# MDA 中的模型转换技术综述*)

张德芬　李师贤　古思山
(中山大学信息科学与技术学院　广州 510275)

**摘　要**　模型转换是模型驱动软件开发的关键技术。本文首先对 MDA 中的模型转换技术进行了分类，然后介绍了模型转换技术的研究现状，并对当前支持模型转换的 MDA 工具作了总结和比较，最后展望了模型转换技术的发展前景。

**关键词**　模型转换技术，MDA，MDA 工具

## Overview of Model Transformation Technology in MDA

ZHANG De-Fen　LI Shi-Xian　GU Si-Shan
(School of Information Science and Technology, Sun Yat-sen University, Guangzhou 510275)

**Abstract**　Model transformation is the key to model-driven software development. First this paper gives a classification for model transformation technology in MDA. Then the paper surveys the research works on model transformation technology. After that the paper summarizes the current MDA tools, which support the model transformation, and makes some comparison between them. Finally the future direction of model transformation technology is presented in the paper.

**Keywords**　Model transformation technology, MDA, MDA tools

## 1　MDA 概述

模型驱动体系架构 MDA(Model Driven Architecture)是 2001 年 7 月对象管理组织 OMG(Object Management Group)发布的一个软件开发框架。OMG 提出 MDA 的目的是应对不断变化的软件开发需求和软件技术，保护企业的 IT 投入。MDA 使模型成为软件开发中的“一等公民”[1]，模型不仅仅是程序员的文档，对软件系统的建模行为将直接驱动系统开发过程[2]。

MDA 中对模型有不同的分类方法，如业务模型和系统模型、逻辑模型和物理模型、需求模型和计算模型等[3]。PIM(Platform Independent Model)和 PSM(Platform Specific Model)是 MDA 中用于开发工作的计算模型，正是这两种模型使 MDA 方法和其他仅将模型作为设计工件的方法区分开来[3]。PIM 是独立于平台的计算模型，记录了业务知识及与具体实现平台无关的设计决定[3]；PSM 中描述了与实现平台相关的技术决定。两种模型处于不同的抽象层。PIM 位于抽象层次较高的业务层，PSM 位于相对具体的实现层。

MDA 的主要思想是将业务规约与实现技术分离，为应用系统建立平台无关的模型 PIM，以便更好地适应业务和技术的变化。通过建立针对实现平台的映射，PIM 可以自动生成特定技术平台的模型 PSM；通过建立针对实现语言的映射，PSM 可以自动生成代码，从而实现最终的系统。

MDA 开发过程由一系列的模型映射组成，这些映射将最初的 PIM 转换成最终的 PSM，这些 PSM 精确到可以直接翻译成可执行程序[4]。

## 2　MDA 中的模型转换技术分类

模型转换是模型驱动软件开发的关键技术[5]。MDA 中的模型转换技术就是 PIM、PSM 和代码之间相互演化的技术。目前 MDA 中的模型转换技术多种多样，对模型转换技术的分类也有很多种。

### 2.1　模型到模型的转换和模型到代码的转换

根据 MDA 开发过程中模型的演化结果，可以分为模型到模型的转换和模型到代码的转换。

MDA 中有四类模型到模型的转换：PIM 到 PIM 的转换，指平台无关模型的不断改进；PIM 到 PSM 的转换，指 PIM 转换到针对目标实现平台的设计模型；PSM 到 PSM 的转换，指平台相关模型的设计优化；PSM 到 PIM 的转换，指从已存在的实现中抽象出平台无关模型，是 MDA 中的逆向工程。

MDA 中模型到代码的转换也有四种：PSM 到代码的转换，指从平台相关模型生成可执行代码；PIM 到代码的转换，指从平台无关模型直接生成可执行代码；代码到 PSM 和代码到 PIM 的转换是 MDA 中的逆向工程，前者指从可执行代码中抽取出平台相关模型，后者则从可执行代码中抽取出平台无关模型。

