模型驱动开发中模型演化语法和语义特性研究

孙为军[1,2] 李师贤[1] 严玉清[1,3]
(中山大学计算机科学系 广州 510275)[1] (广东工业大学计算机学院 广州 510006)[2]
(广东外语外贸大学思科信息学院 广州 510006)[3]

摘 要 模型演化由一系列复杂的变化活动组成,要遵循一定的约束以保持模型的某些特性。以一个实例描述模型演化的过程,并以集值映射为基础,定义模型成分与语义域的映射,通过定义模型演化的语义函数,研究模型演化的语法和语义性质,包括特性保持、一致性、等价性和吸收性等。

关键词 模型驱动体系结构,模型演化,特性保持,一致性
中图法分类号 TP311.5 **文献标识码** A

Study on Syntax and Semantics Properties of Model Evolution in Model Driven Development

SUN Wei-jun[1,2] LI Shi-xian[1] YAN Yu-qing[1,3]
(Department of Computer Science, Sun Yat-sen University, Guangzhou 510275, China)[1]
(Faculty of Computer, Guangdong University of Technology, Guangzhou 510006, China)[2]
(Cisco School of Informatics, Guangdong University of Foreign Studies, Guangzhou 510006, China)[3]

Abstract Model evolution involves a series of complex change activities and should follow certain constraints to preserve certain properties of models. An illustration of model evolution was showed. Based on the set-valued mapping, the mapping from model elements to semantics domain was defined. Syntax and semantics properties of model evolution such as property preserving, consistency, absorbing and equivalence were studied by defining the semantic function.

Keywords Model driven architecture, Model evolution, Property preservation, Consistency

1 引言

构造性和演化性是软件的两个基本特征[1]。随着运行环境以及需求的不断变化,软件(模型)也需要不停地演化以满足新的需求[2]。软件演化是软件生命周期中的重要组成部分,由一系列复杂的变化活动组成。软件在生命周期的各个阶段,都可能发生演化。模型驱动体系结构(Model Driven Architecture, MDA)是以模型为中心,将业务和应用逻辑与底层平台技术分离开来的系统开发方法。MDA对业务逻辑建立抽象模型,通过模型转换和逐步精化,自动生成最终的软件产品。模型驱动体系结构降低了用户需求变化对实现技术的影响,某个模型发生改变时,这些变化信息可以被自动地演化,为软件演化提供了一个较为理想的解决方案。模型驱动体系结构能为软件系统的演化、集成、互操作性、重用性提供支持。

需求变化是导致软件演化的根本原因,MDA以模型为中心,因此需求变化导致模型发生相应的变化。元模型或者建模语言的变化,可能导致模型的结构、类型和约束无效,模型也发生相应的变化。在模型驱动开发中,模型的变化被描述为对模型的一系列模型变化操作,模型的演化过程就是将一系列的模型操作应用到模型上的过程。模型演化操作也分为基本的操作和复杂的操作。模型演化需要遵循一定的约束规则,以保持模型的某些特性(Property)[2]。模型演化不需要重新构造一个全新的模型,但是为了能与原有软件相兼容,模型演化通常要求保持原模型的某些特性,比如外部行为特性、接口、性能等。这种类型的约束主要用于保证不会破坏原模型的某些特性,一般称为特性保持(Property preservation)约束[2]。

模型演化是模型驱动软件演化的一项关键技术。文献[3,4]对需求演化进行了定性研究,主要采用逻辑方法建立逻辑意义上的需求演化模型。文献[5]总结了元模型演化的研究热点。文献[6]设计了一种基于图转换的可视化的模型转换语言,专门用于模型演化,其通过工具将旧模型转换为语义等价的新模型,以保证新旧模型的一致性。文献[7]给出了7种基本演化操作,在基本演化操作基础上研究了模型差异、合并和组合的通用算法。文献[8]在模型合并算法中引入了语法和语义两个方面,以进行更精确的冲突检测并且确定冲突的原因,从而即使存在冲突也能为合并进程提供更好的支持。文献[2]提出了一种基于图转换的特性保持约束机制,其将模型演化中转换规则以及特性保持约束都描述为图转换规则,并给出了转换规则是否满足特性保持约束的算法。上述文献着重于具体模型演化技术的研究,缺乏模型演化的语法和语

