一种提高元建模语义完整性的方法

刘俊莉
(华南师范大学 南海校区 计算机工程系,广东 佛山 528225)

摘 要:元建模发生在模型驱动架构中的元元模型、元模型和模型层中,传统的元建模采用统一建模语言UML描述。但是实践证明,UML无法提供与对象有关的所有信息,缺少描述模型中关于对象的附加约束,而且无法描述不同模型之间的转换。针对传统元建模中的语义缺陷,特引入对象约束语言来提高元建模的精确性。文中阐述了如何结合UML和OCL应用于元模型,提高元建模语义完整性,加强元建模的可读性和可执行性,并且使用该方法能够检测出约束冲突。

关键词:元建模;模型驱动架构;统一建模语言;对象约束语言;语义完整性
中图分类号:TP311.5 **文献标识码**:A **文章编号**:1673-629X(2009)12-0040-04

New Approach to Improve Semantics Integrity of Metamodeling

LIU Jun-li
(Dept. of Computer Eng., Nanhai Campus of South China Normal University, Foshan 528225, China)

Abstract: Metamodeling is executed in the metametamodel, metamodel and model layer of model driven architecture. And unified modeling language is used to describe traditional of metamodeling. However, a UML diagram is typically not refined enough to provide all the relevant aspects of a specification, and a need to describe additional constraints about the objects in the model. And a UML diagram is typically not refined enough to express the convert of different model too. In order to improve semantics integrity, object constraint language is introduced. States how to combine UML with OCL to improve semantics integrity and to improve the readable and the executive of metamodeling, and the use of the method can detect the conflict constraints.

Key words: metamodeling; MDA; UML; OCL; semantics integrity

0 引 言

传统的建模行为发生在MDA框架的模型M1层,即对现实对象的建模,元建模[1]行为发生在元元模型M3层、元模型M2层、模型M1层。各层的元模型由通用的面向对象的对象软件建模语言统一建模语言(UML)语言来描述。虽然UML有8种图形建模语言[2],但是由实践经验得知,UML也不足以充分描述建模对象的特征和需求,UML缺少精确的语义,通常无法提供与对象有关的所有相关部分。这其中就缺少描述模型中关于对象的附加约束。约束为元模型提供新的元素以及可以附加在已有元模型元素上的各种需求或语义约束信息。在实践中,这些约束常常用自然语言描述[3],这样做经常造成歧义,使得UML模型不能进一步地分析和验证。为了精确地表达模型中的约束,引入了对象约束语言(OCL)。OCL是一种具有精确数学定义基础的形式化规范语言,它具有精确定义的语义模型、自动化的验证工具。它在元建模中的作用是:约束UML图形中的元素,使模型更加精确,从而由元建模这种机制定义的语言更具备完整性。

为了充分实现这个目的,需要做:

(1)建立元建模和语义完整性的概念;

(2)分析用UML建模的缺陷之处;

(3)提高元建模语义完整性的方法应该具备的特点;

(4)提高元建模语义完整的方法。

1 元建模和语义完整性的概述

1.1 元建模的概述

MDA框架的主要元素:模型、PIM、PSM、语言、变换、变换定义以及变换工具。元建模是一种创建元模型的活动,它和一般的建模很相似,都是对特定信息和对象建立模型,但是它们是针对不同特性的建模。所以研究元建模中语义完整性,相当于使模型语义具备完整性。元建模在MDA环境中是一种定义语言的机制[4,5]。OMG对MDA使用了4层构架(见表1)。元

收稿日期:2009-03-09;**修回日期**:2009-06-25
基金项目:2008年度广东省自然科学基金(8151063101000040)
作者简介:刘俊莉(1981-),女,助教,研究方向为元建模、本体等。

建模发生在三个元层中(M3、M2、M1)，定义各元层语言，同时，为了源模型转化为目的模型，元建模特定义相邻元层模型之间的变换定义语言[5]。

表 1　四层元建模体系结构

层	说明	例子
元元模型 M3 (Metametamodel)	元建模体系结构的基础构造。定义了描述元模型的语言	元类、元属性、元操作
元模型 M2 (Metamodel)	元元模型的实例。定义了描述模型的语言	类、属性、操作、构件
模型 M1 (Model)	元模型的实例。定义了描述信息论域的语言	StockShare, askPrice, sellLimitOrder
用户对象 M0 (User object)	模型的实例。定义了一个特定的信息论域	654.56, sell-limit-order

