VechileService 进行组合操作求出的余极限构件,表示它们的复合构件。

假设经模型转换后得到的系统结构如图 2 中第 2 层的范畴图所示。这一层中 FlightService 子系统又包括 5 个构件:航班信息查询构件(FlightQuery)、机票预订处理构件(FlightBooking)、机票更改处理构件(FlightChange)、航空公司数据外联构件(AirlineContact)、送票服务(TicketDelivery)构件。宾馆服务系统(HotelService)包括宾馆信息查询(HotelQuery)和宾馆预订(HotelBooking)两个子构件。车辆接驳服务系统(VechileService)包括车辆信息查询(VechileQuery)和车辆接驳预订(VechileBooking)两个子构件。当用户选择了机票服务 FlightService 并在 UserWindow 部件提供的交互窗口中输入待查询的航班信息后,FlightService 将调用 FlightQuery 提供的对应查询子构件进行信息查询;用户浏览航班信息并做出订票动作后,FlightService 将调用 FlightBooking 完成机票预订;当用户需取消已订机票或者改签机票时,要通过 FlightChange 调用其相应的功能完成。其中,对航班信息的查询、机票预订、机票改签及退票等业务还要通过 AirlineContact 构件连接航空公司的数据管理系统才能实现信息的处理。

第 2 层的范畴图中存在态射的复合,f_6 表示构件 FlightChange 依赖 FlightQuery,f_8 表示构件 FlightQuery 依赖 AirlineContact。这种依赖关系可传递,自然有 FlightChange 依赖构件 AirlineContact,即图表上的态射 $f_7=f_8\circ f_6$。

再经过一次模型转换后,得到图 2 中第 3 层的范畴图。

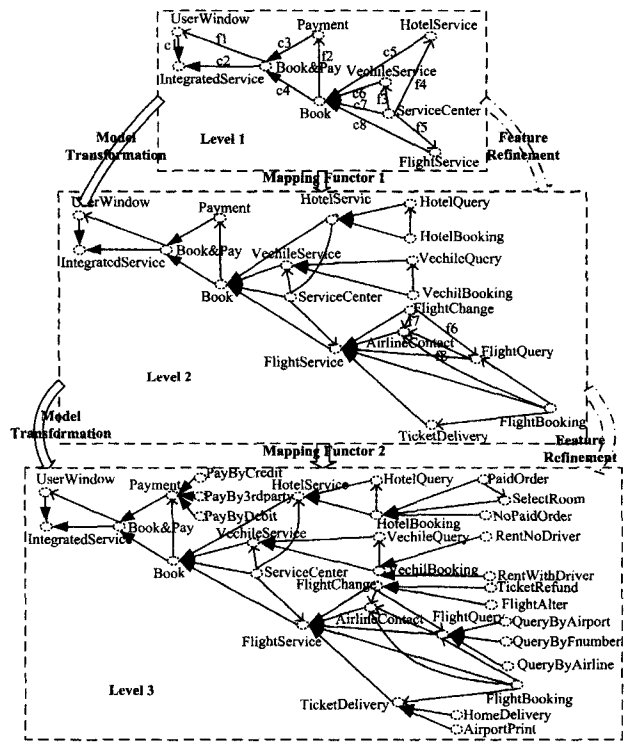

图 2 不同抽象层次间的体系结构模型映射

其中,FlightQuery 构件包括 3 个子构件:按起降机场查询(QueryByAirport)、按航班班次查询(QueryByFnumber)和

(下转第 181 页)

tions of the ACM, 2003, 46(3): 30-33.

[7] OVIATT S, DARRELL T, FLICKNER M. MultimodalInterfaces That Flex, Adapt, and Persist[J]. Communications of the ACM, 2004, 47(1): 30-33.

[8] SONG Y, DAVIS R. Continuous body and hand gesture recognition for natural human-computer interaction[C]// International Conference on Artificial Intelligence. AAAI Press, 2015.

[9] KIM K, KIM J, CHOI J, et al. Depth camera-based 3D hand gesture controls with immersive tactile feedback for natural mid-air gesture interactions[J]. Sensors, 2015, 15(1): 1022-1046.

[10] XU R, ZHOU S, LI W J. MEMS Accelerometer Based Nonspecific-User Hand Gesture Recognition[J]. IEEE Sensors Journal, 2012, 12(5): 1166-1173.

[11] ZHANG Z. A Flexible New Technique for Camera Calibration [J]. IEEE Transactions on Pattern Analysis & Machine Intelligence, 2000, 22(11): 1330-1334.