到稿日期:2011-09-26 返修日期:2012-01-29 本文受国家自然科学基金(60774095),广东省自然科学基金(9451009001002777)资助。
孙为军(1975—),男,博士生,讲师,主要研究方向为软件工程、模型驱动体系结构、软件演化,E-mail:gdutswj@gdut.edu.cn;李师贤(1944—),男,教授,博士生导师,主要研究方向为软件工程、形式语义学;严玉清(1963—),女,博士生,副教授,主要研究方向为软件工程、需求管理。

义性质研究。

从不同层次研究模型演化的机制，特别是研究模型演化的语法和语义，对工具支持的模型演化过程有重要的意义。与研究具体的模型演化技术不同，本文研究模型演化的语法和语义性质，为更广泛的模型演化研究做了准备工作。

2 模型的语法和语义

2.1 建模语言的语法和语义

在软件工程中，我们常用图形或文本的建模语言建立模型去描述系统的结构、行为和交互等，这些建模语言种类很多，但它们的定义都涉及语言的抽象语法、具体语法和语义。

定义 1(建模语言) 一个建模语言可以用五元组 $L=(A,C,D,M_C,M_D)$来表示[9]，其中：

(1) A 为一个集合，表示语言的抽象语法；

(2) C 为一个集合，表示语言的具体语法；

(3) D 为一个集合，表示语言的语义域；

(4) M_C 为一个函数 $M_C:C\rightarrow A$，表示语言的语法映射；

(5) M_D 为一个函数 $M_D:A\rightarrow D$，表示语言的语义映射。

建模语言 L 的语法由抽象语法 A、具体语法 C 和语法映射 M_C 定义。抽象语法 A 描述该语言能够提供的概念集合、概念之间的关系以及语法规则。语法规则用来检查概念在进行组合时是否满足约束条件。模型语言需要定义一个良好形式(well-formed)的抽象语法模型，用于判断采用这个语言书写的模型是否合法。抽象语法具体语法、语义等的基础，而系统分析员或建模人员描述模型时使用的是具体语法。具体语法是具体表达模型的方式，它们是抽象语法的不同视图。具体语法通常分为两种：文本表示语法和图形表示语法。建模语言的语法映射定义了抽象语法概念和具体语法元素的对应关系。Java 语言的语法文本表示语法的例子，UML 各种图则是图形表示语法的例子。

建模语言 L 的语义由语义域 D 和语义映射 M_D 定义。给定一个模型语言 M，每个元素的含义通常由众所周知的语义域 D 来定义。语义域的详细信息和表示形式有很多种。例如，一个操作语义通过构造一个抽象机来描述用一个语言书写的模型或代码如何能被执行[10]。指称语义是指把语言的每个概念映射到数、函数或者元组这样的数学对象，这些概念映射的数学对象构成语言概念对应的语义域。这些数学对象被称为相应语言的概念的指称(denotation)。其他的语义域的例子有系统模型、抽象状态机或纯数学等。

假设 M 为符合建模语言 L 的良好定义的模型集合，则对任意一个模型 $m\in M$，m 在语法上是良好形式的。每个模型通过从建模语言到语义域的映射得到它的一个语义。对编程语言来说，每一个语法成分都有一个良好定义的含义，这个含义描述了它在操作语义或指称语义方面的作用。模型的含义(语义)的形式化的映射函数的好处是，能够对映射、语言和实例本身进行推理。

2.2 集值语义

集值分析是一个现代数学分支[11]。作为建立非线性问题的数学模型，解决非线性问题的数学理论和有力工具，它已经成为非线性分析的重要组成部分，在控制论、决策论、非线性最优化、拓扑学等众多邻域内有着广泛的应用。本文借鉴文献[12]的研究方法，基于集值分析和代数理论，研究模型演化的语法和语义性质。

定义 2(集值映射，Set-valued Mapping) 设 X,Y 为两个 Hausdorff 拓扑空间，2^Y 表示 Y 中所有非空子集构成的族。映射 $F:X\rightarrow 2^Y$ 使得 $\forall x\in X,F(x)$ 是 Y 中的一个非空子集，则称 F 为点到集的映射或从 X 到 Y 的一个集值映射[11]。

