基于 J2EE 平台的 MDA 模型转换技术*)

史耀馨　崔　萌　李宣东　郑国梁
(南京大学计算机科学与技术系　南京 210093)

A Mapping Approach for MDA Model Transformation Based On J2EE Platform

SHI Yao-Xin　CUI Meng　LI Xuan-Dong　ZHENG Guo-Liang
(Department of Computer Science and Technology, Nanjing University, Nanjing 210093)

Abstract The OMG's MDA defines an approach to IT system specification that separates the specification of system functionality from the specification of the implementation of that functionality on a specific technology platform. To this end, the MA defines two kinds of models: platform independent model (PIM) and platform specific model (PSM). The MDA approach allows the same model specifying system functionality to be realized on multiple platforms through auxiliary mapping standards from PIM to PSM. This paper presents a mapping method from PIMs to PSMs based on J2EE platform. This transformation can promote the efficiency of system analysis and design.

Keywords MDA, PIM, PSM, J2EE, UML, Mapping

1. 引言

1.1 UML

UML[1](Unified Modeling Language)是一种可视化的通用面向对象建模语言。它被用于描述和构建复杂的软件系统。面向对象建模语言产生于 20 世纪 70 年代中期，此后曾出现过数十种不同的建模语言。这些语言各有千秋，并且存在细微的差异，给用户的交流和使用带来了不便。自 1995 年以来，Booch、Rumbaugh 和 Jacobson 就致力于设计一种统一的建模语言，命名为 UML。此后很多的软件公司也加入了 UML 的设计工作。到 1997 年，UML 1.1 成为 OMG 的标准。UML 得到了工业界、科技界和应用界的广泛支持，是目前最流行的面向对象建模语言。

UML 中提供 9 种图来描述系统不同方面的属性，分别是：用例图、类图、包图、组件图、配置图、顺序图、合作图、状态图和活动图。这些图可以分为两类：静态视图和动态视图。它们互为补充，共同定义了整个系统。类图、包图、组件图和配置图属于静态视图。静态图描述了系统中恒定的信息，这些信息是不随系统状态的变化而变化的。这些信息包括：类的组织结构(属性和行为)，类的接口，类之间的相互关系(继承、聚集和组成)，包的结构，以及系统的物理结构。动态图描述了对象间的通信(消息交互)以及多个对象是如何结合在一起来完成某个功能的。动态图包括用例图、顺序图、合作图、状态图和活动图，它们各自的侧重点不同，分别用于不同的目的。顺序图、合作图主要用于描述对象之间的交互，状态图、活动图主要用于描述对象的内部行为。另外 UML 还提供了类描述语言和对象约束语言来描述不能由静态图和动态图表示的信息。它直接使用文本方式来描述概念。类描述语言用于定义类的具体信息，相当于传统的数据字典。对象约束语言用于描述对象的约束、条件和断言。

1.2 MDA

早在 1989 年，OMG(Object Management Group)就开始致力于解决企业间应用集成的问题，并制定了跨平台的应用集成和互操作的标准——CORBA。CORBA 允许对象在异构的硬件平台、异构的操作系统和异构的编程语言中平滑地进行交互。然而 CORBA 并不是唯一的中间件平台，像 .NET，J2EE 都提供了类似的服务，所以目前的中间件技术并不足以解决企业间跨平台计算的要求。为此，OMG 提出了 MDA[2](Model Driven Architecture)技术。MDA 将系统的功能描述和系统在特定平台上的实现分离开来，它把系统的模型分为两种：一种是与平台无关的模型(PIM，Platform Independent Model)，一种是与平台有关的模型(PSM，Platform Specific Model)。PIM 是对系统的功能和结构的形式化描述，是抽象的、与平台细节无关的。PSM 是在特定的目标平台上的系统功能和结构的描述，它在 PIM 中添加了与平台有关的信息，但是 PSM 并不等同于实现，它仍然是用模型语言来描述的。在 MDA 中，PIM 和 PSM 都用 UML 来描述(对于具体的平台，要对 UML 进行适当的扩充以描述与平台有关的信息)。一个 PIM 可以映射到多个 PSM，如图 1 所示。