当为代码所采用的程序语言提供一个元模型时，可以把模型到代码的转换视作模型到模型转换的特殊情况[6]。

### 2.2　模型重构和模型求精

根据变换前后模型抽象程度的变化，MDA 中的模型转换可以分为模型重构和模型求精。

模型重构是从代码重构发展而来的概念，是在设计级的软件重构。模型重构旨在改变软件内部结构，提高某些软件质量特性(如可理解性、可修改性、可重用性、模块性、适应性)，而不丢失可观察到的行为[7,8]。

模型求精是指通过不断增加更具体的细节，使模型逐步完善，更接近成熟的实现。求精是贯穿面向对象技术的开发

*)基金项目：本文得到广东省科技计划工业攻关项目(编号：2003A1030403)资助。**张德芬**　博士研究生，高级工程师，研究方向：软件工程。**李师贤**　博士生导师，教授，研究方向：软件工程。

过程中的一种重要思想，同样，模型求精也是MDA中的主要转换类型。通过模型求精，系统模型从抽象的视图转换成更接近真实实现的模型。

从源模型到目标模型的变化分析，模型重构是一种水平转换，变换前后的模型保持了同样的抽象级别。而模型求精则是一种垂直转换，变换前后的模型可以在不同的抽象级别[7]。

### 2.3 单向转换和双向转换

转换可以是单向的或双向的。单向转换仅能在一个方向执行，双向转换则可实现正逆双向执行。从设计到代码的转换是正向转换，如PIM到PSM、PSM到代码；反之，称为逆向转换或逆向工程。正向转换增加了语义信息，而逆向转换则从低级别规范中提取高级别的规范[7]。双向转换在模型间的上下文同步中很有用。双向转换能用双向转换规则得到，或者通过定义两个单独的互补的单向规则得到，每个规则用于一个方向[6]。

### 2.4 基于图的转换和基于文本的转换

根据转换过程中模型的表示方式可以分为基于图的转换和基于文本的转换。基于图的转换是指转换过程中模型始终是以UML图的方式存在，转换前后模型的表示方式未变。基于文本的转换是指转换过程中模型先转换成文本方式表示，再根据文本表示的模型进行转换处理。

## 3 模型转换技术的研究现状

### 3.1 基于XMI的模型转换技术

XMI(XML Model Interchange)是OMG组织提出的元模型交换标准。它通过标准化的XML文档格式和DTDs(Document Type Definitions)为UML元模型和其他模型定义了一种基于XML的数据交换格式。它同时也定义了一个从UML到XML的映射。XMI的主要作用就是能够利用XML来连接不同的分析模型和设计模型。XML、MOF、UML和OCL标准很好地集成在XMI中，共同组成了一个强大的模型序列化工具[1]。

XMI是文本表示的，基于XMI的模型转换技术也就是基于文本的转换技术，其主要优点是相对简单，缺点是不够直观，并且转换步骤较多，开销较大，容易引起前后模型的不一致性[9]。由于XSLT和XML耦合得不紧，即使一个简单的模型转换也需要花费相当的精力去构造XSLT。同时，基于XMI的转换常常是通过批处理方式进行的，无法在转换时和用户进行对话，用户无法定义转换的参数和控制转换进程[10]。

### 3.2 基于图文法的转换

将UML模型视为图，利用形式化的图文法(graph grammars)进行转换，这是一类基于图的转换方法[11]。其中，UML模型被视作图，用节点表示组件单元，用边对组件间的关系建模。应用图重写规则(graph rewriting rules)进行图转换。规则由一个待匹配的图和一个替换图组成，前者常称为左手边(LHS, left-hand side)，后者常称为右手边(RHS, right-hand side)。当为LHS图发现了匹配时，应用相应的图重写规则进行转换，结果是RHS图置换了相匹配的LHS图[10]。

图文法中对象不得不分解成相当细颗粒的子图，这将导致非常庞大而混乱的模型，大大增加转换的复杂性。而且这种方法没有表示出对于面向对象相当重要的一组特性，没有标识对象身份的概念，如图文法不知道元模型定义了有效的对象模型结构。因此难以用这种方法转换属性的值，如用算法将华氏温度值转换成摄氏温度值比图文法容易得多[9]。