集值映射实际上是单值映射在概念上的推广，将单值映射的概念推广到集值映射中。

在模型驱动开发的早期阶段建立的模型通常是非精确的，并且在某种程度上是抽象的。相应地，通常不是一个单一的系统而是一组系统来实现这个模型。因此，从一个非精确定义的模型到具体程序代码的转换将导致代码不完整或者代码与模型不相符合。为了充分表达精确语义，我们以集值映射为基础，定义集值语义，语言的语义被定义为满足给定特性的所有系统的集合，而不是一个单一的系统。基本思路是将任何模型 $m\in M$ 映射到服从该模型表示的约束的所有系统。

定义 3(集值语义映射，Set-valued Semantics Mapping) 对一个模型 $m\in M$，其集值语义映射用模型 M 到语义域 D 的一个函数来表示[12]：

$$sm:M\rightarrow D$$

式中，M 表示良好定义的模型的集合，S 表示所有系统的集合，语义域 D 就是 S 的幂集 $D=\wp(S)$。

这个定义说明每个模型实例 $m\in M$ 都将要通过 sm 被映射到满足模型约束的最大的系统实现集合。这里并不需要进一步研究 S 的细节，但要明确 S 捕获一个系统的相关特性。这些通常是结构特性(如对象的值和对象之间的关系)以及行为和交互特性(如对交互的跟踪等)。

考虑一个简单的类图，类图有一个类 Event，这个类中有一个 String 类型的属性 date。那么类图描述的系统满足以下约束：一个类 Event；所有的类 Event 的实例有一个属性 date，其类型为 String；类 Event 的所有子类的实例有一个属性 name，其类型为 String；除此之外没有更多的可用信息。

这个简单模型的真正语义被所有服从上述的约束的系统指定了。通常，这些系统可能包含其他的类，类 Event 除了 date 属性，可能还包含更多的其他属性，但在我们的集值语义中，这些系统仍然满足模型中定义的约束。此外，并没有规定系统一定会有类 Event 的一个实例。相反，类 Event 也可以是抽象类。这种捕获非精确的定义的方法被称为“松散语义”[12]。

集值语义映射具有下列一些重要性质：

(1)对一个模型 $m\in M$，如果 $sm(m)\neq\emptyset$，则 m 是一致的。

$sm(m)\neq\emptyset$ 意味着至少有一个系统满足模型实例的特性。否则，模型 m 本身存在一些相互矛盾的约束。

(2)对一个模型 $m\in M$，如果 $sm(m)=S$，则 m 不包含任何信息。

$sm(m)=S$ 意味着任何一个系统可以作为一个实现。

(3)对两个模型 $m_1\in M,m_2\in M$，如果 $sm(m_2)\subseteq\in sm(m_1)$，则 m_2 是对 m_1 的精化[12]。

通过给模型 m_1 增加信息，得到模型 m_2，模型 m_2 比 m_1

包含了更多的信息，相当于对模型 m_1 做了进一步的约束，因此满足新的约束的系统集合可能变得更小。这意味着需求越明确，模型表示的信息就越多，满足模型约束的系统实现就越少。

3 模型演化

MDA 中，“一切都是模型”，为了描述模型的变化，我们给定两个模型 M_1 和 M_2。M_1 为变化前的模型，即源模型；而 M_2 则是变化后的模型。M_1 经过一定的变化，得到 M_2，我们将这些变化称为变化模型，记为 $M_2-M_1=\Delta M$。