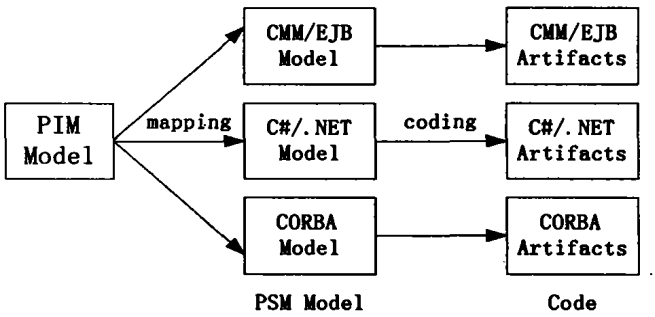

图 1　PIM 到 PSM 的映射

*)国家自然科学基金资助项目(69703009)；国家 863 高科技发展计划资助项目(2001AA113203)。史耀馨　硕士生，主要研究领域为软件工程，形式化方法。崔　萌　博士生，主要研究领域为软件工程，形式化方法。李宣东　教授，博士生导师，研究领域为形式化方法，模型检验。郑国梁　教授，博士生导师，研究领域为软件工程，软件开发环境。

MDA 定义了 PIM 和 PSM 模型之间的 4 种映射关系：(1) PIM 到 PIM。该映射包括模型之间的增强、过滤和精化，其中最主要的就是从分析模型到设计模型的转换。(2) PIM 到 PSM。该转换是从抽象模型映射到与目标平台相关的模型。(3) PSM 到 PSM。该映射是对 PSM 模型的精化，主要包括组件的实现、封装和配置。(4) PSM 到 PIM。该转换类似于逆向工程，是从 PSM 中提取与平台无关的模型信息，这是一个信息挖掘的过程。

MDA 有 3 个层次：其核心是 UML、MOF、CWM 这 3 个建模标准；其次是基于核心的一些公共服务，如：目录服务、事件服务、持久性、安全性；最外层是 MDA 在各个领域的应用，如：通讯业、电子商务、金融领域等等。MDA 的核心思想就是从 PSM 中抽取出与平台无关的 PIM 模型，这样做有如下的好处：(1)有利于验证模型的正确性；(2)易于在不同的平台上开发企业应用组件；(3)方便组件的移植；(4) 易于实现跨平台的应用集成和互操作。

1.3 Java 和 J2EE

Java 是 SUN 公司开发的一种跨平台的开发语言。它具有简单、面向对象、分布式、解释执行、安全、跨平台、高性能、多线程等特点。Java 自从诞生后得到了广泛的应用，它为开发者提供了"编写一次，任意运行"的优点，提高了应用程序的可移植性。Java 语言特别适合网络应用的开发，包括客户端和服务器。在 Java 之后，不断有新的 API 和标准被开发，比如：JSP、JDBC、JavaBean。最终 SUN 和其合作伙伴将所有企业相关的标准和 API 统一到 J2EE[3]平台上。J2EE 在各种领域内创建了适用于企业计算需要的一系列标准，如：数据库连接、企业业务组件、面向信息的中间件、Web 相关组件、通信协议、协同工作等。J2EE 提供了组件运行的底层架构和基本服务，并规定了组件开发的一组规范，因此开发人员可以专注于业务需求，而不需要花时间去创建底层复杂的应用架构，从而提高软件开发的效率和质量。

J2EE 是一组开放的标准，其核心是 EJB[3](Enterprise JavaBean)。EJB 结构提供开发可重用的、运行在应用服务器上的 Java 服务器组件的标准。EJB 规范和 API 独立于供应商和应用服务器的编程接口，它为企业应用提供持久性、业务处理、事务处理、分布式处理的能力。简而言之，EJB 提供了业务组件的可移植性。EJB 是一个打包的 JAR 文件，它包括一组 Java 类和用 XML 写成的配置描述信息。其中，Java 类包括 Remote 接口、Home 接口、Local 接口、Local Home 接口、实现 Bean 功能的主类，以及一些辅助的类。EJB 可以分为 Session Bean、Entity Bean 和 Message-Driven Bean。Session Bean 用于和客户端进行短暂的会话，当会话结束时，Session Bean 就消亡了。一个 Session Bean 只能被一个客户端访问，是不可以被共享的。Session Bean 可以分为有状态的和无状态的两种。Entity Bean 描述了数据库中的一个持久对象，当客户端终止会话或服务器关闭的时候，底层服务将保证 Entity Bean 中的数据被保存。Entity Bean 中数据的持久性可以由 EJB 容器来实现，也可以由 Bean 自身实现。Entity Bean 可以同时被多个客户端访问，事务管理和数据完整性由底层设施实现。Message-Driven Bean 既是一个 Session Bean，又是一个 JMS(Java Message Service)消息的接收者，它允许业务组件监听异步的 JMS 消息，提供消息驱动的服务。

