ISSN 1000-9825, CODEN RUXUEW
Journal of Software, Vol.17, No.6, June 2006, pp.1423−1435
DOI: 10.1360/jos171423

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

一种模型转换的编织框架*

王学斌+, 王怀民, 吴泉源, 史殿习

(国防科学技术大学 计算机学院 网络与信息安全研究所,湖南 长沙 410073)

A Weaving Framework for Model Transformation

WANG Xue-Bin+, WANG Huai-Min, WU Quan-Yuan, SHI Dian-Xi

(Institute of Network Technology and Information Security, School of Computer, National University of Defense Technology, Changsha 410073, China)

+ Corresponding author: Phn: +86-731-4573694, Fax: +86-731-4512504, E-mail: wxb_nudt@163.com, http://www.nudt.edu.cn

Wang XB, Wang HM, Wu QY, Shi DX. A weaving framework for model transformation. *Journal of Software*, 2006,17(6):1423−1435. http://www.jos.org.cn/1000-9825/17/1423.htm

Abstract: Model transformations are the core and research focus of MDA(model driven architecture). Nowadays, there are many model transformation approaches and tools, and the heterogeneity of them causes that the model transformation code can’t be reused easily and the cost of studying and using model transformation increases. To eliminate these shortcomings, inspired by model weaving technique, this paper proposes a QVT(model query/view/transformation)-based model transformation weaving framework named QMTW(QVT-based model transformation weaving framework). The conception, semantics, metamodel and syntax of QMTW added with a mapping from itself to the QVT Relation language are subscribed. A study case is provided to show how developers can use and take advantage of QMTW. The approach promotes the abstraction layer of model transformations, unites other model transformation languages, and supports QVT specification. So it eliminates the heterogeneity of model transformation partly. It is simple, specification-based and extensible.

Key words: model weaving; model transformation; MDA(model driven architecture); QVT(model query/view/transformation); software architecture

摘 要: 模型转换是 MDA(model driven architecture)的核心技术之一,也是目前 MDA 研究的热点.目前,MDA 范畴内存在多种模型转换方法和工具,它们之间的异构性造成了模型转换代码重用的困难,并使学习和使用模型转换方法的成本增加.受到模型编织技术的启发,提出了一种基于 QVT(model query/view/transformation)规范的模型转换编织框架 QMTW(QVT-based model transformation weaving framework)来解决以上缺点.展示了模型转换编织的概念、语义、元模型和语法,以及到目标语言的转换定义,并以一个具体实例说明了本框架的使用方法和优点.QMTW 提高了模型转换的抽象层次,统一了多种模型转换语言,并支持 OMG 最新的模型转换规范,在一定程度上消除了模型转换技术的异构性,同时具有简单、规范、扩展性强 3 个优点.

* Supported by the National Natural Science Foundation of China under Grant No.90412011 (国家自然科学基金); the National Grand Fundamental Research 973 Program of China under Grant No.2005cb321800 (国家重点基础研究发展规划(973)); the Program for Changjiang Scholars and Innovative Research Team in University of China under Grant No.IRT0446 (教育部创新团队)

Received 2006-01-09; Accepted 2006-03-13

关键词: 模型编织;模型转换;MDA(model driven architecture);QVT(model query/view/transformation);软件体系结构
中图法分类号: TP301 文献标识码: A

提高软件质量和弥补不同技术平台间的差异是目前软件界最为迫切需要解决的问题.模型驱动架构(model driven architecture,简称 MDA)[1]被认为是达到上述目的的有效途径之一.作为由 OMG 组织推动的一种新的软件开发方法,它以模型作为软件开发的核心元素,并为其提供了多种技术规范作为基础框架.在 MDA 出现之前,软件体系结构(software architecture,简称 SA)的研究受到了广泛的关注和重视,被认为将会在软件开发中发挥十分重要的作用.SA 的目的是弥补需求分析工程和软件建模技术之间的鸿沟,避免需求分析的结果不能体现在软件建模中,从而导致需求分析和软件实现的脱节[2].而 MDA 的目的是解决从软件建模到具体实现中面临的种种问题,减弱各种语言和中间件平台的差异对软件开发造成的影响.MDA 和 SA 是两种相辅相成的技术,SA 中可以使用 MDA 中的建模语言和建模方法来表示其体系结构,而 SA 中的体系结构又可作为 MDA 中的模型来进行转换、检验、查询和维护.

MDA 中需要解决的两个主要问题是如何有效地建立软件模型以及如何在模型之间进行有效的转换.前一个问题目前已经达成了较为一致的意见,即以 UML 及其扩展机制作为建模的标准语言*;而后一个问题目前仍然没有一个统一的答案.模型转换是模型驱动架构中的核心技术,其目的是为了解决软件开发过程中以模型作为基础元素的查询、变换、视图等技术问题.目前,支持模型转换的方法和平台很多,例如 MTF,ATL[3],MTL[4]等,但现有的各种模型转换方法和工具各自使用了自定义的语言和规范,导致了模型转换技术的异构性.这种异构性带来了两个问题:一是现有的模型转换代码不能在其他工具中重用;二是多种并存的模型转换语言导致模型转换技术的学习成本增加,使用困难增大.

为了消除此异构性,OMG 在 2002 年提出了一个 QVT(model query/view/transformation) RFP[5],其目的是为了对模型的查询、视图和转换定义一个统一的规范.其后相继收到了 8 个 QVT 的提案[6],经过筛选、淘汰与合并,最后在 2005 年 7 月形成了一个最后的联合修正案[7]**.此修正案即 QVT 规范的初稿,OMG 期望这个标准能够统一模型转换技术.但是,短期内将各种模型转换语言统一到 QVT 中是不现实的,毕竟在各个工具上已经积累了众多的模型转换代码.从另一个角度说,也不一定所有的研究机构和工具厂商会完全支持 QVT 规范.那么,模型转换技术的异构性仍然是当前一个不可回避的问题.

