一种面向方面的模型转换语言 AOMTL*

方义秋[a], 贺 蕾[a], 葛君伟[b]
(重庆邮电大学 a. 计算机学院; b. 软件学院, 重庆 400065)

摘 要: 针对面向方面的 PIM 到面向方面的 PSM 的转换,提出了一种面向方面的模型转换语言 AOMTL。首先建立方面模型的转换框架;然后根据此框架建立 AOMTL 的元模型,设计 AOMTL 的具体语法;最后使用 AOMTL 完成方面模型转换。该方法为形式化及自动化方面模型转换的描述与实现提供了一种有效的解决办法。

关键词: 面向方面软件开发; 模型驱动架构; 平台无关模型; 平台相关模型; 模型转换
中图分类号: TP311.5 **文献标志码**: A **文章编号**: 1001-3695(2010)06-2150-03
doi:10.3969/j.issn.1001-3695.2010.06.044

Aspect-oriented model transformation language AOMTL

FANG Yi-qiu[a], HE Lei[a], GE Jun-wei[b]
(*a. College of Computer, b. College of Software, Chongqing University of Posts & Telecommunications, Chongqing* 400065, *China*)

Abstract: Through the transformation of aspect-oriented PIM to PSM, this paper presentd a model for aspect-oriented transformation language AOMTL. Firstly, built the transformation framework of aspect model, based on which built AOMTL metamodel and designed AOMTL concrete syntax. Finally, fulfilled the transformation of aspect model by using of AOMTL. This method provided an effective solution for the description and implementation of aspect model transformation with formalization and automation.

Key words: aspect-oriented software development; model driven architecture; platform independent model; platform specific model; model transformation

随着软件开发过程的复杂化,软件开发的方法越来越受到重视,近年来提出了将 AOSD 和 MDA 的融合及相互支持作为研究课题[1]。而方面模型转换是这个研究领域发展的核心环节,直接影响到软件开发的结果。方面 PIM 到方面 PSM 的转换方法的研究,其有效性和准确性能够提高软件的开发效率。目前,PIM 到 PSM 的模型转换方法虽然都有自己的特点,但是这些模型转换方法普遍都存在较复杂、不直观、转换效率不高的问题,而且缺乏形式化的定义,也很少在实践中得到检验。

ATL(atlas transformation language)是一种混合语言,语法简单易懂,操作方便,对于编程人员来说很容易理解和应用[2],而 OCL 是一种声明式语言[3],具有定义简洁、抽象层次高等特点。以 ATL 与 OCL 结合为基础,本文提出了一种模型转换语言 AOMTL 来进行方面模型转换。该方法结合了 ATL 和 OCL 的特点和长处,用 OCL 对方面模型进行约束和补充定义,用 ATL 来实现方面模型的转换,既能弥补 ATL 不能体现系统动态特性和保证数据一致性的不足,又能弥补 OCL 转换效率低下的不足,因此是实现方面 PIM 到方面 PSM 转换的一种有效方法。

1 面向方面模型转换框架

1.1 基于 MDA 模型转换过程

一个模型生成另一个模型,必须把源模型中的每个元素同目标模型中的一个或多个元素建立联系。这样做的最直接方式是把源模型元素的元类与目标模型元素的元类建立联系。因为模型元素与元类之间有实例—类型关系,元类在元模型中的每个实例都必须遵从对元类设置的规则。转换过程如图 1 所示。根据图 1 可以得知模型转换是用转换定义来描述的,转换定义包含一组转换规则,这些规则由转换工具来执行;转换规则应当具有可调性、可追溯性和增量一致性[4]。

1.2 面向方面的模型转换框架

根据模型转换过程,结合 AOSD 的思想,可以得出面向方面的模型转换是将面向方面 PIM 元模型中所定义的元类转换到面向方面 PSM 元模型中定义的元类上,可以通过定义面向方面的模型转换语言来形式化定义转换。一个语言的元模型,也被称为该语言的抽象语法,描述了所有能够在该语言中使用到的概念及其之间的联系。由此建立方面模型转换框架如图 2 所示。

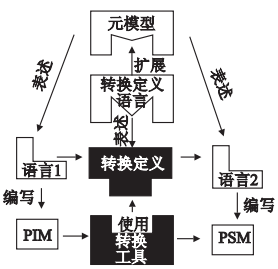

图1 模型转换过程

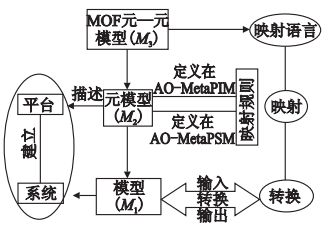

图2 方面模型转换框架

收稿日期: 2009-12-07; **修回日期**: 2010-01-24 **基金项目**: 国家自然科学基金青年基金资助项目(60403009);重庆市教委基金资助项目(KJ090519)

作者简介:方义秋(1963-),女,副教授,主要研究方向为软件体系结构、空间信息集成等;贺蕾(1981-),女,硕士研究生,主要研究方向为软件工程(helei81924@hotmail.com);葛君伟(1961-),男,教授,主要研究方向为地理信息系统、GIS 中间件等.