J2EE 架构是一个多层结构，包括：用户层、Web 层、业务层和 EIS 层。用户层用来与用户交互，并把来自系统的信息显示给用户。J2EE 支持不同类型的用户，可以是 HTML 用户、Java Applet 或 Java 应用。Web 层产生表示逻辑，并接受来自客户端的用户反馈，这些用户通常为 HTML 客户端。这一层由 JSP 和 Servelet 来实现。业务层处理应用的核心业务逻辑，由 EJB 来实现。EIS 层为企业的信息系统服务，包括数据库系统、事务处理系统、遗产系统和企业资源计划系统。该层是 J2EE 应用和非 J2EE 应用的连接点。如图 2 所示。

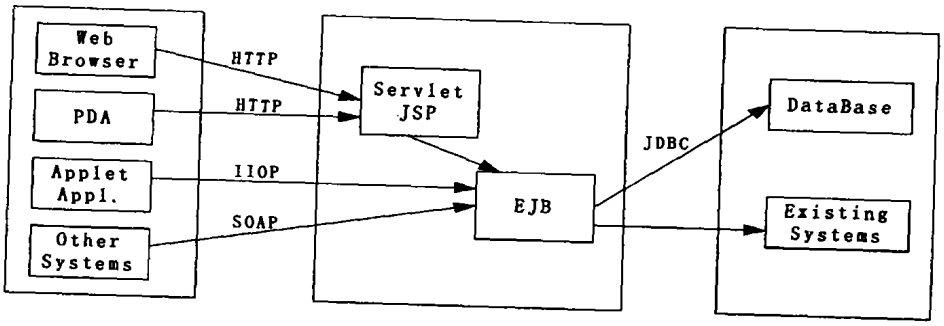

图 2 J2EE 的架构

2. 对 UML 的扩充

UML 作为通用的面向对象建模语言，它没有为特定的实现平台提供必要的建模设施。事实上，UML 并不能全面而准确地描述 J2EE 架构中的语义信息。UML 是一种可扩充的语言，它提供了一组标准的扩展机制来定义新的语义结构。这些扩充的内容通常称为 UML 侧面(UML Profile)。UML 的扩展机制包括：约束(Constraint)、构造型(Stereotype)和标记值(Tagged Value)。

在 MDA 中，MOF(Meta Object Facility)定义了一组元模型用以描述其他的模型，比如：UML 和 CWM。UML Profile 也用 MOF 进行描述。一个构造型是 MOF 中某个 metaclass 的子类，它对 metaclass 的实例进行分类，并扩充了新的约束和标记值。比如构造型《TopLevelPackage》就扩充 metaclass 中的 Package，并增加了约束：TopLevelPackage 的实例不可以包含在其它的包中。标记值是 metaclass 的一个属性，用于描述 metaclass 的某些信息。一个构造型通常有一组的标记值，比如构造型《JavaClass》就包括 isPublic、isAbstract、isFinal 3 个布尔型的标记值。约束是用 OCL 描述的语义信息，它可以应用在所有的模型元素上，比如：构造型和标记值。

2.1 Jave 的 UML Profile

UML 对 Java 语言的扩充如表 1 和表 2 所示。两张表分别列出了 UML 侧面中的构造型和标记值。

表 1 UML Profile for Java——构造型

构造型	对应的 MOF 中的元类	说明
《JavaClass》	Class	Java 语言中的类
《JavaInterface》	Class	Java 语言中的接口
《JavaPackag》	Package	Java 语言中的包
《JavaMethod》	Operation	类中的方法
《JavaParameter》	Parameter	方法中的参数
《JavaClassFile》	Component	Java 编译后生成的 .class 文件

2.2 J2EE 的 UML Profile

UML 对 J2EE 平台的扩充如表 3 和表 4 所示。两张表分别列出了 UML 侧面中的构造型和标记值。

除了构造型和标记值，UML 侧面中还包括一些约束，比如：《EJBRemoteMethod》中的参数必须是 RMI/IIOP 中允许的类型，《EJBImplementation》的可见性必须为 public 等等。由于约束不是本文的主要内容，在后面的模型转换中也没有用到，所以不作深入分析。

表 2 UML Profile for Java——标记值

标记值	类型	所属的构造型	说明
JavaStatic	布尔型	《JavaClass》,《JavaInterface》,《JavaMethod》	类、接口或方法是否是静态的
JavaAbstract	布尔型	《JavaClass》,《JavaMethod》	类或方法是否是抽象的
JavaConstructor	布尔型	《JavaMethod》	是否为构造函数或析构函数
JavaNative	布尔型	《JavaMethod》	是否为本地方法
JavaFinal	布尔型	《JavaClass》,《JavaMethod》,《JavaParameter》	类、方法或参数是否为 final
JavaThrows	字符串	《JavaMethod》	方法抛出的异常
JavaDimensions	整型	《JavaParameter》	参数的维数