基于图文法的转换技术中，整个转换的分析过程及算法非常复杂，要经过相当的培训和学习才能掌握这种方法。

### 3.3 形式化的UML类图转换语言

文[9]提出一种形式化的UML类图的转换语言BOTL(the Basic Object-oriented Transformation)，试图在基于文本的转换和基于图文法的转换方法之间找到一个平衡点。UML类图是软件工程领域的主要建模手段，BOTL给出了形式化的基于UML类图的映射规则，用于描述UML类图表示的模型之间的匹配。BOTL语言既保留了图文法的直观性优点，又具有形式化语言的精确性的优点。BOTL下一步的研究目标是实现双向变换以及开发支持BOTL转换的工具。

仅能描述UML类图模型的转换，是BOTL方法的局限性。但这种方法为模型转换技术的发展提供了一个很好的研究思路。

### 3.4 基于MDA规范的模型转换技术

#### 3.4.1 利用可执行UML和UML ASL(action semantics language)进行模型转换

UML动作语义可以直接处理UML结构，和OCL规则结合在一起编写前置和后置条件，适用于模型重构[8]。

可执行UML和动作语义的结合可用于模型到代码的转换[12]。可执行UML对系统建立PIM模型，利用ASL建立各种领域PIM之间的映射和转换关系。可执行UML模型可直接转换成可执行代码，从而在开发过程的分析阶段通过域模型建立系统的原型框架，有助于在设计早期获得用户对系统的反馈，延迟设计决定，降低开发成本。

但是由于动作语义的概念是在2GL编程序语言指令级上定义的[13]，更像UML汇编语言，动作表达式常常相当复杂，开发人员难以阅读。由于UML动作语义规范仅定义了一个语言的元模型，而对具体的语法没有任何限制，缺乏一个统一的动作语言(Action Language)来描述转换，许多UML CASE采用自己的动作语言，造成工具间的协同性不好。并且，开发人员除了掌握UML外，还要学习工具使用的动作语言[8]。

#### 3.4.2 利用MOF、OCL和CWM进行模型转换

文[5]利用MOF、CWM及其对象约束语言OCL来描述和规定模型转换。首先是定义元模型级的源模型和目标模型的转换制式(transformation scheme)，转换制式本身也是一个UML模型，是通过扩展CWM概念来定义的。转换制式的实例是转换实现(Transformation implementation)，描述了源模型实例和目标模型实例的关系。在模型实例级，转换制式通过生成XSLT或嵌入在一个CASE工具中成为转换实现，将源模型转换成目标模型，如图1[5]。在源模型和目标模型的关联端，用OCL描述了各种classifier的映射规则。

UML、MOF、CWM、OCL等是MDA的核心技术，用MOF、OCL和CWM描述模型转换的优点是能用同一种语言来描述对系统的建模和不同抽象层次的模型之间的关系，避免了为建立不同层次模型之间的关系而去学习和掌握一门新语言的情况。同时，用UML描述的元模型的转换制式，定义了源元模型和目标元模型间的模型元素的关联，能用于管理模型实例间的关系，比如验证模型变换前后的有效性和一致性等。缺点是对于模型到代码的转换尚不能很好地描述[5]。

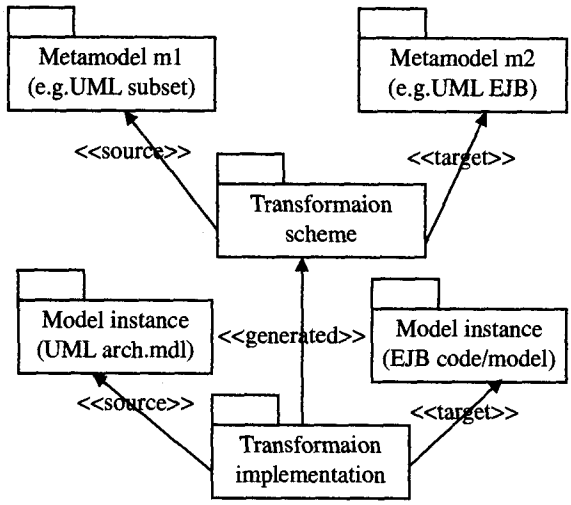


图 1　元模型、模型实例和转换制式、转换实现

从以上的模型转换技术来看，大部分转换技术是基于语法规范的，许多技术采用了文本语法，还有一些采用了图形文法，有的则结合了两种语法规范。另外一类转换技术是利用MDA本身的规范进行转换。但是现有的模型转换技术普遍存在的问题是对于转换前后模型的完备性和一致性缺乏有力的保证，另外模型转换的双向性和交互性有待改进。