为了解决模型转换的异构性,同时遵循 QVT 规范,还要支持现有的模型转换语言,本文基于模型编织的思想提出了一种解决方案——基于 QVT 的模型转换编织框架 QMTW(QVT-based model transformation weaving framework).模型编织是模型转换领域中的新概念,它能够提升模型转换的抽象层次,构造一个称为模型编织(model weaving)的抽象层,在此抽象层上可以定义编织模型(weaving model),它代表了一组模型转换规则,然后生成目标语言的模型转换代码.文献[8,9]描述了基于 ATL 语言构造的一个模型编织框架 AMW,它可以描述基于 ATL 语言的模型转换规则.但是,这种方法存在 3 个弱点:1) 描述模型转换的能力不足,例如,它只能描述一个源模型和一个目标模型间的转换,不能描述两个以上模型间的转换定义;2) 没有与 QVT 规范相结合,不具有规范性;3) 没有定义从编织模型到具体模型转换语言的代码生成机制.

QMTW 将模型编织的思想运用于解决模型转换异构性的问题,在更高的抽象层次上定义了模型转换的语义.遵循此语义的编织模型可以转换为多种模型转换语言,包括 QVT 规范中的 Relation 语言.因为它基于 QVT 规范,所以对转换规则的描述能力更强,基本上能够涵盖所有基于规则的模型转换语义.同时还定义了可扩展的代码生成机制,可以将编织模型转换为多种目标语言.与 AMW 相比,它更加符合规范,具有更强的规则描述能力和可扩展能力.

---

* 虽然目前仍存在 UML 和 DSL 之间的争论,但在通用建模领域,UML 已成为事实上的标准.

** 本文中所指的 QVT 规范即这个联合修正案,在提交本文时,这个修正案还没有正式成为规范.

图 1 比较了 QMTW 的层次结构与常用模型转换语言的差异.从层次结构上来看,QMTW 比常用模型转换语言略为复杂.图中左边是模型转换的层次结构,右边则是 QMTW 的层次结构.从图中可以看出:在模型转换语言中,源元模型 SrcMM、目标元模型 DesMM 和转换语言元模型 MMt 同属于 MOF[10]中的 M2 层,而源模型 SrcM、目标模型 DesM 和转换定义 Mt 同属于 M1 层.在 QMTW 中增加了一个模型编织元模型 QMTW 和一个编织模型 WM.从理论上来说,QMTW 是由 MOF 直接定义的,属于 M2 层,而 WM 是由 QMTW 定义的,属于 M1 层.但 Mt 又是由 WM 生成的,所以它应该属于 M0 层,这样就与前面的结论形成了矛盾.事实上,模型编织中的 WM 和 Mt 同属于 M1 层,它们只是相同语义的不同表达方式,并能够互相转换.本方法的优势在于:QMTW 定义了一个平台/语言无关的模型转换元模型,符合这个元模型的编织模型 WM 描述了一个平台无关的模型转换规则模型.利用此平台无关模型,WM 可以生成多种符合特定技术平台的目标语言代码,这样,部分地解决了模型转换技术的异构性问题.

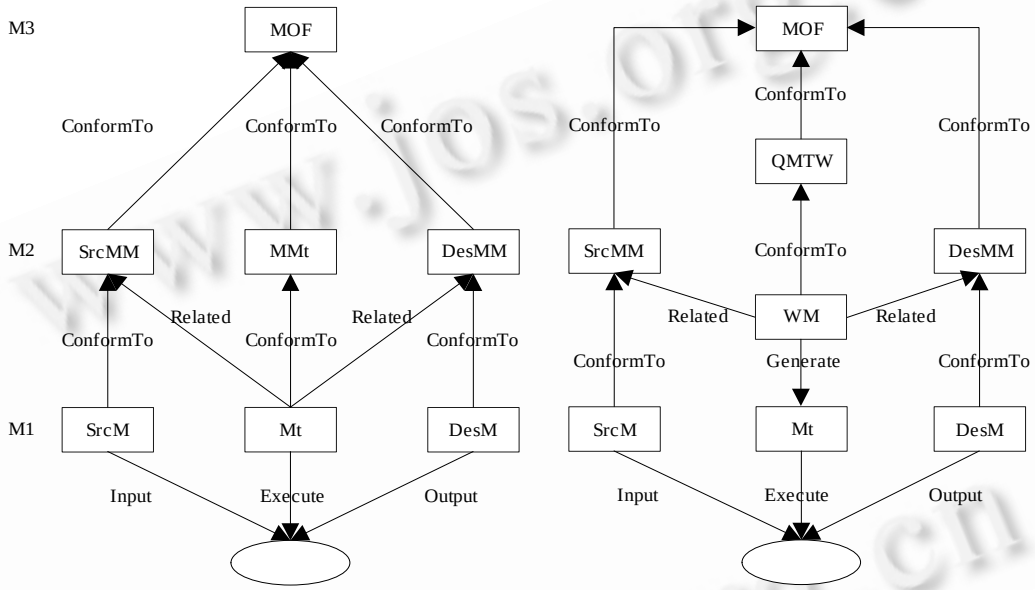

Fig.1 Comparison between model transformation layers and QMTW layers

图 1 模型转换层次与 QMTW 层次比较图

与模型转换语言相比,QMTW 具有如下优点:

(1) 简单性,降低了学习模型转换方法的复杂性,且不必一一学习复杂的模型转换语言;

(2) 规范性,基于 QVT 规范以及 OMG 的其他规范 MOF,OCL[11]等;

(3) 扩展性,可以扩展到多个模型转换语言,并且自身可以再扩展,以适应不同领域的需要.

本文的第 1 节介绍 QVT 规范的框架结构及其关系型模型转换语言 Relation.第 2 节介绍 QMTW 的基本架构,包括概念、抽象语法和语义以及具体语法.第 3 节介绍一个使用 QMTW 构建编织模型的实例.第 4 节以上述编织模型为例来描述从 QMTW 到 Relation 语言的转换定义,并附有相关代码.第 5 节介绍本领域内的国内外相关工作.第 6 节对本文工作进行总结,并指出未来的工作内容.

1 QVT 架构

QVT 是用 MOF 定义的,它位于 MOF 四层模型中的元模型层.它主要由 4 种模型转换语言和一个转换组成,而模型的查询则是使用 OCL 完成的,模型的视图功能在此规范中并没有得到解决.