本节首先给出一个模型演化的实例作为启发实例，如图 1 所示。这个简单例子将作为本节后面进一步解释的基础。

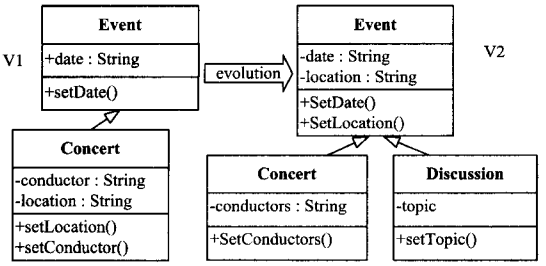

图 1 模型演化实例

UML 类图用来显示系统中各个类的静态结构和关系。图 1 显示了描述系统静态结构的 UML 类图的演化过程。左边的类图是演化的源模型，右边的图是演化目标模型。演化的源模型由两个类组成：Event 类代表事件，事件类中包含一个属性 date 和一个操作 setDate，属性 date 表示事件发生的日期。Concert 类是 Event 类的子类，代表音乐会事件。目标模型由 3 个类组成，其中包含一个类 Event 的子类 Discussion，其代表讨论事件。类 Concert 和类 Discussion 都有自己特有的属性和操作。

在图 1 中，右边的类图代表的目标模型是左边的类图代表的源模型经过一定的演化操作步骤得到的。观察演化前后的两个类图，可以发现演化操作步骤中包含下列的变化：

(1)类 Discussion 被添加，该类具有属性 topic 和操作 setTopic，类 Discussion 的父类是 Event；

(2)类 Event 的属性 date 的可见性被修改，由 public 改为 private；

(3)类 Concert 的属性 conductor 更改为 conductors，操作 setCondctor 更改为 setConductors；

(4)属性 Location 和操作 setLocation 从类 Concert 移动到了类 Event。

显然，源模型经过演化发生了变化，演化后的目标模型中，一部分信息与源模型中信息相同，一部分信息是由源模型中的信息经过修改得到的，一部分信息是新增加的，还有一些信息被删除。

模型的变化可以分为基本变化和组合变化：基本变化是可以直接施加在模型元素上的变化，对于每个模型元素可以施加的基本变化有增加、删除和修改等，如实例中属性和操作的增加、删除和修改，输入基本变化；组合变化是基本变化的组合，能够分解成基本变化，如实例中属性和操作的移动是删除和增加的组合。

相应地，模型演化操作也分为基本的演化操作和复杂的演化操作。可以将变化模型 ΔM 表示为引起模型变化的演化操作序列，这些演化操作包括增加、删除或更改模型中的一个或多个元素。将模型差异 Δ 应用于 M_1，就是按照一定的先后顺序执行 ΔM 中的演化操作，从而得到 M_2。

4 模型演化的数学表示

4.1 模型演化算子

最简单的模型演化形式，是指将一种基本演化操作作用在模型上，生成一个新的模型。将模型用 M 表示，模型演化操作序列用 ΔM 表示，我们得到了模型演化算子的如下定义。

定义 4(模型演化算子)

模型演化算子是一个函数，具有两个输入，一个输入是模型 M，一个输入是模型操作序列，它产生了一个演化模型作为输出：

$\otimes: M \times \Delta M \rightarrow M$

定义 5(特性保持 PP，Property Preserving)

一个演化算子：$\otimes: M \times \Delta M \rightarrow M$ 是特性保持的，对 $\forall m \in M$，它满足 $sm(m \otimes \Delta m) \subseteq sm(m)$。

特性保持对模型演化很重要，在模型演化中，大部分的模型演化约束都是特性保持约束。特性保持确保不会破坏原模型的某些特性，特别是系统设计信息不能在模型演化中丢失。特性保持的例子如图 2 所示。

图 2 特性保持的例子

推论 1 特性保持等价于：

$\forall m \in M: sm(m \otimes \Delta m) \subseteq sm(m) \cap sm(m \otimes \Delta m)$

请注意，左边和右边不一定是相等的，因为演化算子可能允许添加在模型 m 中没有出现的信息。

定义 6(全特性保持 FPP，Fully Property Preserving)

一个演化算子 $\otimes: M \times \Delta M \rightarrow M$ 是全特性保持，当且仅当

$\forall m \in M: sm(m \otimes \Delta m) = sm(m) \cap sm(m \otimes \Delta m)$

FPP 对模型非常重要。首先，FPP 演化使模型驱动开发中分析和理解演化前的模型及其特性成为可能，也可以从演化后的模型出发，通过对特性的跟踪，回溯到演化前的模型。其次，通过 FPP 演化，开发人员可以确保模型中定义的特性经过演化后仍然保持在系统实现中。最后，一个 FPP 演化可以减少模型演化的变化影响：演化对一个输入模型的改变是局部的，不会影响在其他模型中的特性定义。