表 3 UML Profile for J2EE——构造型

构造型	对应的 MOF 中的元类	说明
《EJBServlet》	SubSystem	Servlet
《EJBSessionBean》	Subsystem	Session Bean
《EJBEntityBean》	Subsystem	Entity Bean
《EJBRemoteInterface》	Class	EJB 的 Remote 接口，是非本地对象与 EJB 交互的途径
《EJBHomeInterface》	Class	EJB 的 Home 接口，提供 EJB 对象的创建、消亡、激活等标准操作
《EJBMethod》	Operation	EJB 的 Home 接口中的方法，如：ejbCreate,ejbRemove,ejbActivate,ejbPassivate 等
《EJBFinderMethod》	Operation	Entity Bean 的 Home 接口中的 findBy 方法，用于数据库查询
《EJBRemoteMethod》	Operation	EJB 的 Remote 接口中的方法
《EJBPrimaryKey》	Usage	扩充了类之间的使用关系，表明是 EJB 的 Home 接口使用了某个数据库
《EJBCmpField》	Attribute	由容器实现持久性的 Entity Bean 中的数据(Bean 的属性)
《EJBPrimaryKeyField》	Attribute	表明该属性是关键字
《EJBRealizeHome》	Abstraction	表示实现了 Home 接口
《EJBRealizeRemote》	Abstraction	表示实现了 Remote 接口
《EJBImplementation》	Class	实现 EJB 功能的主类，以区别 EJB 子系统中的其他辅助的类
《EJBAccess》	Association	角色和 EJB 之间的访问关系，描述了 Bean 的安全策略中的访问控制
《EJBDescriptor》	Component	EJB 的配置描述信息
《EJB-JAR》	Package	为 Bean 打包后生成的 JAR 文件

表 4 UML Profile for J2EE——标记值

标记值	类型	所属的构造型	说明
EJBSessionType	枚举型	《EJBHomeInterface》	Session Bean 的类型：Stateless 或 Stateful
EJBRoleName	字符串	Operation,《EJBMethod》《EJBFinderMethod》,《EJBRemoteMethod》	与方法相关的角色
EJBTransType	枚举型	《EJBSessionBean》	事务管理是由 Bean 实现还是由 Container 实现
EJBPersistType	枚举型	《EJBEntityBean》	持久性是由 Bean 实现还是由 Container 实现
EJBReentrant	布尔型	《EJBEntityBean》	Bean 是否可重入
EJBNameInJAR	字符串	《EJBSessionBean》《EJBEntityBean》	EJB-JAR 中的 Bean 的名称，缺省为其 Remote 接口名

3. 从 PIM 到 PSM 的转换技术

MDA 的一个主要内容就是从 PIM 到 PSM 的映射，也就是从分析模型到实现模型的转换。在这一节中，我们将以一个银行系统为例，给出从 PIM 到 PSM 转换的具体步骤和方法。对于 PIM，我们使用 UML 来建模；对于 PSM，我们使用上一节中的扩充 UML 来建模。银行系统是一个通用而又简洁的例子，能够很好地体现 PIM 到 PSM 转换的思想。

3.1 银行系统的 PIM

银行系统的 PIM 模型如图 3 所示。在其业务逻辑中有 4 个类：Bank 是银行类，它包含银行名、银行地址、银行编号等属性以及对客户和帐户的增、删、查询的操作。Customer 是客户类，包含客户的姓名、编号等属性，对这些属性的 Get 和 Set 方法，以及对一个客户所拥有的帐户的增、删、查询的操作。Account 是帐户类，包含帐户的编号、余额等属性，对这些属性的 Get 和 Set 方法，以及对一个帐户所属的客户的增、删、查询的操作。TransactionRecord 是某个银行帐户的一次交易事务，可以是取款或存款，该类包含交易的编号、数额、种类等属性以及对这些属性的 Get 和 Set 方法。一个 Bank 可以对应多个 Customer 和多个 Account，Customer 和 Account 之间是多对多的关系，一个 Account 可以对应多个 TransactionRecord。另外 Account 还有两个子类：企业帐户和储蓄帐户。银行系统的 Web 客户端只有一个浏览器类。银行系统的 Application 客户端包括 3 个类：ATMWindow 是用户