Relation 语言是 QVT 的核心语言,它可以将模型转换规则描述为一组源模型元素与目标模型元素之间的关系(relation).每个 Relation 包含几个 domain,这些 domain 指向参与转换的模型元素,每个 domain 包含一组

pattern,通过匹配 pattern 可以选择出符合转换规则的模型元素.如果 pattern 不能匹配,则通过修改、删除、创建目标模型元素来达到匹配,同时完成模型转换.Relation 语言与文献[12]中介绍的关系型模型转换语言很类似,与目前一些常用的模型转换语言 ATL,MTF 等也有相同之处.因此,本文使用 Relation 语言作为 QMTW 的主要目标语言.除了 Relation 以外,QVT 还包括 Core,Operational Mapping 和 Black Box 这 3 种语言.其中,Core 与 Relation 同属声明性(declarative)语言,而 Operational Mapping 和 Black Box 则是命令性(imperative)语言.此外,QVT 还定义了从 Relation 语言到 Core 的一个转换定义 RelationsToCoreTransformation.

Relation 是一种规则型的、声明性的模型转换语言,它的语义涵盖了几乎所有规则型或者声明性模型转换语言的语义.本文提出的 QMTW 的语义与 Relation 的语义相等,因此,使用 QMTW 建立的编织模型有能力被转换为任意的关系型或者声明性模型转换语言.

2 QMTW 模型转换编织架构

模型编织是将 AOP(aspect oriented programming)思想引入模型转换领域中而出现的新概念.目前有两种不同的模型编织方法:第 1 种类似于传统的 AOP 方法,在模型中引入 model aspect,model pointcut 和 model join point,把转换规则写入一个模型编织定义文件,然后用 model weaver 读入源模型和编织定义文件,从而生成目标模型[4];第 2 种是将模型转换规则表达为有类型的模型元素连接,连接的类型代表了这种连接的语义.连接之间通过“联合”关联起来,这些联合表达了连接之间关系的语义[8].

QMTW 采用并扩展了第 2 种方法,更加精确地定义了模型转换编织架构并给出了一个代码生成机制.如图 2 所示,模型转换规则用连接来表示,连接之间通过“组织”来定义其间的依赖、引用、继承等关系.每个连接的类型对应一个代码段,可以通过模板的方式把一个连接转换为一个代码段.通过对组织的解释,可以将所有的规则代码段组合为最终的模型转换代码.由于连接类似于纬线,而组织类似于经线,一组转换规则就像一张交织的网,所以这种表达模型转换的方法就被命名为“模型转换编织”.

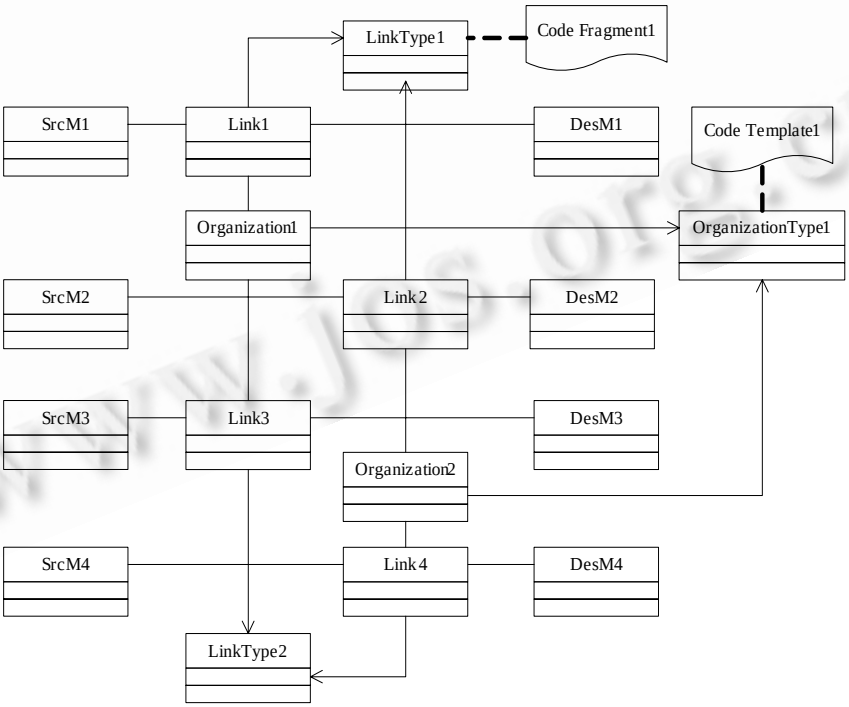

Fig.2 Model transformation weaving sketch map

图 2 模型转换编织示意图

2.1 QMTW概念

简单地说,一个模型转换方法应该具有如下功能:1) 访问源模型;2) 定义模型转换规则;3) 执行模型转换;4) 生成目标模型.根据以上功能,本文提出了一个基于 QVT 规范的模型转换编织元模型 QMTW,它包含如下概念:

模型元素引用:可以使得转换规则能够访问源模型和目标模型,并利用特定的机制对模型元素进行操作;

连接:将源模型和目标模型中的一些模型元素连接起来,并赋予这个连接一定的语义,使其表达一个转换规则;

组织:一个编织模型中往往含有多个连接,它们代表了多种转换规则.组织将这些规则以一定的方式关联起来,定义规则间的依赖、包含等关系,确保转换能够按照一定的步骤执行;

约束:定义了连接中的一些限定条件,例如连接的先决条件和连接的几个端点之间的约束等;

编织模型:定义了整个模型转换算法,包括对模型元素的访问以及转换规则、规则组织和约束.

注意,模型转换的执行要依赖于具体的模型转换语言.目标模型的生成也要依赖于具体的语言.

2.2 QMTW的抽象语法及语义

QMTW 的抽象语法用其元模型表示.为了清晰起见,我们将元模型拆分到两个图中,如图 3、图 4 所示.