(上接第 152 页)

按航空公司查询(QueryByAirline)。送票服务(TicketDelivery)构件包括送票上门(HomeDelivery)和机场打印(AirportPrint)两个子构件。机票更改处理构件(FlightChange)包括机票退订(TicketRefund)和机票改签(FlightAlter)两个子构件。VechileBooking 包含单订车自驾租赁(RentNoDriver)和带司机租赁(RentWithDriver)两种服务构件。HotelBooking 则包括已付费订单处理(PaidOrder)、未付费订单处理(NoPaidOrder)和房间选择(SelectRoom)3 个子构件。从 PaidOrder 到 SelectRoom 的态射表明,只有付费的用户才有权限选择房间。

图 2 中,3 个抽象层次的体系结构模型构成了一个体系结构空间,每一层的模型都是符合定义 9 的一个类型范畴图,相邻模型之间的映射关系即可依据定义 11 描述为体系结构映射函子。位于图 2 左边的两个实线宽箭头表示不同抽象层间的模型转换,而右边的两个虚线宽箭头则表示模型转换的过程中伴随着体系结构特征的逐步精化。范畴理论有许多成熟的支撑工具[13],可用于语义规范的自动化组合、分析和推理。将这些工具集成到基于模型驱动的软件开发环境中,便可实现范畴论语义特性的自动化验证。限于篇幅,这些内容将在后续研究中阐述。

结束语　本文将类型范畴理论和特征建模等概念有机地结合在一起,为构件式软件模型的描述和转换建立了一种统一的语义描述框架。该框架遵循模型驱动的软件开发理念和实质要求,为研究模型驱动开发和模型转换提供了新的理解、认知和学习的指导架构。下一步的研究工作将基于本文的框架,建立常规模型转换的语义计算理论,对转换前后模型描述之间应满足的语义特性和约束进行深入的分析和探讨,最终实现语义特性的自动化验证。

参 考 文 献

[1] MAO X J, DONG M G, QI Z C, et al. Running mechanism and implementation technique of self-adaptive software in open environment [J]. Chinese Journal of Computers, 2015, 38(9): 1893-1906. (in Chinese)
毛新军,董孟高,齐治昌,等. 开放环境下自适应软件系统的运行机制与构造技术[J]. 计算机学报,2015,38(9):1893-1906.

[2] HE X, MA Z Y, WANG R C, et al. Semantics-Configurable model transformation[J]. Journal of Software, 2013, 24(7): 1436-1454. (in Chinese)
何啸,麻志毅,王瑞超,等. 语义可配置的模型转换[J]. 软件学报,2013,24(7):1436-1454.

[3] MENS T. A survey of software refactoring [J]. IEEE Trans. on Software Engineering, 2004, 30(2): 126-139.

[4] MENS T, VAN EETVELDE N, Demeyer S, et al. Formalizing refactorings with graph transformations[J]. Journal of Software Maintenance and Evolution: Research and Practice, 2005, 17(4): 247-276.

[5] LIU H, MA Z Y, SHAO W Z. Description and Proof of Property Preservation of Model Transformations [J]. Journal of Software, 2007, 18(10): 2369-2379. (in Chinese)
刘辉,麻志毅,邵维忠. 模型转换中的特性保持的描述与验证[J]. 软件学报,2007,18(10):2369-2379.

[6] NATHAN W, RUZANNA C, AWAIS R. Formal semantic conflict dection in aspect oriented requirements [J]. Requirements Engineering, 2009, 14(4): 247-268.

[7] MICHAEL B, CHARIES W. Category theory for computing science [M]. New Jersey: Prentice-Hall, 1990.

[8] HOU J K, WAN J C, YANG X, et al. Formal semantics of component-based architecture model mapping [J]. Journal of Computer Reaearch and Development, 2009, 46(2): 310-320. (in Chinese)
侯金奎,万建成,杨潇,等. 构件式体系结构模型映射的形式化语义[J]. 计算机研究与发展,2009,46(2):310-320.

[9] WANG Z J, XU X F, ZHAN D C. Feature-Based component model and normalized design process [J]. Journal of Software, 2006, 17(1): 39-47. (in Chinese)
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.

[10] JIA Y, GU Y Q. Domain feature space based semantic representation of component [J]. Journal of Software, 2002, 13(2): 311-316. (in Chinese)
贾育,顾毓清. 基于领域特征空间的构件语义表示方法[J]. 软件学报,2002,13(2):311-316.

[11] LU R Q. Towards a mathematical theory of knowledge[J]. Journal of Computer Science and Technology, 2005, 20(6): 751-757.

[12] WANG C J, LUO H M, ZUO Z K. Formal software specification generation approach based on problem patterns[J]. Journal of Computer Reaearch and Development, 2013, 50(2): 352-360. (in Chinese)
王昌晶,罗海梅,左正康,等. 基于问题模式的形式化软件规格说明生成方法[J]. 计算机研究与发展,2013,50(2):352-360.

[13] SRINIVAS Y, JULLIG R. SPECWARETM: Formal support for composing software: Technical Report KES. U. 94. 5[R]. California: Kestrel Institute, 1994: 22-39.