2　方面模型转换语言 AOMTL

2.1　AOMTL 分析

使用自然语言书写变换规则,对人类读者而言比较好理解,但对自动系统而言却并非如此。因此,需要一个可以插入转换工具的转换定义,这个转换定义能够自动执行变换。模型转换的形式化定义[5]:假定一个模型 m 是一个系统的形式化描述,用 s 表示系统,用 f 表示描述系统的形式化语言,那么这个系统 s 的模型就为 $m(s)$,就可以用 $m(s)/f$ 来表示领域模型,而模型之间映射的通用表达式就是:$m_1(s)/f_1, m_2(s)/f_2$。f_1 和 f_2 是元—元模型的实例,模型转换可用表达式 $m(f)$/MOF 表达式来处理。所以为了实现 AO-PIM 到 AO-PSM 的模型转换,定义一种新的模型转换语言 AOMTL 来形式化方面模型的转换。

AOMTL 采用基于 ATL 与 OCL 结合的方式描述变换规则,实现模型转换。在 ATL 中的转换规则可以使用完全声明的、混合的,或者是完全可执行的,但 ATL 功能单一,不能对模型进行精确的转换,也没有数据类型的验证功能,不能保证模型转换前后数据的一致性[2]。而 OCL 有前验、后验表达式和不变式的概念,这些则可以描述系统的动态特性,它还能对模型进行补充定义,保证模型在转换前后数据的一致性,因此用 OCL 一定程度上可以消除模型的二义性问题[6]。根据以上分析,分别吸收 ATL 和 OCL 各自的优点,建立模型转换语言 AOMTL。

2.2　AOMTL 的元模型

在统一的 MOF 元模型构架下,根据上节所述建立的方面模型转换框架,定义了 AOMTL 的元模型,使得 AOMTL 能够得以实现。因为方面模型转换是定义在方面元模型级(M_2)上的,它可以被应用在任何由此元模型定义的方面模型(M_1)上。又因为方面元模型是由 MOF 所定,方面模型转换所涉及的参数,也就是方面元模型中的元素,都是由 MOF 元—元模型所定义好的,适用于方面元模型的模型转换语言 AOMTL 也就能够得以实现。图 3 为 AOMTL 的元模型。

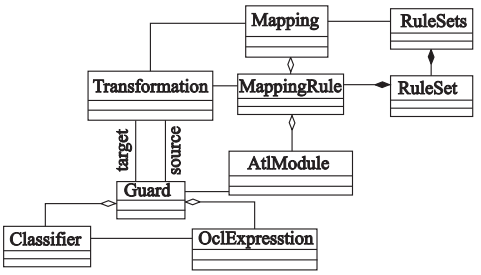

图3　AOMTL的元模型

2.3　面向方面模型转换语言 AOMTL 的设计

2.3.1　AOMTL 的数据类型

AOMTL 是基于 MOF 的,其类型的实例结构的根元素是抽象的 oclany 类型,所有其他的各种类型数据都是从它直接或间接继承的。主要包括六种数据类型,即原始数据类型、集合数据类型、元组数据类型、映射类型、枚举类型和模块元素类型。

2.3.2　AOMTL 模块

一个 AOMTL 模块符合模型到模型的转换,这种 AOMTL 单元能够让开发人员制订从 AO-PIM 产生 AO-PSM 的方法。一个 AOMTL 模块中的源模型 AO-PIM 和目标模型 AO-PSM 必须被各自的元模型"typed"。另外,一个 AOMTL 模型接收一个固定数量的模型作为输入,并产生固定数量的输出。AOMTL 不能处理未知数量的模型。

AOMTL 模型的结构:

```
{Header section;[Import section];Helper;Rulesets;}
```

Header section 定义了转换模型的名字以及相应源模型和目标模型变量的名字。在它内部能对模块的可执行模型进行编码。这部分语法定义如下:

```
Module name
Create AO-PSM [from|refines] AO-PIM
```

Improt section 是个可选项,用于要引入 AOMTL 库的声明,使用如下:

```
Uses library_name;
```

Helper 是 AOMTL 的方法,可以定义 AOMTL 代码,这些代码能够被别的 AOMTL 转换所调用。一个 AOMTL Helper 用下面的元素定义:一个名字、一个环境类型、一个返回值类型、一个 AOMTL 表达式和一些参数集合选项。例如在模型转换中操作名字转换方法的定义:

```
Helper context String def:getGetterOperationName:String = …;
```