界面类，包括按钮、菜单和标签等，用于和用户交互、获取用户的鼠标键盘事件。EventHandler 是事件处理类，用于对 ATMWindow 捕获的事件进行处理。DataModel 用于和银行服务器交互，它获取 ATMWindow 中的数据、对数据进行有效性验证、将数据传给银行服务器并获取返回数据、将结果数据显示在 ATMWindow 上。

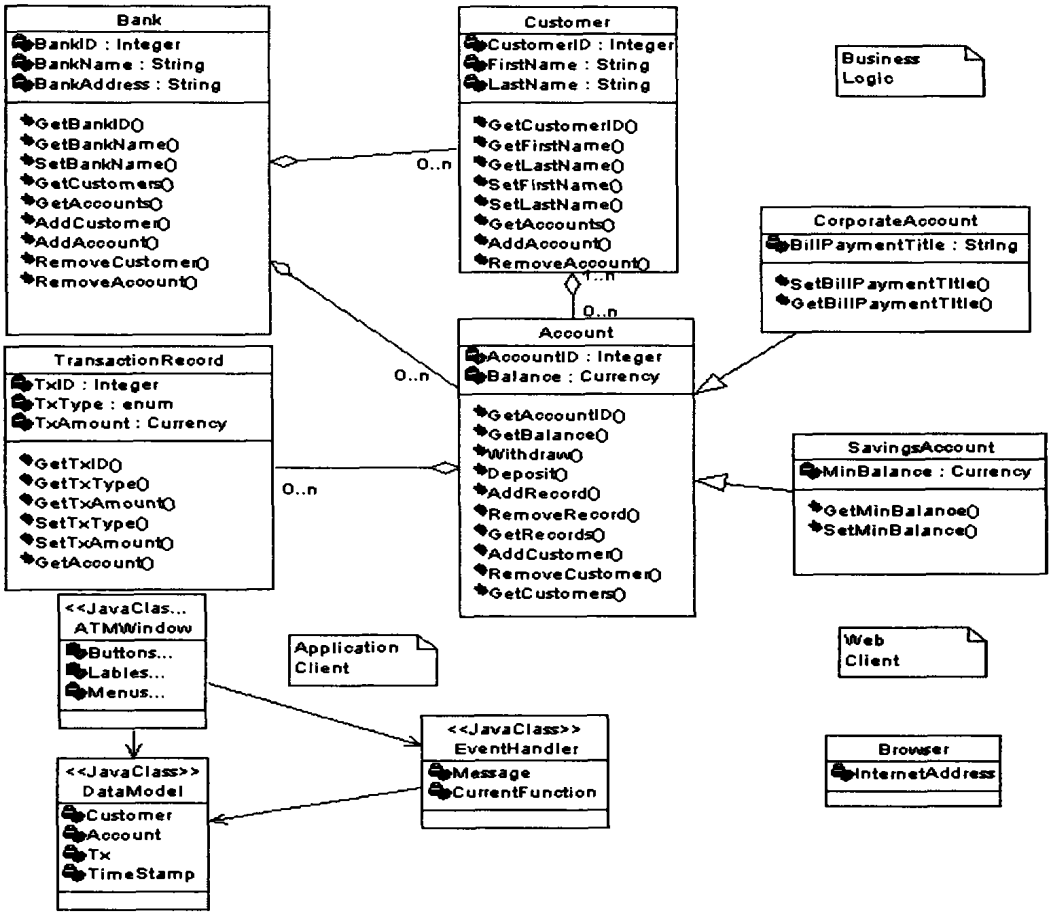

图 3　银行系统的 PIM 模型

在 PIM 模型中，可以使用配置图来描述类图中每个类属于 Client/Server 结构中的哪个层次。配置图的信息是实现 PIM 到 PSM 的自动转换所必需的。在转换过程中，对属于不同结点的类进行不同的映射操作。银行系统的配置图如图 4 所示，该系统有两个 Client 结点，一个 Server 结点。在每个结点中列出了运行在其上的类。

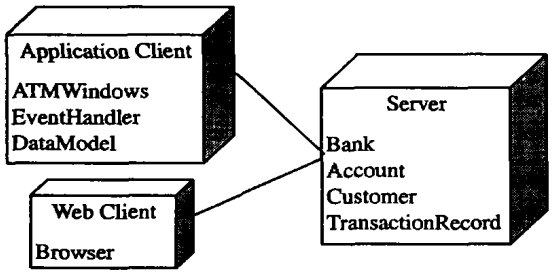

图 4　银行系统的配置图

3.2　客户端的 PIM 到 PSM

如果系统使用 B/S 结构，则客户端是通用的浏览器，不需要设计人员为之建模。如果系统使用 C/S 结构，那么客户端是 Java 应用程序，对原先的 PIM 模型图进行如下精化：