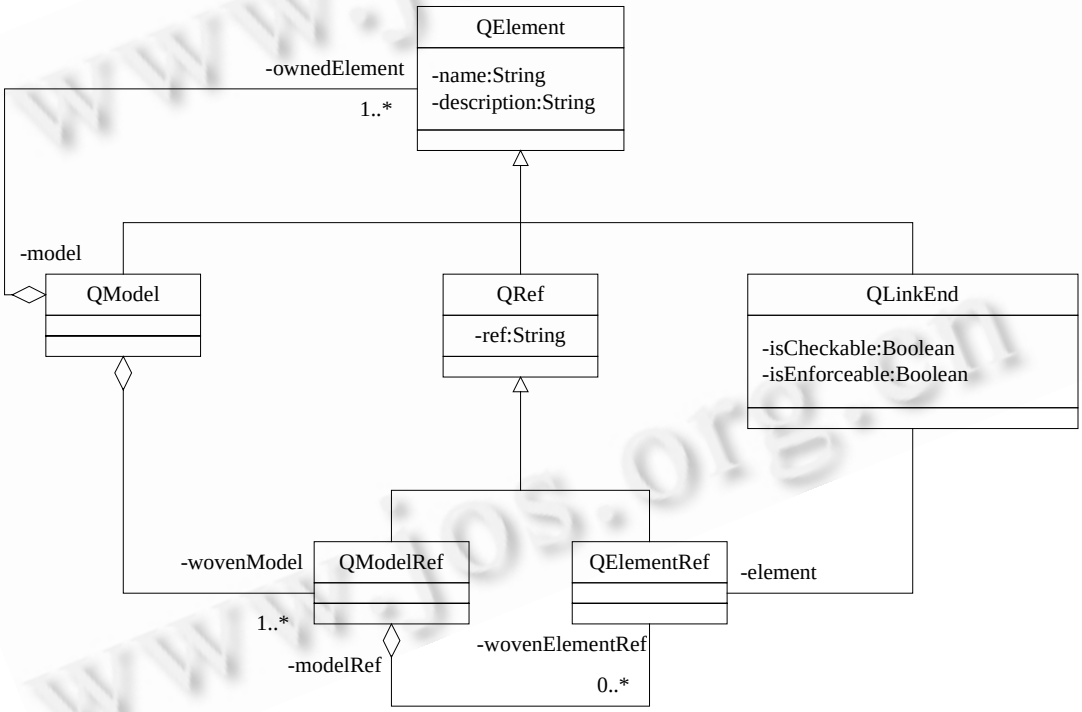

Fig.3 The part 1 of metamodel diagram of QMTW

图 3 QMTW 元模型图第 1 部分

下面依照上述的 5 个 QMTW 概念来说明图中各元模型元素的语义、属性和约束.

模型元素引用主要解决的是转换中对模型元素访问的功能.它主要由图 3 中描述的 QElement,QRef, QElementRef 和 QmodelRef 这 4 个元素来组成.QElement 是 QMTW 中所有元素的父类,对应于 EMOF 中的 Element;QRef 是一个抽象类,它定义了一个引用,指向一个源或目标模型元素.它可以看作是编织模型和源模型以及目标模型间的接口;QElementRef 和 QModelRef 都是 QRef 的子类,前者指向一个具体的源或者目标模型元

素,后者指向一个源模型或者目标模型.

连接的概念主要由图 4 中的 Qlink,QLinkEnd 和 QlinkType 这 3 个元素组成.QLink 定义了源模型元素和目标模型元素之间的一个连接,代表着一个模型转换规则.每个 QLink 包含两个以上的连接端 QlinkEnd.它有两个主要的作用:一是从连接端可以引用到参与此连接的模型元素;二是可以在连接端定义约束和连接端属性.每个连接有一个连接类型 QLinkType,它代表着此连接的具体语义,并且指向一组目标代码模板.一个 QLink 的 isTop 属性指明了这个连接是否为入口规则,一个 QModel 中应该最少具有一个入口规则.约束如下:

context QModel **inv**:

self.ownedElement->select(l|l.oclIsKindOf(QLink))->exist(lo|lo.isTop=true)

组织的概念由图 4 中的 Qorganization,QOrganizationEnd 和 QOrganizationType 共同组成.QOrganization 定义了多个连接之间的关系,例如依赖、包含等,一个编织模型中所有的组织共同定义了一个模型转换算法的结构,即模型转换规则的执行步骤和顺序.每个组织包含了两个以上的组织端 QOrganizationEnd,它定义了组织到一个连接的引用.每个组织有一个组织类型 QOrganizationType,它代表着此组织的具体语义,并含有一个组装连接代码模板的方案.

约束的概念由 QConstraint 来表达.约束可以附加在连接端上面,它继承了 OCL 规范中的表达式,可以用 OCL 来直接表达模型编织中的各种约束.

编织模型概念由 QModel 来表达.一个编织模型可以包含 QMTW 中的任何元素,例如连接、组织、引用和约束,即一组完整的模型转换规则.