1.2　语义完整性的定义

元建模和建模活动一样，元建模同样包括用户层(具体语法)、逻辑层(抽象语法)、数据层(执行化，导出，转换)、语义(操作语义或约束语义)。

具体语法(Concrete Syntax)是一种标记，一种符号(Notation)。其定义是：这种语言(Language)表现在屏幕上或者纸张上的形式(形状，连接，文本注释)，这些可以被用户看到并且被用户操作。

抽象语法(Abstract Syntax)是一种语法(Grammar)，其定义是：为具体语法中的符号所表达的含义命名，并且使这个名字有效。

语义(Semantics)是定义语言(Language)的有效表达式的含义(Meaning)。明确表达式含义的方法是：把这个表达式映射到其他一个或者多个已知含义的结构(Structure)上去，这个映射(Mapping)和返回值(Result)的含义都必须被精确定义。

语义完整性是用语言描述的模型具备的特性之一。具备语义完整性的模型可以使用工具由源模型自动生成能够了解建模者用意的目标模型。

研究元建模中语义完整性问题的目的，一是为了精确描述对象，二是为了保证模型在变换中保持某种一致性。在元建模中，变换是按照变换定义从源模型自动生成目标模型[6]。

2　元建模中的约束

2.1　UML 建模

元建模和建模活动一样，需要工具定义元模型。文中沿用已经被广泛使用的 UML 工具，并结合 OCL 描述元模型。UML 作为一种图形建模语言，包括具体语法、抽象语法、语义，共有 8 种图形建模语言[2]：用例图、类图、状态图、顺序图、活动图、协作图、构件图和配置图。其中，

(1)UML 的抽象语法用元对象设施标准(MOF)描述；

(2)UML 的具体语法采用类图等描述；

(3)UML 的优化规则用 OCL 描述；

(4)具体语法到抽象语法的映射，以及抽象语法到语义的映射可以用模型转换描述；

(5)UML 的语义方面，可以采用翻译语义，例如 MOF 之类的映射，MOF to JMI mapping，也可以采用操作语义，即解释一个形式良好的 UML 抽象语法树的执行结果是一个什么样的 UML 抽象语法树。

MOF 元模型的核心是元建模的构件，也就是 MOF 的“抽象语言”，包括 4 个主要的建模构件。

1)类(Class)，对 MOF 元对象建模。

2)关联(Association)，对元对象之间的二元或多元关系建模。

3)数据类型(DataTypes)，对其他的数据，例如基本类型，外部类型等建模。

4)包(Package)，使模型模块化。

2.2　UML 建模的缺陷

使用 UML 进行元建模存在的缺陷有：

(1)对象的属性：无法明确对象属性的初始值，以及其取值范围；

(2)对象的操作：无法明确如何具体实现对象的操作；

(3)派生类：无法明确派生类的由来；

(4)属性以及操作的参数：无法明确对象的属性和操作的参数值的具体含义以及规定；

(5)关联：无法明确对关联端的要求；

(6)操作返回的数据类型在图形上也是无法明确表达的。

3　实现元建模语义完整性的方法的特征

既然 UML 在元建模中，无法描述元模型对象的属性、操作等，就需要设计出一种办法，解决其缺陷，使得元建模的语义更完整，模型更加健壮。那么这种方法应该具备的特征是：

1)能够约束各元模型的构件。

2)不可以超出 MDA 框架。

3)不改变模型本身。

4)这个方法表达的，同 UML 一样可以通过模型转换生成代码。

4　OCL 语言提高 UML 的精确性

4.1　OCL 与 UML 的联系

OCL 是一种用于表达施加在模型元素上的约束的语言[7]。OCL 表达式以附加在模型元素上的条件

和限制的形式来指定规则。这包括指定附加在模型元素上的不变式(Invariant)或约束的表达式(Expression),附加在操作和方法上的先验条件和后验条件(Precondition and Postcondition),监护条件(Guard condition),以及模型元素间的导航(Navigation)。OCL 表达式附加在模型元素上,模型元素的所有实例都必须满足该表达式的条件。虽然 OCL 是文本形式,而不是可视化的,但是这正好体现了 UML + OCL 半文本半图形的特性。它们相辅相成,如果没有 OCL 表达式,模型就会不完善;如果没有 UML 图,OCL 表达式就没有可以依赖的模型元素。只有结合 UML 和 OCL,才可以相得益彰,完善元建模。