FPP 的演化既不增加也不丢失信息。然而，不是所有的模型演化都支持这个特性。在这种情况下，演化后出现的特性不一定能回溯到原始模型，而是来自演化运算本身，比如模型演化的求精。通过演化运算增加错误的信息可能产生不一致的输出模型 ($sm(m \otimes \Delta m) = \emptyset$)，即使演化前的模型是一致的。因此，需要演化保持一致性。

下面给出模型演化的一致性定义。

定义 7(一致性保持 CP,Consistency Preserving)　一个演化算子$\otimes: M\times\Delta M\to M$是一致性保持的,当且仅当

$\forall m\in M: sm(m)\cap sm(m\otimes\Delta m)\neq\emptyset\Rightarrow sm(m\otimes\Delta m)\neq\emptyset$

推论 2　一个 FPP 演化算子的一致性保持。

证明:根据 FPP 的定义,就可以得到这个结论。

一般情况下,本文后继部分的讨论,就基于这样的假定,特性保持的演化也是一致性保持的(并非都是完全特性保持)。

基于集值语义的演化算子可以推广到语义域 D 中,假设在语义域中的演化算子$\oplus$是可用的,上文使用的交$\cap$就是一个语义域的算子。

定义 8(一般语义演化算子)

$\oplus$演化算子的语义是一个函数,这个函数以两个系统实现的集合作为输入,产生一个系统集合作为输出:

$\oplus: D\times D\to D$

这是两个不同层次的算子,语义演化算子$\oplus$可理解为语法演化算子的语义。

4.2　演化的语法性质

根据模型语言的具体语法,结合演化算子$\otimes$的特点,我们总结模型演化可能存在一些演化操作,Δm 具有吸收性或者中立性。比如一个空的操作序列,对任何模型处理后得到的模型都是相同的,或者在演化操作序列中存在删除某个模型元素,然后又添加该软件成分,则这个操作序列作用在任何模型上,也不会改变模型语法的表现形式。

定义 9(演化语法吸收性)

一个演化操作序列 Δm 对模型 m 是中立的,当且仅当

$\forall m\in M: m\otimes\Delta m=m$

定理 1(演化的语法等价)

演化操作 Δm_1 和 Δm_2 是语法等价的,记为 $\Delta m_1\triangleq m_2$ 当 $\forall m\in M: m\otimes\Delta m_1=m\otimes\Delta m_2$。

演化的语法等价性要求非常严格,事实上,模型有一个具体的语法,当模型经过演化或者修改,模型中的空白空间和图形元素的位置经常改变,这时元素的表现形式不影响语义,但是演化后布局发生了变化,如图 3 所示。

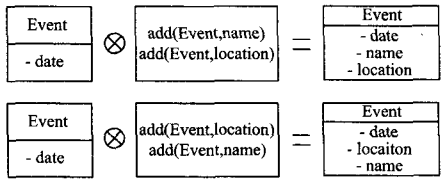

图 3　演化算子与操作序列顺序有关(语法层次)

首先,演化结果语法上依赖操作序列的顺序,因此演化操作在语法上不能互换,其次,演化结果语义上不依赖操作序列的顺序,因为两次演化的输出模型是语义等价的,这意味着模型通过 sm 映射到同样的系统集合。所以,很自然地,我们将模型演化的语法概念推广到语义概念上。

4.3　演化的语义性质

模型语义特征与模型语言的具体语法无关,而是关注模型的语义(含义)。类似于模型演化的语法特征的定义,我们给出下列定义。

定义 10(演化语义吸收性)

一个演化操作序列 Δm 对模型 m 在语义上是吸收的或者中立的,当且仅当

$\forall m\in M: sm(m\otimes\Delta m)=sm(m)$

定理 2(演化的语义等价)

演化操作 Δm_1 和 Δm_2 是语义等价的,记为 $\Delta m_1\cong\Delta m_2$ 当 $\forall m\in M: sm(m\otimes\Delta m_1)=sm(m\otimes\Delta m_2)$。