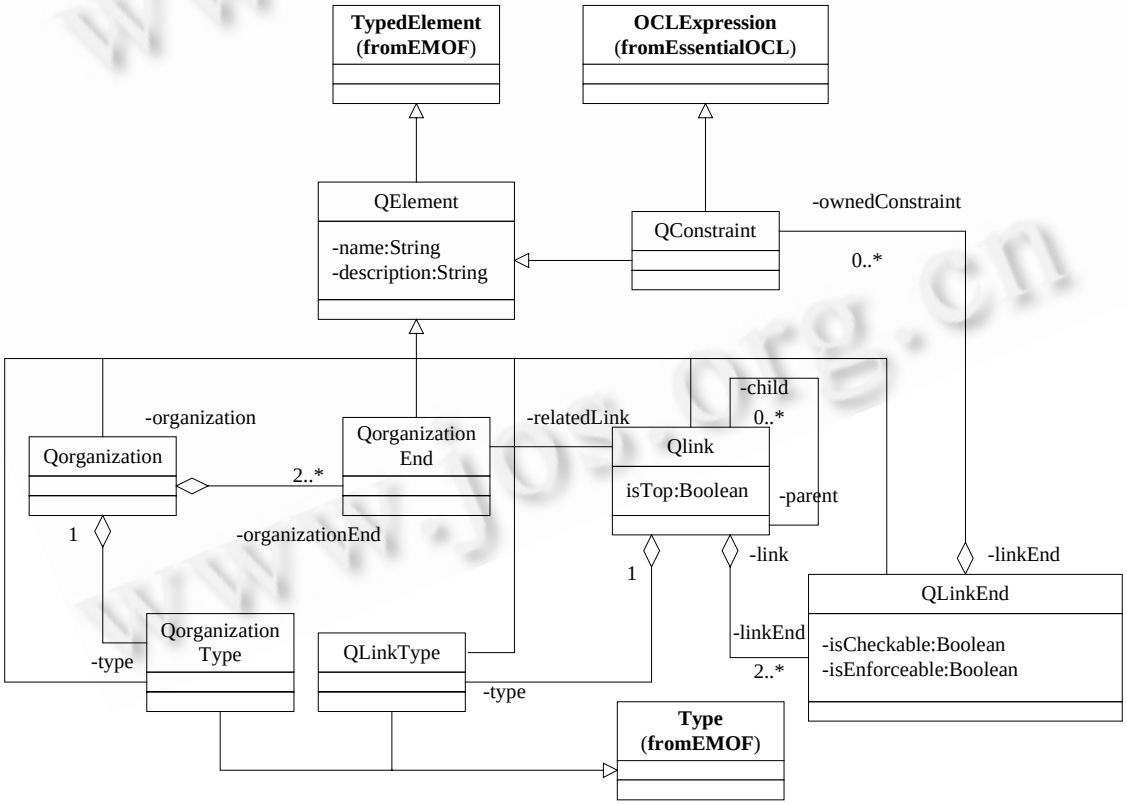

Fig.4 The part 2 of metamodel diagram of QMTW

图 4 QMTW 元模型图第 2 部分

2.3 QMTW的具体语法

QMTW是一个中间语言,最终会被转换为可执行的模型转换语言,所以没有必要以BNF范式的方式来定义其具体语法和关键字,而是定义了其元模型的XMI格式以及一个XML Schema***.符合此Schema的XML文件可以用来作为从编织模型到模型转换代码的输入.由于目前QMTW的工具尚未实现,所以并没有定义QMTW的图形符号.

3 一个QMTW的实例

本节介绍了一个应用QMTW进行模型转换编织的实例.源模型是一个Book类,它含有title属性和Chapter类,每个Chapter又包含title, nPages和author属性.目标模型是一个Publication类,包含title,nPages和author属性.它们之间的转换规则如下:

规则1. Book的title不变的转换为Publication的title;

规则2. Publication的nPages属性为Book中全部Chapter的nPages值相加所得;

规则3. Publication的author属性为Book中全部Chapter的author值相连,并除去重复值,然后在中间加上空格所得.

规则2中的约束:Book中的Chapter对象的数目不能少于2.

下面用QMTW对这个转换建立编织模型,为了清晰起见,编织模型被拆分在图5和图6两个图中,并作了适当简化.

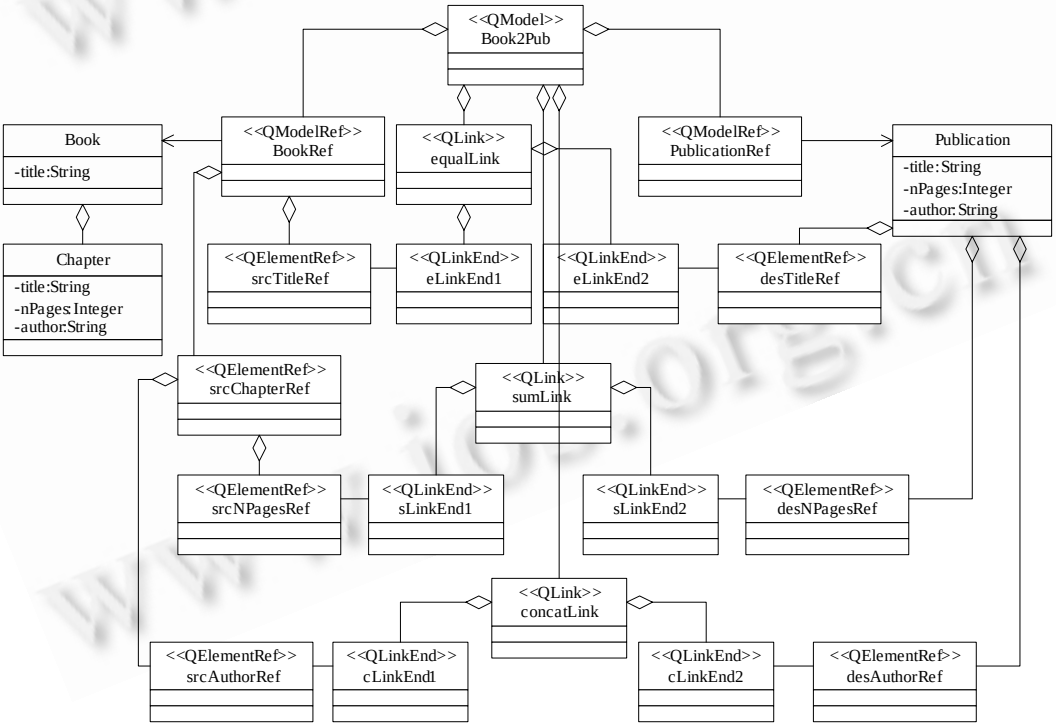

Fig.5 Book2Publication weaving model instance diagram—structure

图5 Book2Publication编织模型实例图——整体结构

***由于篇幅所限,本文没有详细列出XMI以及XML Schema.其代码可以在http://www.gfkd.org/blog/UploadFiles/2006-3/322549035.rar下载.元模型的XMI使用MagicDraw10.5来定义并去掉了其扩展部分.XML Schema使用hyperModel0.2生成并手动做了修改.

图 5 中 Book 和 Chapter 类是源模型,Publication 类是目标模型.编织模型中的连接使用 QLink 的实例来表示,连接端为 QLinkEnd 的实例.可以看到:规则 1 使用 equalLink 来表达;规则 2 和规则 3 分别使用 sumLink 和 concatLink 来表达.

图 6 描述的是编织模型的约束和组织部分.从图中可以看出:sumLink 的一个连接端 sLinkEnd1 带有一个约束 sLinkEnd1Constraint,此约束使用 OCL 进行了描述.sumLink 和 concatLink 之间存在依赖的关系,其语义是只有在 sumLink 已经执行的情况下 concatLink 才能执行,因此,使用一个组织的实例 prevOrganization 将这两个连接组织起来.

从这个编织模型的实例中可以看出:它完整地表达了整个模型转换算法的规则以及约束,并规定了整个转换执行的顺序,确定了此转换过程可以精确地执行.但是,QMTW 本身是不可运行的,要在生成目标代码以后才能够以目标代码的形式运行.下一节将介绍目标代码的生成.