- 对类图中的类、接口、包、方法、参数设置相应的构造型（对应关系参见表 1）；
- 设定每个构造型中标记值的值（比如是否为 static、abstract 等，参见表 2）；
- 修改类中属性的类型，使之成为有效的 Java 类型（比如 Integer->int，Currency->double）；
- 为每个类添加构造函数、析构函数；
- UML 中包的层次关系用“::”表示，在 Java 中用“.”表示；
- 类图中的使用关系对应为 Java 中的 import。

3.3　后台数据库的 PIM 到 PSM

为数据库建模时首先考虑业务逻辑层中哪些类的数据是要存放在数据库中的，然后按以下规则设计数据库：

- 如果类 A 和类 B 是“一对一”的关系，则把 A 的关键字作为 B 的外关键字，或者把 B 的关键字作为 A 的外关键字；
- 如果类 A 和类 B 是“一对多”的关系，则把 A 的关键字作为 B 的外关键字；
- 如果类 A 和类 B 是“多对多”的关系，则增加一个新的关系表，其属性为 A 的关键字和 B 的关键字。

在银行系统中，Customer 类、Account 类和 TransactionRecord 类中的数据要保存在数据库中。由于 TransactionRecord 类和 Account 类是“多对一”关系，因此在 TransactionRecord 的数据库中使用 AccountID 作为外关键字。由于 Account 类和 Customer 类是“多对多”关系，因此需要一个额外的数据库来保存 Account 和 Customer 的对应关

系。这一步转换后的结果如图 5 所示。

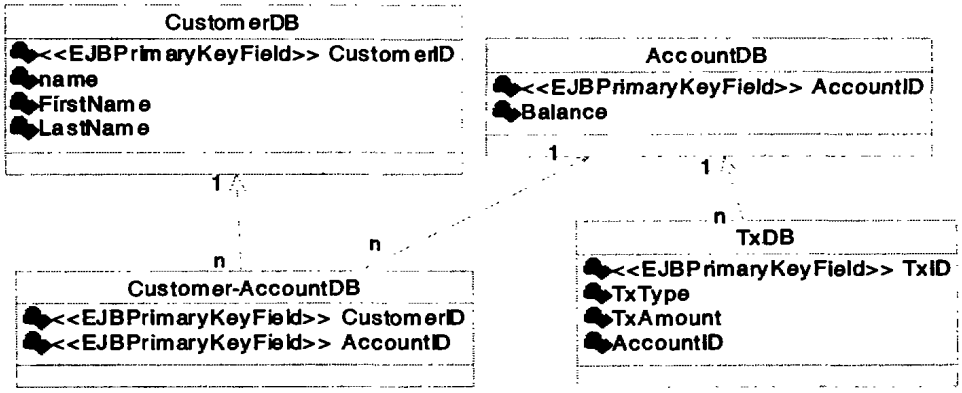

图 5　数据库的 PSM 模型

3.4　业务层的 PIM 到 PSM

业务层的 PIM 模型到 PSM 模型的映射是 MDA 映射中最关键的部分。该转换过程分为 4 步：(1)为业务逻辑中的每个类生成对应的 Session Bean 子系统和 Entity Bean 子系统。Session Bean 用于和客户端的交互，Entity Bean 用于和后台数据库的交互。(2)精化每个 EJB 子系统，建立 Bean 所需的接口和类。(3)建立角色和 Bean 之间的访问关系，用于描述 EJB 的安全策略。(4)生成整个业务逻辑的组件图。下面将详细分析各个步骤。

3.4.1　*生成 EJB 子系统及其相互关系*　在图 1 的业务逻辑中，有 3 个类：Customer，Account，TransactionRecord (Bank 类描述了一些系统信息，与业务逻辑无关)，我们为每个类构造一个 Session Bean 和一个 Entity Bean，如图 6 所示。由于客户端可能是浏览器，所以在业务逻辑中要增加一个 Servlet 来进行 HTTP 交互。

