模型驱动架构(MDA)相关技术研究与实现*)

徐晓钟

(上海师范大学数理信息学院计算机系　上海 200234)

摘　要　MDA的实现关键在于正确地建立模型和模型之间、模型和平台之间的关系，并准确地定义并实现不同模型之间的映射，本文探讨在MDA框架下，从元模型的构建、模型的映射到最终代码的生成一系列的实现技术。

关键词　MDA，J2EE，UML，模型

The Research and Implementation of Technology in MDA

XU Xiao-Zhong

(Department of Computer Science and Technology, Shanghai Teachers University, Shanghai 200000)

Abstract　The key of Model Driven Architecture is building up models and the relations between them correctly, and Giving the mapping rule between different models accurately. In this paper we discuss a serial of technology which includes the building up of metamodel, the mapping between models and code generating.

Keywords　MDA, J2EE, UML, model

1　引言

MDA是OMG提出来的软件开发过程中的模型组织管理框架，将模型提到了驱动软件体系结构和开发过程的地位，在MDA框架下，模型是多样化的，有纵向不同层次关系的模型，即平台无关模型(PIM)和不同层次的平台相关模型(PSM)[1]，有横向关系的模型，即针对于不同支撑平台的PSM，这使得MDA的应用和实现呈现多样性。

本文探讨在MDA框架下，从元模型的构建、模型的映射到最终代码的生成一系列的实现技术。

2　MDA元模型的实现

显然，MDA的核心在于模型。模型是系统的抽象，它比实现系统更容易获取、理解和计算[2]，MDA的关键在于准确地建立模型和模型之间、模型和平台之间的关系，主要的工作是在元模型和模型两个不同的层次定义模型及其关系，其次在编程实现的层面加以实现。

根据MDA的基本思想，需要抽象出与实现技术无关、完整描述业务功能的平台无关模型(PIM)，针对不同实现规则制定多个映射规则，将这个平台无关模型转换成与具体实现技术相关的应用模型(PSM)，从理想情况看，PIM模型和PSM模型属于不同种类的模型，属于模型空间不同的定位点，而不同的定位点应该采用不同的模型描述语言[3]，从实际来看，如平台无关的CLASS与基于J2EE的应用中的EJB-Class有意义上的不同，而这两种类型的模型形式上如果都由UML元模型语言描述显然不合逻辑，UML提供了扩展机制来对元模型进行扩展，即不同平台相关的模型语言采用不同的标准扩展，这样的标准就是针对不同实现的UML Profile，从而实现了模型描述的差异性，又保持了元模型的一致性，如图1所示。

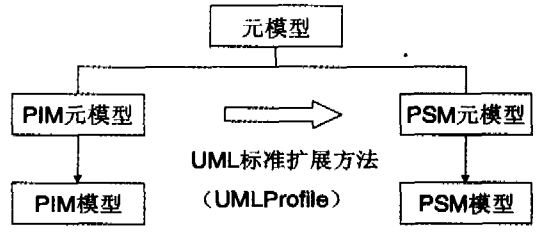

图1　UML模型的扩展

元模型显式定义建模语言的元素及之间的关系[2]，它的建模对象是语言。UML是M2层的标准建模语言，它是用MOF[4]定义的，其简化的模型图如图2[5]所示。

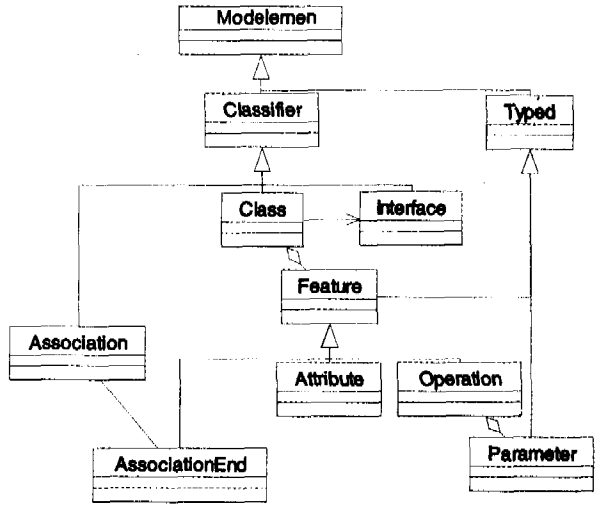

图2　UML元模型

在编码实现上我们采用的模型结构略有不同，主要在于减少了模型中的继承层次数，如图3所示将模型中所有的元素都直接从ModelElement继承，ModelElement类中将保持