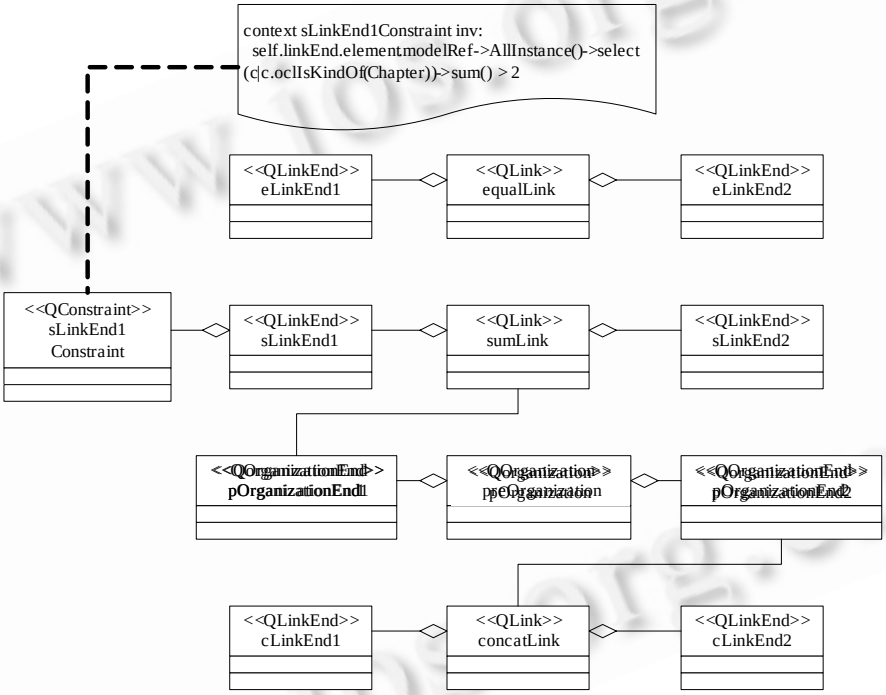

Fig.6 Book2Publication weaving model instance diagram—Link and organization

图 6 Book2Publication 编织模型实例图——连接与组织

4 从 QMTW 到目标语言的转换

QMTW 可以完整、精确地表达一个编织模型,即一组模型转换规则.但是,模型转换还是需要具体的模型转换语言来执行的.为了达到此目的,本文定义了从 QMTW 到 QVT Relation 语言之间的映射.为了适应更多的模型转换语言,可以定义从 QMTW 到其他模型转换语言的算法.因为其他语言,例如 ATL 和 MTF 的语义都弱于 Relation,所以,以它们作为目标语言的转换会更加简洁.

4.1 QMTW到Relation的转换定义

从 QMTW 到 Relation 的转换也属于模型转换的一个实例,因此也可以使用一个编织模型来定义.为了清晰起见,仅仅给出用自然语言描述的转换规则:

规则 1. QModel 转换为 Transformation,表示整个转换的定义;

规则 2. QModelRef 转换为 TypedModel,表示被转换的模型;

规则 3. QLink 转换为 Relation,表示转换规则或者映射;

规则 4. QElementRef 转换为 Domain,表示一个转换规则中涉及的模型元素;

规则 5. QConstraint 转换为 Pattern 中的一个断言 Predicate,表示一个转换规则中的约束;

规则 6. 所有转换后的 Relation 以 QOrganization 定义的关系来组织.

说明:Relation 中的 RelationImplementation 在 QMTW 中没有对应的元素,因为 QMTW 是基于声明性的转换,不涉及命令式的转换规则.

4.2 连接类型和代码模板

上面的实例涉及了 3 种类型的连接,分别是 equal,sum 和 concat. equal 的语义是各个连接端的模型元素的值相等,其代码模板如下:

```
$isTop relation $linkName //equle link template. A link is mapped to a relation, represent a transformation rule.
{
$var: $varType;
$linkEnd_isCheckable_end1 $linkEnd_isEnforceable_end1      //A linkEnd is mapped to a domain.
   domain $linkend_element_modelRef_end1:$modelType1 {$elment_attribute=$var}; //domain can access
$linkEnd_isCheckable_end1 $linkEnd_isEnforceable_end2                          //source or target model.
   domain $linkend_element_modelRef_end2:$modelType2 {$elment_attribute=$var}; //the source and target
}                                                                  // element attribute both equal to $var.
```

说明:以$符号开头的变量都是模板中需要替换的参数,其含义可以从名称中看出来.另外,所有的连接端都有一个接口用来生成编织模型中的约束代码,为了清晰起见,省略了这个接口.

sum 的语义是某个连接端的模型元素是一个集合,这个集合中所有的值相加,等于另一个连接端元素的值,其代码模板如下:

```
$isTop relation $linkName                                  //sum link template
{
$varName: $varType;
$linkEnd_isCheckable_end1 $linkEnd_isEnforceable_end1      //checkable means the domain could be check,
   domain $linkend_element_modelRef_end1:$modelType1 {     //while enforceable means it could be change.
      self->collect(a|a.$attribute)->iterate(i; acc:    $varType=0|    //the OCL expression could be executed to
         acc+i)=$var                                          //compute the sum of all attribute values.
};
$linkEnd_isCheckable_end2 $linkEnd_isEnforceable_end2
domain $linkend_element_modelRef_end2:$modelType2 {$elment_attribute=$var};
}
```

concat 连接的语义是将某个连接端的模型元素组成一个集合,过滤掉所有相同的值,然后用空格将这些值串联起来,等于另一个连接端模型元素的值,其代码模板如下:

```
$isTop relation $linkName //concat link template
{
$varName: $varType;
$linkEnd_isCheckable_end1 $linkEnd_isEnforceable_end2
   domain $linkend_element_modelRef_end1:$modelType1 {
      self->collect(a|a.$attribute)->asSet->iterate(i; acc:    $varType=''|//the OCL expression put all attribute
         acc+                                                  //values in a set, which could remove
         if acc='    '                                         //all repeat value. Then concatenate
```

```
            then I                                    //these values with a blank between
            else ‘   ’+I                              //every two of them.
          endif)
        = $var};
    $linkEnd_isCheckable_end1 $linkEnd_isEnforceable_end2
      domain $linkend_element_modelRef_end2:$modelType2 {$elment_attribute=$var};
    }
```

本文只使用了这 3 个连接类型.连接类型及其代码模板以库的方式存储.用户可以自己扩展这个连接类型库,以表达更加丰富的连接类型.

4.3 组织类型和代码模板

组织的类型包括依赖(先决、后继)、包含等.其语义表明了组织中各个连接之间的关系,根据组织的关系可以对不同的目标语义定义代码模板,这些模板的定义是比较直观和简洁的,由于篇幅所限,文中省略.本文只使用了先决组织.