OCL 表达式描述模型的附加信息,通过建立与 UML 图相连的机制,和 UML 一起放入模型仓库中。OCL 表达式通过 OCL 表达式中的上下文(Context)与 UML 图中的实体相连。上下文可以是类、接口、数据类型、组件(Component)等。

4.2 OCL 应用于元建模

OCL 应用于元建模时,既可以完善源模型和目标模型,又可以应用于变换定义语言[8]。在这里考虑 OCL 提供初始化、派生规则、查询操作、不变式等规则应用于源模型,即由 PIM 变换成精确的 PIM。

使用 UML 为学生学习成绩系统建模,见图 1,可以获知:每个学生 student 拥有一张名片 studentcard;一个学生可以选择多门课程;一门课程可以由多个学生选择;每门课程有一份成绩单;一个学生有一份成绩单。

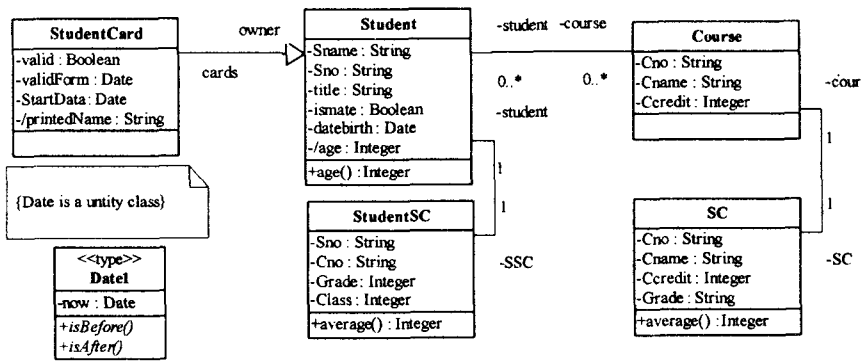

图 1 学生成绩管理实例

在这个 UML 描述的元模型中,不能够体现的信息包括:

(1)PrintedName 的构成;

(2)age()函数的计算;

(3)每个属性的初始值;

(4)约束:一个学生的成绩单上的课程包含他选择的所有课程;

(5)每门课程的成绩单上的学生必须选择了这门课。

这些都是构件系统时必须了解的信息。下面导入 OCL 语言解决这些问题。

(1) 派生规则(Derivation rule)。

在派生规则中,用 OCL 表述派生。例如,所有的 printedName 值是从 title 和 name 的组合而来的。

```
context StudentCard::printedName
derive:
owner.title.concat(' ').concat(owner.name)
```

(2) 初始化(Initial)。

属性值和关联角色(Association role)值可以用 OCL 表达式初始化。初始化和派生的规则的不同在于:派生规则描述不变式,派生元素的值永远符合这个规则;初始化的值只在建立这个语境实例时生效。

```
context SC :: Grade
init : 0
context StudentCard :: Valid
init : ture
```

(3) 查询操作(Body of query operations)。

查询操作不能改变系统实例的值,它只返回值或值的集合(Set)。例如返回选择某门课程的学生姓名:

```
context Course::getStudentName():String
body:self.Student.sname->asset()
```

(4) 不变式(Invariant):是对系统状态的断言,表示系统状态一定要满足不定式。在某种意义上,不变式必须可以看作所有操作的普适先验条件和后验条件。

例如:SC 中出现的 Sno 和 Cno 在 Student 和 Course 表中都必须存在且唯一。

```
context SC
init : self.course.student.Sno
->
        asset()->size()=1
and self.course.Cno->asset
()->size()=1
```

(5) 先验和后验条件(pre-/post-condition)。

先验/后验条件是对操作的断言。此表达式指定调用一个操作必须满足先验条件,或者一个操作完成后必须满足后验条件。如果先验条件和后验条件未被满足,操作的行为就不确定,而不是表示条件不能被违背。

OCL 表达式还包括:

(6) 后验条件中的消息(Message),在后验条件检

验消息是否已经被发送完。

(7) 监护条件(Guard condition),指定触发消息、循环、转换的附加条件。

(8) 导航(Navigation),在一个类图中通过导航可以由一个对象写所有对象的约束。

(9) 定义派生类(Defining Derived Class),一个派生类的特征,能够从已有的类或者其他的派生类中完整的派生出来。

(10)多样性选择(Optional Multiplicity),关联的选择多样性,表明选择是自由。

(11) 类模型中的循环。OCL 通过这些机制可以把一个元模型变换成精确的元模型。

5 约束冲突

在系统设计或者建模中,往往会产生异常[9]。许多异常可以直接从约束推断出来:操作的先验条件可以直接映射成为调用操作时先验条件不满足而导致的异常;为类声明的不变式可以映射成不变式不满足时引发的异常;因为关联的关系,约束之间可能会产生冲突,例如,关联中的继承,假设使用 OCL 语言约定父类的属性 A 的数值大于 10,而同样的,可能用 OCL 语言约定子类的相应属性小于 10,从而导致产生冲突。一旦产生异常,可以直观地从约束表达式中检测出。