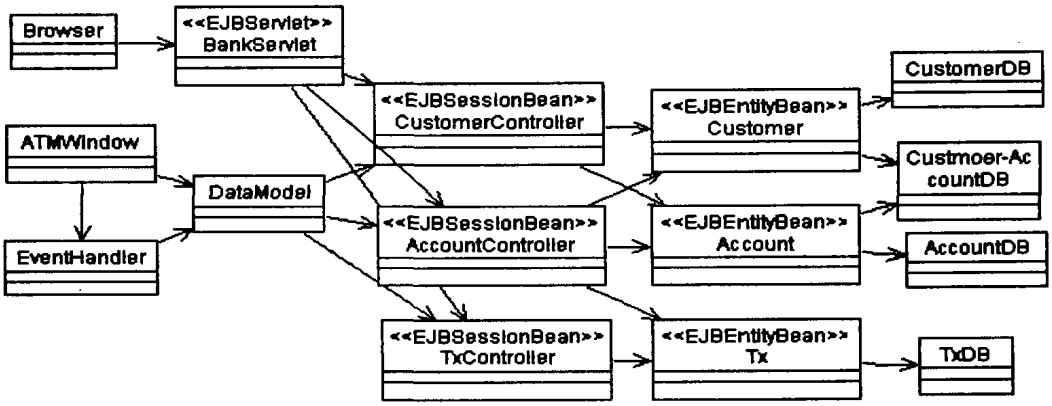

图 6　EJB 子系统及其相互关系

3.4.2　*精化每个 EJB 子系统*　对于 Servlet 子系统，它可以分成两部分：Dispatcher 和 BeanManager。Dispatcher 是 javax.servlet 包中 HttpServlet 的子类，用于获取浏览器提交的信息以及生成返回的 HTML 页面。BeanManager 类用于和 Enterprise Bean 交互，它需要用到 Enterprise Bean 所提供的接口。

对于 EJB 子系统，它包括一组相关的类和接口，其精化操作如下：

1. 生成 Bean 的实现类。如果是 SessionBean，则该类继承 javax.ejb 包中的 SessionBean 类；如果是 EntityBean，则该类继承 javax.ejb 包中的 EntityBean 类；类的构造型为《EJBImplementation》。类中的属性就是 PIM 类图中对应的类(比如：PSM 中的 CustomerController 对应于 PIM 中的 Customer 类)中的属性，其构造型为《EJBCmpField》。类中的方法包括：PIM 类图中对应的类的方法，与 Home 接口对应的 ejbCreate、ejbRemove 等 Bean 的方法，以及一些私有的访问数据库的方法。类映射中的其他内容如 3.2 节所述。

2. 生成 Remote 接口。Remote 接口继承了 EJBObject，其构造型为《EJBRemoteInterface》。Remote 接口中的方法是 PIM 类图中对应的类中的方法，方法的构造型为《EJBRemoteMethod》。

3. 生成 Home 接口。Home 接口继承了 EJBHome，其构造型为《EJBHomeInterface》。如果是 Session Bean，则 Home 接口中的方法为：Create 和 Remove，构造型为《EJBMethod》。如果是 Entity Bean，则除了 Create 和 Remove 方法外，还有一组 findBy 方法，其构造型为《EJBFinderMethod》。

4. 生成 Local 接口和 Local Home 接口。Local 接口在有的系统中并不需要，其构造方法类似于 2 和 3。

5. 建立 Bean 和 Remote 接口、Home 接口之间的关系：《EJBRealizeHome》和《EJBRealizeRemote》。以及建立 Home 接口和数据库之间的关系：《EJBPrimaryKey》。