*)上海市教育委员会科学项目资助，项目编号(04DB22)。**徐晓钟**　高级工程师，主要研究方向是软件工程、网络。

Stereotype 和 Tag 以及 Constraint 信息，这样所有继承的模型元素都继承了这些扩展信息。

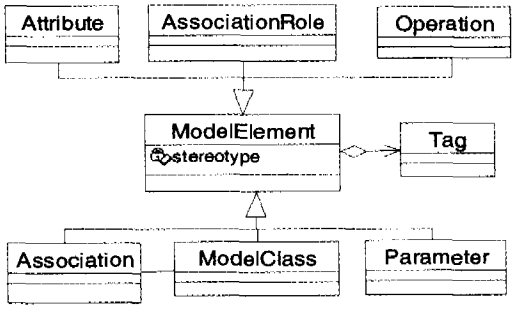

图 3　元模型的编程实现

3　模型的持久化与实例化

MDA 模型的持久化与实例化是实现 MDA 工具软件开发的重要工作，模型的持久化有两种方式：

第一种是直接将 MDA 元模型对象序列化到文件中，这种方式速度快，但占用内存大，同时模型存储文件无法和其他工具兼容；

另一种方法是将模型用 XML 格式化保存，换句话说就是 XML 来描述模型，因为 UML 基于 MOF，需要有一种基于 MOF 的数据定义标准来定义 XML，这就是 OMG 核心技术之一的基于 XML 的数据交换（XML Meta Model Interchange，XMI）[6]，通过 XMI 实现模型在不同应用之间的交换。

在工具实现中，我们使用简化的元模型结构来存储模型信息，图形化的 UML 图形类元素用如图 4 所示的 XML 文件来存储，所用模型是 UML 元模型的一个子集，在程序实现的层面上，可以实例化成元模型对象，这些元模型对象在程序工具实现时并不是在载入 XMI 时就全部创建，而是在需要时创建，创建后放入内存的缓冲区中，以备使用[7]。