在系统设计时,也可能会设计出错。一些错误可以通过 OCL 检测,下面以 UML 序列图为例说明。

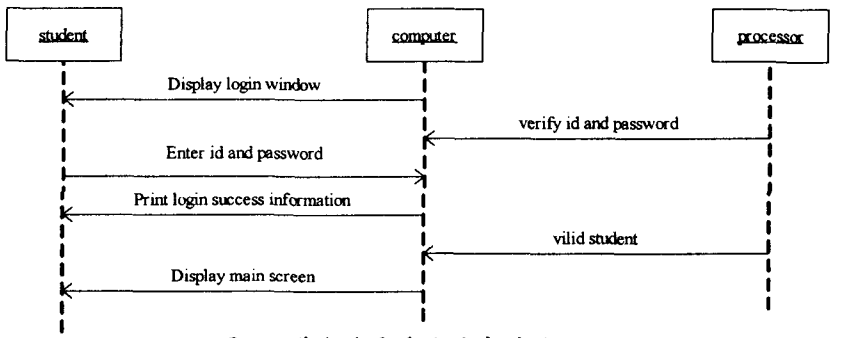

图 2　学生登录系统的序列图

如图 2 学生登录系统的序列图所示,为了确保登录的正确性,对每一个对象都附带一些约束。Computer 有三种状态 logstate(1:logout,2:logging,3:logged),id 和 password 有两种状态:是否被输入 idpasgiven(false:no,true:yes)。

context computer:: Display login window
pre : logstate=1 and idpasgiven=false
post: logstate=2 and idpasgiven=false
context computer:: Enter id and password
pre : logstate=2 and idpasgiven=false
post: logstate=2 and idpasgiven=true
context computer:: Verify id and password
pre : logstate=2 and idpasgiven=true
post: logstate=2 and idpasgiven=true

将这些约束映射到时序图上发现:Verify id and password 的先验条件中的 idpasgiven 为 fasle,不满足约束条件,从而发现序列图中的语义出现了错误,可以及时纠正过来。

6 结束语

UML 难以保证元模型的语义完整性。文中针对 UML 的不足,提出了结合 OCL 这种约束语言,增强元建模机制中元模型语义的完整性。并通过一些实例验证该方法的有效性和实用性。但是,OCL 语言的语义还不是很完善,还要继续深入研究、实践。在现在已有的工具中,支持 MDA 建模自动从 PIM 到 PSM 的工具也还不完善,需要继续研究形式化的语言来保证。

参考文献:

[1] 王　远, 范玉顺. 工作流时序约束模型分析与验证方法[J]. 软件学报, 2007,18(9):2154-2161.

[2] 庞　辉,方宗德,赵　勇.时间约束工作流模型的简化分析与可调度性验证[J].计算机集成制造系统, 2008,14(11):2217-2223.

[3] 吴　健,吴朝晖,李　莹,等.基于本体论和词汇语义相似度的 Web 服务发现[J].计算机学报,2005,28(4):595-602.

[4] 孙建召,曾巧明.基于面向对象 Petri 网的工作流建模及性能分析[J]. 计算机技术与发展, 2007,17(10):73-75.

[5] Boronat A, Meseguer J. An Algebraic Semantics for MOF[J]. Fundamental Approaches to Software Engineering, 2008(4):377-391.

[6] Engels G,Soltenborn C,Wehrheim H. Analysis of UML Activities Using Dynamic Meta Modeling [J]. Formal Methods for Open Object - Based Distributed Systems,2007(6):76-90.

[7] 杨玉梅,刁永锋.基于 UML 顺序图的 Petri 网建模[J]. 计算机技术与发展, 2007,17(10):130-133.

[8] Rivera J E, Vallecillo A. Adding behavioral semantics to models[C]//Enterprise Distributed Object Computing Conference. [s.l.]:[s.n.],2007:169-178.

[9] 李　丹,陈启璋,刘　强.一种基于 Petri 网的时间工作流模型的研究与验证[J].计算机工程,2007,33(4):78-80.