## 4　MDA 工具

MDA引起了市场的积极响应，当前至少有40余种工具号称支持MDA，还不包括正在研制的MDA工具。这些工具往往涉及UML建模、PIM到PSM的转换和生成代码中的一个方面或全部。许多MDA工具侧重点不同，可以根据需要选择某些MDA工具，将不同的工具贯穿在一个完整的MDA开发过程中。

下面对常见的开源和商业MDA工具作一些简单介绍。

开源的MDA工具包括MOFScript，IBM的Model Transformation Framework(MTF)框架，ATL引擎，MTL引擎、Eclipse的GMT插件，Kent Modelling Framework(KMF)，AndroMDA，Middlegen等[14]。其中，MOFScript是一种源模型到文本的转换工具，它最初是作为一个Eclipse的插件，支持源模型的解析、检查和执行转换。2005年5月MOFScript作为MOF源模型到文本的转换过程提案提交给OMG。OpenMDA可以生成J2EE、.NET等目标模型的代码。AndroMDA是基于模板的J2EE代码生成工具，可以从UML模型生成EJB、Web Services等应用。

商业化的MDA工具包括：ArcStyle，MCC(ModelComponent Compiler)模型组件编译器，Codagen Architect，OptimalJ，Model-in-Action等。其中OptimalJ是Compuware公司推出的、针对J2EE环境的开发工具，号称是目前最先进的。OptimalJ使用的模型转换方式是Pattern Based，意即它使用了基于模式的转换方式。在OptimalJ中把模式分为两种：Transform Patterns和Functional Patterns。Transform Patterns用于在不同层次的模型之间的转换，分为Technology Patterns和Implementation Patterns两种，分别对应model-to-model和model-to-code这两个不同的层次。Technology Patterns负责把对业务逻辑的描述转换成用特定技术实现的模型，Implementation Patterns则把特定技术的模型转化为实际的代码。Functional Patterns则是用于在同一个层次的模型之间进行的转换。它的目的是增加可重用性，提高生产率，比如在Code Model层，有Code Pattern可以进行对代码的不同实现策略进行描述并实现，这也是对Implementation Patterns实现其可定制的重要基础。

综观目前的工具，尚未有一款工具能够覆盖MDA架构里面的所有需求，原因有以下两点：一是目前尚缺少标准的模型转换规范，即使是OMG的QVT，其最终标准还没有正式出台；二是要做到MDA所需求的覆盖全部领域的要求不太现实，特别考虑到商业成本的情况下更是如此。

## 5　模型转换技术的展望

### 5.1　模型编译器

模型转换技术是成功应用MDA的关键。从软件工程的角度看，MDA的一个直接目标是消灭软件开发中的编码工作，让更接近问题域的模型取代程序代码成为软件开发的重要内容和推动力。因此，类似第三代语言通过编译技术实现了全自动地将高级语言翻译为机器语言，模型转换技术的发展目标是将描述软件系统的模型自动翻译成第三代语言，因此模型转换技术又被称为模型编译器[6]。模型编译器将是模型转换技术的一个发展方向，未来的模型转换技术将会像当前的编译程序一样可靠而高效。软件开发人员将主要从事建模和设计工作，编码和实现主要由模型编译器和代码编译器执行，软件开发的抽象层次提高到了模型的设计层。已经颁布的UML2.0规范和正在制订的OMG模型转换规范QVT都将给模型转换技术注入新的内涵，并促进模型转换器最终成为类似现在3GL编译器的模型编译器。