图 7 是对 Customer 子系统精化的结果。

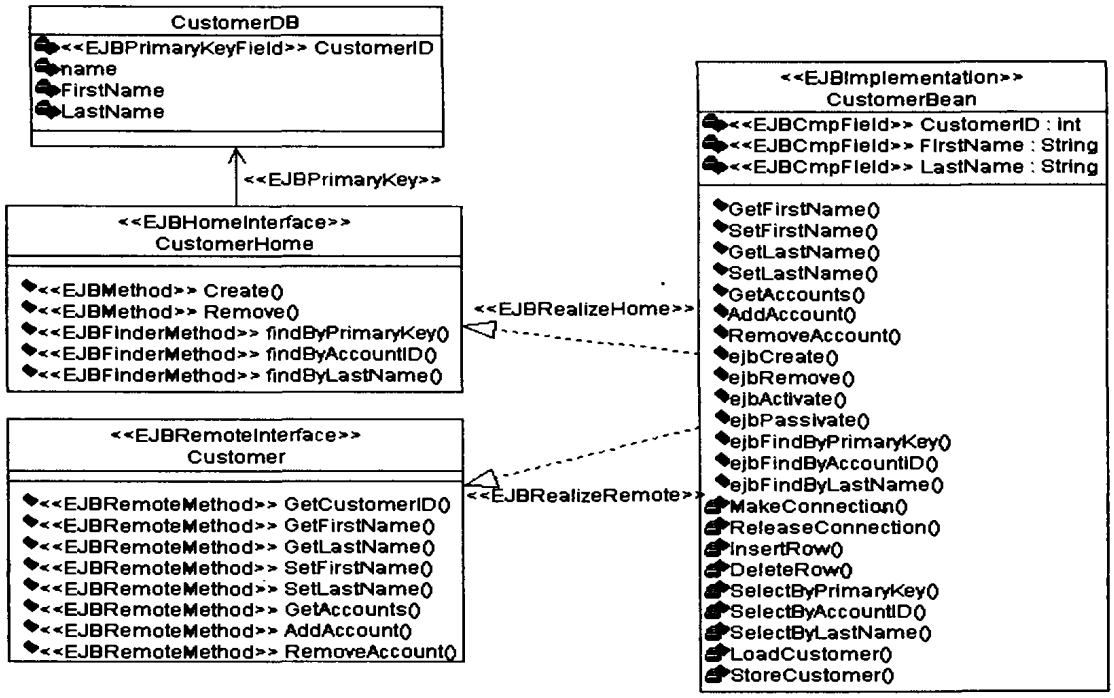

图 7　Customer 子系统

3.4.3　*建立访问控制关系*　在 J2EE 平台中使用如下机制来保护 Bean 资源，实现 Bean 的安全性：首先定义系统中的角色；然后为每个方法指定哪些角色可以调用它；最后在程序中可以用 EJBContext 接口的 getCallerPrincipal 和 isCallerInRole 方法来实现安全控制。该机制实际上是一个访问控制列表(Access Control List)。

在银行系统中，有两个角色：银行客户和系统管理员。系统管理员允许执行：创建/删除帐号，创建/删除客户，建立帐号和客户之间的关联，设置一些初始值等操作。银行客户允许执行：取款，存款，转帐，查询余额，查询交易记录等操作。Operation 的字符串型标记值 EJBRoleName 列出了可以执行该操作的所有角色名(参见表 4)。

3.4.4　*生成组件图*　最后是生成系统的组件图。在每个子系统(Enterprise Bean)在实现时都对应一个 JAR 包，它包含各个类和接口的 .class 文件以及配置描述信息。根据 3.4.2 节对子系统精化的结果，很容易得到子系统的组件图以及整个系统的组件图。图 8 是 Customer 这个 Entity Bean 的组件图。

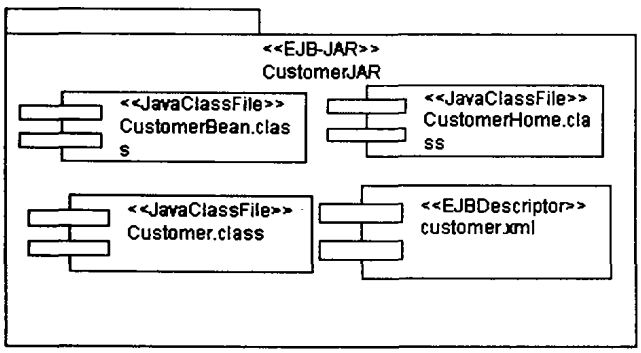

图 8　Customer 的 JAR 包

结论　在 MDA 中，PIM 到 PSM 的转换按自动化程度的高低可分为 4 个层次：(1)分析设计人员手工将 PIM 转换成 PSM；(2)使用一些精化模式来辅助手工转换；(3)设计一个算法从 PIM 生成 PSM 的框架，然后再手工精化 PSM 框架；(4)设计一个算法从完整的 PIM 模型生成完整的 PSM 模型。在本文中，主要使用类图来描述与平台无关的系统信息，并使用配置图来描述每个类处在 Client/Server 结构的哪一层。根据这些 PIM 模型的信息，本文给出了一个产生 PSM 模型的步骤，这些步骤很容易用算法来实现，基本上达到了 MDA 中第 4 个层次的要求，即实现模型从 PIM 到 PSM 的自动转化。

本文所作的工作主要基于系统的静态模型。今后的工作是：针对系统动态模型的 PIM 到 PSM 的转换步骤，以及设计一个基于 Rational Rose 的工具以实现模型的自动转换。

参考文献

1　OMG Unified Modeling Language Specification. Available at: http://www.omg.org

2　OMG Architecture Board ORMSC. Model Driven Architecture (MDA). OMG document number ormsc/2001-07-01. Available at: http://www.omg.org

3　Java 2 Platform Enterprise Edition Specification. Available at: http://java.sun.com

4　UML Profile for EJB Public Draft. Available at: http://www.jcp.org

5　Kent S. Model Driven Engineering