4.4 转换步骤

在转换定义、连接类型与代码模板以及组织类型与代码模板这 3 个条件都具备以后,就可以进行从 QMTW 到目标语言的映射了.转换步骤如下:

1. 从 QModel 中得到 Transformation 的各种属性,并从 QModelRef 中得到源模型和目标模型;
2. 根据 Qlink,QlinkEnd,QConstraint 以及目标语言的代码模板来具体生成每一条规则;
3. 根据 QOrganization 的实例来修改规则中的调用与约束代码,体现连接间的依赖与包含等关系,并将所有规则组合成一个完整的程序;
4. 最后,利用目标语言的解释器/编译器对生成的目标语言进行解释或者编译,检查目标代码的正确性.

最后得到的 Relation 代码如下:

```
transformation Book2Publication (book:Book, pub:Publication)
{
  top relation Book_equal_Publication
  {
    var:String;
    checkonly domain book:Book{title=var};
    enforce domain pub:Publication{title=var};
  }
  top relation Book_sum_Publication
  {
    var:Integer;
    checkonly domain book:Book{
      self->collect(chapter.nPages)->iterate(i; acc: Integer=0|acc+i)=var and
      self.chapter->AllInstance()->sum()>2};
    enforce domain pub:Publication{nPages=var};
    where{
        Book_concat_Publication(book,pub);
    }
}
```

```
relation Book_concat_Publication
{
    var:Integer;
    checkonly domain book:Book{
      self->collect(chapter.author)->asset()->iterate(i; acc: String=''|
        acc+
        if i=''
        then author
        else i)
      = var};
    enforce domain pub:Publication{author=var};
  }
}
```

同上述方法一样,还可以定义从 QMTW 到其他模型转换语言之间的转换.需要注意的是,如果编织模型定义了超出目标语言功能的规则,则无法生成正确的目标语言.

5 相关工作

目前,国际上 MDA 研究的热点集中在模型转换技术的发展和应用上.在模型转换概念出现之初,各种各样的模型转换方法不断涌现,它们各自致力于表达模型转换技术中的一个侧重点.文献[12]强调用集合以及关系代数的方法来描述模型转换的结构;文献[13]强调能够在一个模型转换方法中描述双向的转换规则,让模型转换可以双向执行;文献[3]强调了如何更有效地执行模型转换;在文献[14]中,将图转换技术和产生式编程应用到模型转换过程中,定义了一种模型转换语言 Genermorphous.它将图形转换通过一组产生式规则来定义,每个产生式规则包括一个 LHS 图形和一个 RHS 图形.如果 LHS 图形匹配了源模型中的图形,那么它被规则中的 RHS 所代替,在对源模型进行匹配和重写的过程中逐步生成目标模型;文献[15]中也描述了一种将图形转换和重写技术应用到模型转换过程中的方法,并构造了一个图形重写和转换系统 GReAT(graph rewriting and transformation),其中包含一种基于图重写技术的模型转换语言 UMT.

但是,上述种种模型转换方法的异构性反而给自身的发展造成了障碍.为了统一模型转换技术,OMG 推出了 QVT[7]标准,但它还有待成熟和完善,目前还不是事实上的标准.况且,遗留的模型转换代码不能被简单地丢弃.简单地将目前的一些模型转换语言与 QVT 结合起来困难很大[16].在 AOP 的启发下,出现了模型编织方法.目前的模型编织方法有两种:一种类似于 AOP 方法,它给出了模型转换中的 aspect,pointcut 和 join point,使用类似于 AOL 的语言来描述模型转换规则,例如文献[4,17];另一种提高了模型转换的层次,给出了一个模型转换的元模型,此元模型的实例称为编织模型.一个编织模型代表了一组模型转换规则,它可以被转换为不同的模型转换目标语言,从而部分地消除了模型转换语言的异构性[8,9].本文提出的方法对第 2 种模型编织方法作了扩充,还定义了一种到 QVT Relation 语言的映射,具有符合标准、简单易用等优点.

国内的 MDA 研究也逐渐成为热点,研究者在以下几个方向做出了努力:文献[18]对 MDA 技术作了一个整体的介绍,并讨论了 MDA 技术中有待研究的 3 个问题,为其他研究者作了一个很好的综述;文献[19]描述了一个具体的 MDA 工具的架构,并实现了该工具,这个工作在 MDA 的建模领域做出了贡献;文献[20]将模型转换技术运用到了某个具体的领域,并用形式化的语言描述了其算法,此工作属于模型转换的应用;文献[21]对 MDA 技术中从 CIM 到 PIM 这个问题进行了表述,提出了一种面向特征的、基于组件的 CIM-PIM 转换方法,对转换中的可跟踪性以及 PIM 的形成问题做出了解答.这个工作关注了模型转换中很少涉及的 CIM-PIM 转换,填补了这个领域的空白.本文提出的 QMTW 是在模型转换的更高层次上力求统一模型转换的方法,消除模型转换技术的异构性,促进 QVT 规范的统一和模型转换技术的广泛运用.

6 未来的工作和结论

未来的工作包括以下几个方面:1) 继续完善 QMTW 的抽象语法和具体语法,使其更加精确地与 QVT 规范结合,并随着 QVT 规范的改进而变动;2) 扩充连接和组织类型库,使其能够表达更加复杂的规则与规则间的关系;3) 构造基于 QMTW 的模型编织工具.

本文提出了一种基于 QVT 规范的模型转换编织架构 QMTW,给出了其概念、抽象语法和语义、具体语法、使用实例以及从 QMTW 编织模型到 Relation 语言的转换.QMTW 的主要目的是为了消除模型转换技术中的异构性,同时可以让使用者掌握模型转换技术的实质,降低模型转换的技术门槛和使用难度.其优点包括:1) 使用简单.因为 QMTW 将模型转换的实质以最简单的方式表达了出来,所以学习和使用 QMTW 的难度要低于学习某种具体的模型转换语言.2) 符合规范.QMTW 是完全符合 QVT 标准的,它也可以作为一种模型转换算法的标准表达形式来进行交流.3) 使用范围广.因为 QMTW 的最基本元素连接和组合都是可以扩展的,因此使用者可以按照自己的需要扩展 QMTW 使其满足自己的需要,从而在各个领域应用这种方法.QMTW 的缺点一是难以描述命令式的模型转换规则,对于非规则型的模型转换描述能力不强;二是描述复杂转换规则的能力不够,因为此类规则难以映射到关系型的转换方法中.