Rulesets 是由多个规则集(Ruleset)组成。Ruleset 反映了一个模型通过一组变换规则映射到另一个模型的变换过程,它是组成一个变换所用到的一系列规则的集合,包括多条变换规则(Rule)。Rule 描述了从源模型的一个或多个元素到目标模型的一个或多个元素的映射关系。一个规则节点包含名称、源条件、目标条件、动作,以及参数声明等几个主要部分。Rulename 属性是变模规则的名称。源条件用于描述规则的匹配条件,它是一个集合,可以容纳多个条件项的定义。这些不同的定义包括类型条件、值条件、一致性条件等;其内容可由扩展的 OCL 书写。目标条件与源条件结构类似,但意义不同,源条件是对源模型中已有的模型元素作出判断,目标条件是被变换引擎用做生成模型元素以及建好这些模型元素的关系。动作指定从源到目标模型元素的变换过程中要采取的一系列行为,包括值拷贝、子规则调用。参数部分指定了规则中设定的可变部分。具体表示如下:

```
Rule rulename {
  From source To target
  //source condition;
  //target condition;
    Map();}
Ruleset sname {
    rule rulename1;
    rule rulename2;
  …;}
Rulesets name {
    ruleset sname1;
    ruleset sname2;
  …;}
```

2.3.3　AOMTL 库

AOMTL 库是由定义的一系列的 AOMTL Helper,这些 Helper能够被不同的 ATL 单元所调用。与其他单元一样,AOMTL 库也包括一个可选的引入部分,当 AOMTL 库导入 AOMTL 单元中后,这些 Helper 是可用的。

3　AOMTL 的实现

为了能够使用 AOMTL 形式化定义方面 PIM 到方面 PSM 的转换,需要方面 PIM 和方面 PSM 的元模型,在变换定义中需要引用这些元模型中的元素以便给出形式化定义。因此,需要建立方面 PIM 元模型和方面 PSM 元模型。通过对 UML 元模

型[7]的扩展,建立了方面 PIM 的元模型。图 4 描述了这个元模型,它主要扩展了 Classifier、Feature 和 Association 元类,增加了 Aspect、PointCut、Advice 和 Crosscut 四个元类。

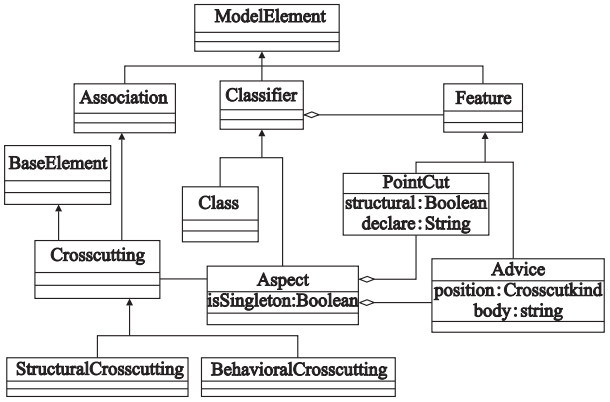

图4　方面PIM元模型(AO-Meta PIM)

AspectJ 是一种针对 Java 扩展的面向方面语言,本文就以 AspectJ[8]为例,构建面向方面 PSM 的元模型。图 5 显示了该 PSM 元模型。

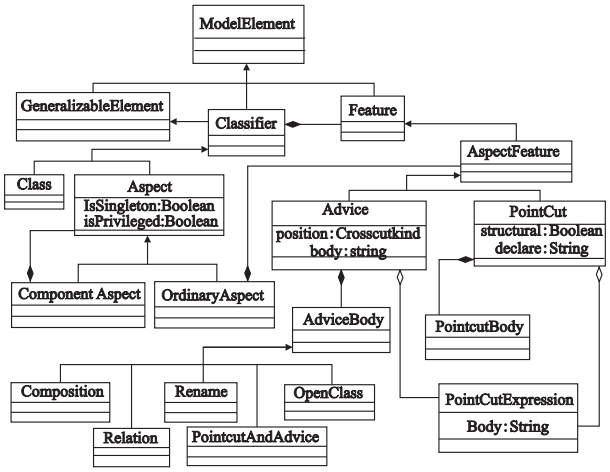

图5　方面PSM元模型(AspectJ PSM)

基于方面 PIM 和 PSM 的元模型,使用 AOMTL 描述变换规则,完成方面 PIM 到 AspectJ PSM 的转换。图 6 描述了方面 PIM 到方面 PSM 的转换过程。