### 5.2　面向服务的模型转换技术

MDA解决了应用系统对平台和业务演化的适应性。然而全球化和电子商务的飞速发展加速了企业之间的竞争与合作，单点的企业应用已不再能满足不断扩大和变化的业务需求。面向服务的体系结构(Service-Oriented Architecture，SOA)成为企业应用的新型解决方案，用于应对应用系统和IT环境的复杂性，创建一个灵活、适应性强的IT基础设施。

SOA与MDA是正交的技术方向，SOA在更为宏观的层面定义了系统间的无缝互联，但它对于系统本身从高层次的体系结构到低端的运行代码的构建过程没有提供方法论的指导，而MDA填补了空缺，MDA提供了应用系统从高层抽象到代码执行级的开发框架。SOA是目的，MDA是手段。

在抽象层上，服务位于业务和技术之间。SOA通过抽象出服务来隐藏提供这些服务的底层技术。服务消费者不必了解服务的实现技术，而服务提供者也可以在保持服务的前提下改变服务的实现技术。MDA开发方法通过建立平台无关模型PIM和平台相关模型PSM分离了业务和技术的关注点。通过调整PIM，可以应对不断变化的业务需求，而保持PSM的相对稳定；而技术平台的改变可以通过生成新的PSM来实现，而PIM不变。因此，MDA和SOA对系统的抽象思想都是殊途同归的，都旨在增强系统对于业务和技术变化的适应性。

通过MDA可以建立平台无关的服务模型，在服务模型中记录服务提供的业务功能，通过模型映射，将平台无关的服务模型转换成平台相关的功能模型、数据模型，集成到SOA基础设施上。基于SOA，模型的粒度和精化与服务的封装密切相关。模型转换技术将在服务模型的精化和服务的封装中

(下转第290页)

退出时，就会删除相应的在线用户记录。这样，从服务器端保证了即使有人取得已退出用户的 GUID，也不能冒充那个用户。

## 5 文档管理的实现

文档管理系统将所有的文档信息分为两部分：一部分称为元数据，即文档的描述信息，如文档名称、类型、创建日期、版本、所有者以及存取的相对路径等，保存在数据库表格中；另一部分即具体的物理文件，存放在磁盘上。从而建立起数据库表格，每一条记录与一个物理文件的连接和对应关系。

原则上每一种文档对应一个数据库表格，但是由于文档种类繁多，为了便于用计算机管理，对各种文档从总体上进行分类管理。这样，系统屏蔽了文档存储的实际物理位置。当 Web 用户进入系统界面单击某一文档对象时，便获得了操作该对象的指针，在对该对象进行签入、签出和复制等操作时，系统自动到相应的表格中进行记录的修改和增加，来跟踪文档信息的变化，把文档的元数据存入元数据库中，相应的具体文档则放入应用服务器中指定的文件系统的相关路径中。

### 5.1 文档的上传和下载

在 Web 客户端，文档签入实质进行的是文档上传操作。Web 用户选定要签入的电子仓库后，签入文档首先从 Web 客户端上传到 Web 服务器，再从 Web 服务器通过文件流保存到应用服务器中，并将元数据存入数据库保存。文档上传使用自定义的 Upload 组件实现。当客户端提交上载文件的 Form 表单时，Web 服务器就调用 Upload 来完成文件上传的请求。

### 5.2 Web 的文档管理

Web 的文档管理功能包括：

(1)文档的版本控制，如文档在检出、修订时版本的增加和扩展；

(2)文档的操作，包括文档的创建、注册、查询，文档的检入、检出、复制、删除，以及文档的冻结、修订等；

(3)文档的安全控制，通过用户注册登录口令认证以及文档操作过程中角色权限认证确保文档管理的安全性；

(4)消息系统，通过系统内建的 JMS 同步和异步消息，实现文档管理的协同工作；

(5)邮件系统，便于系统用户进行信息交流。

### 5.3 Dynamic HTML 技术

经典的 Web 站点所发布的页面都是静态的。而目前的动态 HTML，把静态的属性标签增强成可以动态访问和修改的对象。

### 5.4 动态页面发布

在文档管理系统中，根据实际情况采用了微软的 ASP 和 ISAPI 技术。

文档管理系统在所有涉及数据库操作的 ASP 动态页面中使用了 ADO 对象。而且，在 ADO 的数据源中，指定使用 SQL OLEDB Engine For SQL Server。实际的测试表明，这样实现虽然在与数据库建立连接时需要稍多的开销，但进行数据查询和数据修改操作的效率却令人满意。