总而言之,与 QVT 规范相比,QMTW 更简单,更容易学习和使用.与其他模型编织方法相比,QMTW 语义丰富,且符合规范.与具体模型转换语言相比,QMTW 可以通过扩展自己的目标语言来完成到这种模型转换语言的映射,从而保护其遗留代码.

致谢 感谢在此工作中给予我们帮助和支持的同行,包括北京大学的马浩海同学,南京大学的于笑丰同学等.

References:

[1] OMG. MDA guide version 1.0.1. 2003. http://www.omg.org/cgi-bin/apps/doc?formal/03-06-01.pdf

[2] Sun CA, Jin MZ, Liu C. Overviews on software architecture research. Journal of Software, 2002,13(7):1228−1237. (in Chinese with English abstract) http://www.jos.org.cn/1000-9825/13/1228.htm

[3] Jouault F, Kurtev I. Transforming models with ATL. In: Bruel JM, ed. Proc. of the Model Trans. in Practice Workshop at MoDELS. LNCS 3844, Jamaica: Springer-Verlag, 2005.

[4] Silaghi R, Fondement F, Strohmeier A. “Weaving” MTL model transformations. In: Proc. of the 2nd Int’l Workshop on Model Driven Architecture, Foundations and Applications (MDAFA). Sweden: Springer-Verlag 2004. 123–138.

[5] OMG. MOF 2.0 query/views/transformations RFP. OMG document ad/2002-04-10, 2002. http://www.omg.org/docs/ad/02-04-10.pdf

[6] OMG. A review of OMG MOF 2.0 query/views/transformations submissions and recommendations towards the final standard. OMG Document: ad/03-08-02, 2003. http://www.omg.org/docs/ad/03-08-02.pdf

[7] OMG. Revised submission for MOF 2.0 query/view/transformations RFP (ad/2002-04-10). OMG Document ad/2005-07-01, 2005. http://www.omg.org/docs/ad/05-07-01.pdf

[8] Marcos DDF, Jean B, Frédéric J, Erwan B, Guillaume G. AMW: A generic model weaver. In: Proc. of the 1ères Journées sur l'Ingénierie Dirigée par les Modèles. 2005. http://www.sciences.univ-nantes.fr/lina/atl/www/ papers/IDM_2005_weaver.pdf

[9] Fabro MDD, Jouault F. Model transformation and weaving in the AMMA platform. In: Proc. of the Generative and Trans. Techniques in Software Engineering (GTTSE 2005). 2005. 71–77. http://www.sciences.univ-nantes.fr/lina/atl/www/papers/CR_2005_GTTSE_transfo_weaving.pdf

[10] OMG. Meta object facility (MOF) specification, version 1.4. OMG Document formal/2002-04-03, 2002. http://www.omg.org/docs/formal/02-04-03.pdf

[11] OMG. UML 2.0 OCL specification. OMG Document ptc/03-10-14, 2003. http://www.omg.org/docs/ptc/03-10-14.pdf

[12] David K, Stuart A. A relational approach to defining transformations in a metamodel. In: Hussmann H, Jézéquel JM, Cook S, eds. UML 2002—The Unified Modeling Language 5th Int’l Conf. LNCS 2460, Dresden: Springer-Verlag, 2002, 243–258.

[13] Braun P, Marschall F. BOTL the Bidirectional Object Oriented Transformation Language. TUM-I0307: Technische Universität München, 2003.

[14] Shane S. Combining generative and graph transformation techniques for model transformation: An effective alliance? In: Proc. of the 2nd OOPSLA Workshop on Generative Techniques in the context of Model Driven Architecture. Anaheim: ACM Press, 2003. http://cui.unige.ch/~sendall/files/sendall-mda-workshop-OOPSLA03.pdf

[15] Karsai G, Agrawal A. Graph transformations in OMG's model-driven architecture. In: LNCS vol. on Applications of Graph Trans. with Industrial Relevance. Springer-Verlag, 2003.

[16] Jouault F, Kurtev I. On the architectural alignment of ATL and QVT. In: Proc. of the ACM Symp. on Applied Computing (SAC 2006). Dijon: ACM Press, 2006.

[17] Milewski M, Roberts G. The model weaving description language (MWDL)—Towards a formal aspect oriented language for MDA model transformations. In: Proc. of the 1st Workshop on Models and Aspects—Handling Crosscutting Concerns in MDSD At the 19th European Conf. on Object-Oriented Programming. 2005. http://www.st.informatik.tu-darmstadt.de:8080/ecoop2005/maw/acceptedPapers/Milewski.doc

[18] Jiang YB, Xing CX. The state of art and roadmap of MDA. Journal of Nanjing University, 2005,41(10):360−366 (in Chinese with English abstract).

[19] Ma ZY, Ma HH, Zhang NB, Lao ZP, Zhu ZG. Development of the software development platform based on UML2.0. Journal of Nanjing University, 2005,41(10):374−381 (in Chinese with English abstract).

[20] Wang XB, Wu QY, Wang HM, Shi DX. A relational model transformation approach between UML metamodel and SQL metamodel. Journal of Nanjing University, 2005,41(10):347−352 (in Chinese with English abstract).

[21] Zhang W, Mei H, Zhao HY, Yang J. Transformation from CIM to PIM: A feature-oriented component-based approach. In: Williams L, Clay B, eds. Proc. of the MoDELS. LNCS 3713, Jamaica: Springer-Verlag. 2005. 248−263.

附中文参考文献:

[2] 孙昌爱,金茂忠,刘超.软件体系结构研究综述.软件学报,2002,13(7):1228−1237. http://www.jos.org.cn/1000-9825/13/1228.htm

[18] 蒋炎冰,邢春晓.模型驱动的体系结构研究综述.南京大学学报,2005,41(10):360−366.

[19] 麻志毅,马浩海,张能宾,劳卓鹏,朱志高.一个基于 UML2.0 的软件开发平台的研制.南京大学学报,2005,41(10):374−381.

[20] 王学斌,吴泉源,王怀民,史殿习.南京大学学报,2005,41(10):347−352.

王学斌(1978-),男,湖北云梦人,博士生,主要研究领域为软件工程,模型驱动架构.

吴泉源(1942-),男,教授,博士生导师,主要研究领域为智能软件,分布计算.

王怀民(1962-),男,博士,教授,博士生导师,CCF 高级会员,主要研究领域为分布式计算,人工智能,Agent 计算.

史殿习(1966-),男,博士,副研究员,主要研究领域为分布式计算.