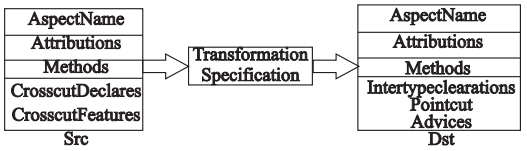

图6　方面PIM到AspectJ PSM的转换

其主体部分的 AOMTL 描述如下:

```
Rule Aspect {//Aspect 的转换规则
  From  s: Src! AOPIM  To  t: Dst! AOPSM
Pre:
s. attribute- > include (att) and att. visibility = #public
Maps to s (
name < - s. name;
attribute( ) < - s. attribute( );
isPrivileged < - s. isPrivileged;
declearedImplements < - s. declearedImplements;
declearedParents < - s. declearedParents;
perType < - s. perType;
Introductions( ) < - s. Introductions( );
PointCuts( ) < - s. PointCuts( );
Advice( ) < - s. Advice( );
Operations( ) < - s. Operations( );
) }
```

对于方面 PIM 中的每个名为 AspectName 的方面,目标模型 PSM 中要有一个同名的方面对应。对于 PIM 中的 AspectName 方面的每一个属性,目标模型 PSM 中要有一个与其同名的属性,并且类型也一样。对于 PIM 中的 Method( ):Returntype,在 PSM 中有同样的 Method( ):Returntype 与其对应。

```
Rule Crosscut{//Crosscut 的转换规则如下:
  From  s: Src! AOPIM  To  t: Drc! AOPSM
Pre:
CrosscutDeclare - > include( declare) and Structural = ture and declare = declare parents
Maps to s(
Intertypedeclare < - s. declare; ) }
Rule other Intertypedeclaration{…}
```

对于方面 PIM 中的横切的声明表达式 CrosscutDeclare,在方面 PSM 中,要有一个选定横切对象的声明性结构与之对应。根据 CrosscutDeclare 值在方面 PSM 中有对应的横切表达式和横切声明成分,对于 PIM 中的 CrosscutDeclare 的每一个结构性横切成分,在 PSM 的横切分栏中有相应的成分对应,如 AspectJ 的 Intertypedeclaration;对于 PIM 中 CrosscutDeclare 的行为性横切,在目标模型中要有对应的包含平台相关描述的横切声明成分 Pointcut;对于 PIM 中动态横切的操作部分 CrosscutFeature,在目标模型方面 PSM 中要有一个使用相同类型修饰符的同名操作,并明确 CrosscutDeclare 的对应关系,使用平台相关的操作。

```
Module Src2Dst;//方面模型的转换
  Create OUT: Dst from IN : Src;
  Ruleset AOPIMtoAOPSM{
    rule Aspect;
    rule Crosscut;
    rule other Intertypedeclaration;
    …;}
```

4　结束语

MDA 与 AOSD 的融合和相互支持代表着今后软件开发的一个方向,其关键就在于方面模型的开发与转换。方面模型转换描述了方面 PIM 如何转换到平台相关的方面 PSM。基于 ATL 和 OCL 设计的方面模型转换语言 AOMTL 目的是形式化方面模型的转换,使得方面模型转换能够用一致的方法来表达,并具有无二义性的特点。因此 AOMTL 对形式化及自动化方面模型转换的描述与实现是可行的和有效的。

参考文献:

[1] STEIN D, HANENBERG S, UNLAND R. Expressing different conceptual models selections in aspect-oriented design[C]//Proc of the 5th International of Join Point Conference on Aspect-Oriented Software Development. 2006:15-26.

[2] JOUAULT F, KURTEV I. Transforming models with ATL[C]//Proc of Model Trans in Practice Workshop at Models. Jamaica: Springer-Verlag, 2005:220-240.

[3] Object Management Group. OCL 2.0 specification[DB/OL]. (2006-05-01). http://www.omg.org/.

[4] 王学斌,王怀民,吴泉源,等. 一种模型转换的编织框架[J]. 软件学报,2006,17(6): 1423-1435.

[5] 陈晓燕,赵建华,李宣东. 一个 MDA 支撑工具的设计与实现[J]. 计算机工程与设计,2005,26(1):39-43.

[6] 马浩海,高光来. 基于图形语法的 UML 模型转换方法[J]. 内蒙古大学学报:自然科学版,2005,36(5):560-564.

[7] Object Management Group. Unified modeling language specification, version2.1.2 [DB/OL]. (2007-11-01). http://www.omg.org/.

[8] COLYER A. Eclipse AspectJ 中文版:利用 Eclipse 和 Aspect J 进行面向方面程序设计[M]. 钱竹青,邹正武,译. 北京:清华大学出版社,2006:109-288.