虽然不同的演化结果模型表示不同,但是语义是相同的。可以根据定理 2 来判断两个演化操作的语义是否等价。该方法为分析、优化模型演化操作描述提供了一种新的手段。复杂的组合演化操作经过优化后,可以被更简单的基本演化操作代替。

定义 11(演化语义交换性)

演化操作 Δm_1 和 Δm_2 在语义上是互交换的,当且仅当

$\forall m\in M: sm(m\otimes\Delta m_1\otimes\Delta m_2)=sm(m\otimes\Delta m_2\otimes\Delta m_1)$

这意味着,虽然对模型的连续两次演化的操作顺序不同,但是演化后的模型具有相同的含义。

4.4　语义映射的性质

两个模型的语义等价是一个模型演化操作的一个重要的性质,模型语义与模型语言的具体语法无关,所以模型的图形表示的布局变化并不影响模型的演化结果。

定义 12(模型等价类)

语义映射 sm 定义了模型之间的等价关系,即 $m_1\equiv m_2\Leftrightarrow sm(m_1)=sm(m_2)$

语义等价模型的集合指称为:

$[m_1]=\{m_2\mid m_1\equiv m_2\}$

定义 13(模型等价类上的演化操作)

将演化操作作用到模型等价类上,则有:

$[m]\otimes\Delta m=\{m\otimes\Delta m\mid m\in[m]\}$

理想情况下,对一个模型演化算子,我们希望满足下列条件[12]:

(1)演化是完全特性保持的,这样演化后的模型的每个特性都可以追踪到演化前的模型中。

(2)演化是语义一致的,这样演化与具体的语法表示无关。

(3)语义等价的演化,与演化的顺序无关。

然而,因为一些原因,很多演化操作并不满足上述理想的演化系统,例如很多演化算子是特性保持和一致性保持的,但并不是完全特性保持的,演化前后的模型并不相等。模型演化如果依赖于演化操作的顺序或者模型的具体语法表示,将使演化过程变得难以管理。例如,演化过程中不得不保存演化操作的顺序以保证模型演化的质量。

结束语　模型演化要遵循一定的约束规则以保持模型的某些特性。本文以集值映射为基础,定义模型成分与语义域的映射,通过定义模型演化的语义函数,研究模型演化的语法和语义性质,特别是模型驱动演化的特性保持和一致性保持。

本文的研究工作,只是在模型演化的语法和语义性质方面的初步探索和实践,无论是理论研究还是应用方面都还远未成熟。进一步的工作是对模型演化的形式化描述和语义的

(下转第 143 页)

结束语 本文提出了一种基于相关反馈的信息检索模型，分析了查询词分解，推导了相关反馈机制和正规化过程，并进一步阐述了文档提取方法。

提出的模型通过相关反馈和查询词扩展，能克服传统方法无法计算文档与查询词之间的相似度的缺点。通过对比分析和性能评价结果表明，本文所提方法是切实有效的。

参 考 文 献

[1] Zhang S D, Chen Y. Research on domain ontology-based intelligent information retrieval system [J]. Key Engineering Materials, 2011, 460-461: 300-304

[2] Hong W-S, Chen S-J, Wang Li-hui, et al. A new approach for fuzzy information retrieval based on weighted power-mean averaging operators[J]. Computers and Mathema-tics with Applications, 2007, 53(12): 1800-1819

[3] Chen S-J, Chen S-M. Fuzzy information retrieval based on geometric-mean averaging operators[J]. Computers and Mathematics with Applications, 2005, 49(7/8): 1213-1231

[4] Tombros A, Sanderson M. Advantages of Query Biased Summaries in Information Retrieval [C] // Proceeding of ACM-SIGIR98. 1998: 2-10

[5] Lee J H, Kim M H, Lee Y J. Information Retrieval Based on Conceptual Distance in Is-a Hierarchies [J]. Journal of Documentation, 1993, 49(2): 188-207

[6] 迟呈英，战学刚，姚天顺. 基于 p 范式模型的检索[J]. 中文信息学报，2000，14(4)：35-41

[7] He D Q. A study of self-organizing map in interactive relevance feedback [C] // Proceedings of the 3rd International Conference on Information Technology: New Generations. Las Vegas: IEEE, 2006: 394-401