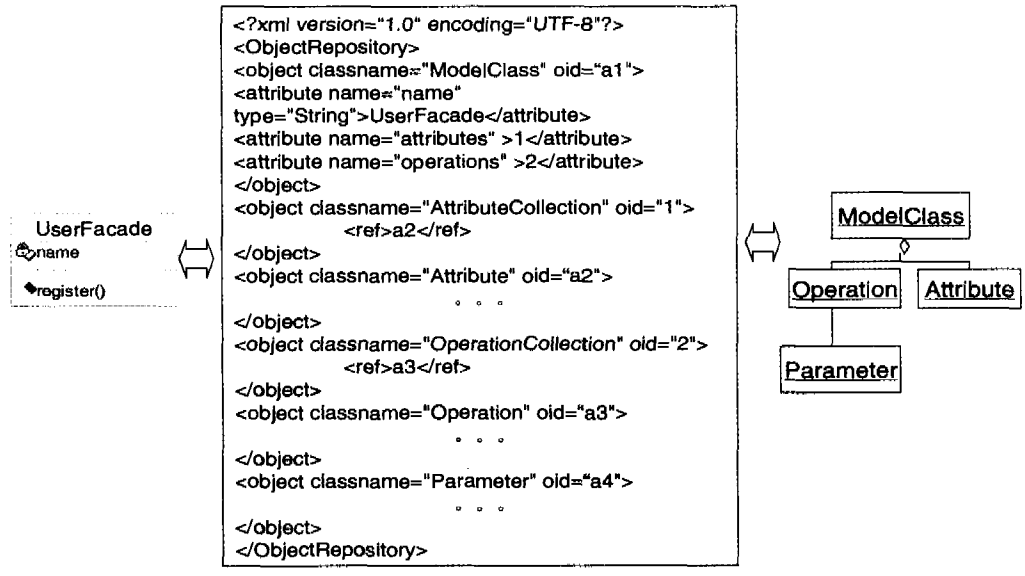

图 4　模型的转换

4　模型映射及其实现

在 MDA 工具实现及其应用中，实现不同模型之间的映射是核心工作，包括 PIM 到 PSM 模型的映射以及 PSM 到生成代码的映射。按照工具实现和应用来划分，MDA 实现的层次也分为三个层次，其一，定义 PIM 和 PSM 的元模型的实现，如前所述，PIM 和 PSM 的元模型是一致的，其差别在于 PSM 是在描述 PIM 的元模型上使用了 UML 的扩展机制，工具必须为定义不同的扩展提供环境和方法，这是工具实现的范畴；其二，设计针对不同平台的扩展机制，并根据这个扩展设计将 PIM 模型转换为 PSM 模型的环境与方法，如运用 UMLProfile for EJB[8] 实现 PIM 到 PSM FOR EJB 的映射环境，这是二级工具实现的范畴[9]；其三，实现将 PSM 模型映射为可编译发布的代码，这是最终用户应用的范畴。严格将三个层次分离在 MDA 实现中是非常重要的。

4.1　PIM 和 PSM

PIM 的定义和性质决定了其必须与 PSM 分离，PIM 的抽象层次越高，它与不同平台 PSM 转换的适应性越强。

在 PIM 和 PSM 的定义中有多种方式：

一种是将 PIM 模型分为 Service Model 和 Class Model 两部分来组织[10]，Service Model 描述系统对用户或外部系统提供的功能，类似于系统的 API，Class Model 指定了系统中必要的类，以支持业务功能。

这种 PIM 的组织方式比较适合于基于 Web 的平台的系统，如 J2EE，相应的 PSM 使用 Application Model 概念，在开发基于 J2EE 架构的系统可由 DBMS Model、EJB Model 和 Web Model 三个子模型组成，转换方法包括两个方面的映射，Class Model 映射到 DBMS Model 和 EJB Model 中的 EJB Entity，Service Model 映射到 Web Model 和 EJB Model 中的 Session Bean。这种方案概念清晰，便于实现，但缺乏灵活性。

另一种方式是进一步提升 PIM 的抽象层次，在 PIM 模型中仅使用 Class Model 概念，真正与平台无关，将功能分类放到 PSM 中去，PSM 的构建通过两个步骤，一是为 Class Model 中模型元素包括类属性和方法关联等设定相应于平台的 Stereotype、Tag 以及 constraint，二是按照特定平台应用的发布文件的结构和功能划分，将 Class Model 中的 Model Class 对象划入不同的 Component 对象中，并将各个 Component 对象以 Profile 为依据作好标记，以便下一步绑定处理映射程序，如图 5 所示。这种方式更加注重 MDA 的工具性，但对工具使用者提出了更高的要求。

（下转封三）

第 12 期略

(上接第 279 页)