对数据库操作进行优化，对某些常用的联系紧密的数据库操作，编写数据库存储过程，像函数一样通过 ADO 对象调用。这样便于 SQL Server 对数据操作进行优化，减少数据在 DCOM 对象间传递的次数，保证数据库操作的高效性。

**结束语** Web 文档管理系统通过对公司的人员分组，使得开发过程中的软件开发人员和数据处理人员能够通过消息及时沟通，保持数据同步，较好的协同工作，集中管理，分散存储也利于数据的备份和恢复，保证了数据的安全性。

通过 Java 语言良好的跨平台特性和面向对象的设计方法使基于 Web 的文档管理系统具有强大的可移值性和较少的代码量。对于打破部门之间空间和时间形成的边界，更好地管理各部门的协同具有十分重要的实用价值。

## 参 考 文 献


1 刘超. 可视化面向对象建模技术-标准建模语言 UML. 北京航空航天大学出版社，1999

2 [美]Roger S. Pressman 著，梅宏，译. 软件工程实践者的研究方法(原书第 5 版). 机械工业出版社，2003

3 [美]Gamma E，Helm R，Johnson R，et al. 设计模式：可复用面向对象软件的基础. 北京：机械工业出版社，2005

4 方浩，程传庆，王金亭. 基于 UML 的库存管理信息系统设计. 武汉理工大学学报(交通科学与工程版)，2004，28(8)：626～628

5 冯东雷，应刚，顾春华. Intranet 的原理与应用-企业信息系统的建设. 上海：上海交通大学出版社，1987


(上接第 230 页)

发挥重要作用。因此，SOA 为模型转换技术的发展提供了一个新方向——面向服务的模型转换技术，不仅能够实现 PIM 到 PSM 的自动转换以及 PIM 之间的精化，还能为业务和服务提供不同抽象层次、不同视点的模型转换，从而实现 SOA 中的服务。

## 参 考 文 献


1 Bézivin J. From Object Composition to Model Transformation with the MDA. In：Proceedings of TOOLS'USA，IEEE TOOLS-39，Santa Barbara，August 2001

2 Kleppe A，等著. 解析 MDA. 鲍志云译. 人民邮电出版社，2004

3 Frankel D S 著. 应用 MDA. 鲍志云译. 人民邮电出版社，2003

4 Caplat G，Sourrouille J-L. Model Mapping Using Formalism Extensions. IEEE Software. 2005. 44～51

5 Oldevik J，et al. Framework for model transformation and code generation. In：Proceedings of the Sixth International Enterprise Distributed Object Computing Conference (EDOC'02)，IEEE，2002

6 Czarnecki K，Helsen S. Classification of Model Transformation Approaches. OOPSLA'03 Workshop on Generative Techniques in the Context of Model-Driven Architecture

7 http://drops. dagstuhl. de/opus/volltexte/2005/11/pdf/04101. SWM2. Paper. pdf

8 Sunye G，et al. Using UML Action Semantics for Executable Modeling and Beyond，Springer-Verlag，2001. 433～447

9 Braun P，Marschall F. Transforming Object Oriented Models with BOTL. Electronic Notes in Theoretical Computer Science，2003，72(3)

10 Sendall S，et al. Model transformation：The heart and soul of Model-Driven Software Development. IEEE Software，2003. 42～45

11 Agrawal A. Graph Rewriting And Transformation (GReAT)：A Solution For The Model Integrated Computing (MIC) Bottleneck. 18th IEEE International Conference on Automated Software Engineering (ASE'03)

12 Raistrick C. Applying MDA and UML in the Development of a Healthcare System. LNCS 3297，Springer-Verlag，2005. 203～218

13 Duddy K，et al. Model Transformation：A declarative，reusable patterns approach. In：Proceedings of the Seventh IEEE International Enterprise Distributed Object Computing Conference (EDOC'03)

14 http://www. modelbased. net/mda_tools. html