[8] Paredes R, Deselaers T, Vidal E. A probabilistic model for user relevance feedback on image retrieval [C] // Proceedings of 5th International Workshop on Machine Learning for Multimodal Interaction. Utrecht: Springer, 2008: 260-271

[9] Xu Y, Jones G, Wang B. Query dependent pseudo-relevance feedback based on wikipedia [C] // Proceedings of the 32nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. Boston: ACM, 2009: 59-66

[10] Borji A, Jahromi M Z. Evolving weighting functions for query expansion based on relevance feedback [C] // Proceedings of the 10th Asia-Pacific Web Conference on Progress in WWW Research and Development. Shenyang: Springer, 2008: 233-238

[11] Vitsentiy V. A user interface of relevance feedback for interactive information retrieval systems [C] // IEEE Workshop on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications. Dortmund: IEEE, 2007: 449-453

[12] Chandramouli K, Kliegr T, Nemrava J, et al. Query refinement and user relevance feedback for contextualized image retrieval [C] // Proceedings of 5th International Conference on Visual Information Engineering. Xi'an: IEEE, 2008: 453-458

[13] Lv Y H, Zhai C X. Positional relevance model for pseudo-relevance feedback [C] // Proceedings of the 33rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. Geneva: ACM, 2010: 579-586

[14] 黄名选，严小卫，张师超. 基于矩阵加权关联规则挖掘的伪相关反馈查询扩展[J]. 软件学报，2009，20(7)：1854-1865

[15] Pu Q, He D Q. Pseudo relevance feedback using semantic clustering in relevance language model [C] // Proceedings of the 18th ACM Conference on Information and Knowledge Management. HongKong: ACM, 2009: 1931-1934

(上接第 126 页)

深入研究。从应用实际出发对模型变化描述进行全面分析和归纳，全面研究模型演化所应遵循的约束，从而提高模型变化的语义表述能力、特性保持和一致性判定能力，结合模型驱动开发工具，使其能更加准确、有效地反映并支持工程化的模型驱动开发。

参 考 文 献

[1] 钟林辉. 构件化软件开发中演化信息的获取和应用技术研究[D]. 北京：北京大学，2007

[2] 刘辉，麻志毅，邵维忠. 模型转换中特性保持的描述与验证[J]. 软件学报，2007，18(10)：2369-2379

[3] Yuan W, Ying S. Modeling requirements evolution with π-calculus[C] // 2nd Conference on Power Electronics and Intelligent Transportation System (PEITS). Shenzhen, China, IEEE Computer Society, 2009: 355-358

[4] Lormans M. Monitoring requirements evolution using views[C] // 11th European Conference on Software Maintenance and Reengineering (CSMR). Amsterdam, Netherlands: IEEE Computer Society, 2007: 349-352

[5] 刘辉，麻志毅，邵维忠，等. 元建模技术研究进展[J]. 软件学报，2008，19(6)：11

[6] Sprinkle J, Karsai G G. A domain-specific visual language for domain model evolution[J]. Journal of Visual Languages and Computing, 2004, 15(3/4): 291-307

[7] Alanen M, Porres I. Difference and Union of Models[C] // Lecture Notes in Computer Science. Springer, 2003

[8] Altmanninger K. Models in Conflict-Towards a Semantically Enhanced Version Control System for Models [C] // Lecture Notes in Computer Science. 2008: 293-304

[9] Chen K, Sztipanovits J, Abdelwalhed S, et al. Semantic anchoring with model transformations [C] // Proceedings of the 3rd European conference on Model driven architecture-foundations and applications (ECMDA-FA05). Springer, 2005, 3748: 115-129

[10] Scheidgen M, Fischer J. Human comprehensible and machine processable specifications of operational semantics [C] // Proceedings of the 3rd European conference on Model driven architecture-foundations and applications (ECMDA-FA07). Springer, 2007, 4530: 157-171

[11] 李雷，吴从. 集值分析[M]. 北京：科学出版社，2003

[12] Herrmann C, Krahn H, Rumpe B, et al. An algebraic view on the semantics of model composition[C] // Lecture Notes in Computer Science. Springer, 2007, 4530: 99-113