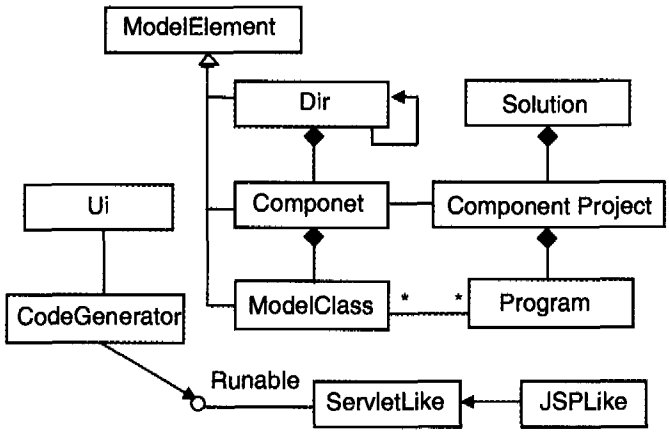

图 5 MDA 工具实现模型

4.2 PSM 到代码的映射

PSM 到代码的映射是 MDA 工程的最终目的，这里从两个层次来描述。

第一层次，从使用者(即用 MDA 工具来开发针对某个开发平台的应用模型工具)的角度来看，开发者建立针对特定平台的映射解决方案(Solution)，在解决方案中为与平台相关的各个模块创建各自的工程项目(Component Project)，这些项目将与 PSM 中的各个 component 相对并相互绑定。一个项目就是一个程序组，这些程序负责代码的转换工作，如图 5 所示。

程序的转换机制由类似于 JSP 和 Servlet 的运作的方式。即程序开发者写出类似于 JSP 的映射程序 JSPLike，再由工具生成类似 Servlet 的程序 ServletLike，如图 6 所示，使 ServletLike 实现 Runable 统一接口的类，由工具环境加载。

```
public class ServletLike_Web_xml  implements Runnable
{
    public ServletLike_Web_xml(){
    }
    public void run(CodeGenerator cg, Model model, Component compo-
    nent){
        String Path="";
      …
        if(Path.length()>0)
            Path=Path+"·";
        cg.Out().write("<? xml version=\"1.0\"encoding=\"ISO-8859-
        1\"?>\r\n");
        cg.Out().write("\r\n");
        cg.Out().write("<! DOCTYPE Web-app\r\n");
        cg.Out().write("PUBLIC\"-//Sun Microsystems, Inc.//DTD
        Web Application 2.2//EN\"\r\n");
        cg.Out().write("\"http://java.sun.com/j2ee/dtds/Web-app_2_
        2.dtd\">\r\n");
        cg.Out().write("\r\n");
        cg.Out().write("<Web-app id=\"WebApp_ID\">\r\n");
        …
        cg.Out().write("</Web-app>\r\n");
        cg.Out().write("\r\n");
        cg.Out().close();
    }
}
```

图 6 ServletLike 类的实现

第二层次，在工具实现的层次，由 CodeGenerator 动态加载 ServletLike 对象，并由界面对象 Ui 发出代码生成指令，系统采用的模型如图 5 所示。

结束语 本文对 MDA 实现技术进行了较为深入的探讨，从此 MDA 工具的实现和使用过程来看，PIM 层的抽象程度越高，其工具性越强，但对使用者的要求也就更高，这是一对矛盾，也是制约 MDA 大规模普及的重要因素。MDA 的成熟还有很长的路要走，同时也离不开底层支撑平台的发展。

参 考 文 献

1 OMG Achitecture Board ORMSC. Model Driven Achitecture(MDA). OMG Document number ormsc/2001-7-01, available from www.omg.org, July 2001
2 周颖，等. 基于 MDA 的 UML 模型转换：从功能模型到实现模型. 计算机应用与软件，2005，22(11)
3 Kent S. Model Driven Engineering. In: IFM 2002, LNC 2335. 286~298
4 Meta Object Facility(MOF)Specification, Version 1.4, April 2002
5 KLeppe A，等著. 解析 MDA. 鲍志云译. 人民邮电出版社，2004
6 OMG XML Metadata Interchange (XMI) Specification, version1.1 [EB/OL]. Http://www.OMG.org/, 2002-10-10
7 Silaghi R. MDA Refinements along Middleware-Specific Concern-Dimensions. Middleware 2004 Companion, 1st International Middleware Doctoral Symposium Toronto, Canada
8 Greenfield J. UML Profile For EJB. Public Draft, Rational Software Corporation, 2001-5-25
9 Emmerich W. Distributed Component Technologies and their Software Engineering Implications. ICSE'02, 2002
10 谢正良，等. 一种基于 J2EE 平台的 MDA 模型转换技术. 计算机应用研究，2005(3)