

基于 MDA 的模型转换研究与应用

王永涛, 刘 勇

(河南科技大学电子信息工程学院, 河南 洛阳 471003)

摘 要: 模型驱动方法解决了软件开发的效率低、可移植性差等问题, 其中的模型转换是开发基于模型驱动构架(MDA)应用工具的关键技术。为此, 在模型驱动方法的基础上, 提出基于模式的平台无关模型到平台相关模型的模型转换方法, 并根据该转换方法确立转换规则, 在一个 MDA 应用系统开发实例中进行验证, 实现从平台无关层模型到 J2EE 平台相关层 EJB 模型的转换。

关键词: 模型驱动构架; 元模型; 域模型; 平台相关模型; 模型转换

Research and Application of Model Transformation Based on MDA

WANG Yong-tao, LIU Yong

(College of Electronic Information Engineering, Henan University of Science & Technology, Luoyang 471003, China)

【Abstract】 Model driven methods are to solve the low efficiency, poor portability in software development, model transformation is the key technology in development of application tools based on Model Driven Architecture(MDA). This paper introduces the theory about MDA technology, proposes the model transformation method of Platform Specific Model(PIM) to Platform Specific Model(PSM) based on pattern, according the transformation method to establish the transformation rules, it is verified in an instance of MDA application development, this method realizes the transformation from platform independent model to enterprise Java bean model of J2EE in platform specific model.

【Key words】 Model Driven Architecture(MDA); meta model; domain model; Platform Specific Model(PSM); model transformation

DOI: 10.3969/j.issn.1000-3428.2011.16.029

1 概述

模型驱动架构(Model Driven Architecture, MDA)核心思想是抽象出与实现技术无关, 完整描述业务功能的核心平台无关模型(Platform Independent Model, PIM), 然后针对不同实现技术制定多个转换规则, 通过这些转换规则及辅助工具将 PIM 转换成与具体实现技术相关的平台相关模型(Platform Specific Model, PSM), 最后经过充实的 PSM 转换为代码。MDA 的目的是通过 PIM 和 PSM 分离业务建模与底层平台技术, 以保护建模的成果不受技术变迁的影响。

模型转换是模型驱动架构的核心技术, 用来解决模型到模型以及模型到代码间的映射问题^[1], 并将模型映射为不同技术平台上的实现。本文以企业人力资源管理系统的软件开发为应用背景, 围绕模型转换应用于软件开发尚待进一步完善的问题展开研究, 并实现了平台无关层模型到 J2EE 平台相关层 EJB 模型的转换。

2 MDA 相关理论

2.1 模型驱动体系构架

MDA 是由对象管理组织(Object Management Group, OMG)于 2001 年提出来的。MDA 将软件系统的模型分为: 平台无关模型(Platform Independent Model, PIM)和平台相关模型(Platform Specific Model, PSM), 并且它们之间通过相应的转换规则联系起来。PIM 是一个不考虑具体实现技术的纯分析模型, 在这个层次上 PIM 是可重用的, 通过 PIM 进一步提高了软件系统的抽象层次, 同时也屏蔽了底层平台技术的变化所带来的冲击; PSM 是与特定平台系统相关的模型, 它基于某一个特定的实现技术, 比如 .NET、J2EE 平台等。在 MDA 方法的开发过程中, 首先使用平台无关的建模语言构建 PIM, 然后根据具体的实现平台, 将 PIM 转换成与特定平台相关的 PSM, 最后由代码生成工具生成最终的应用程序代码。其中,

模型间的变换是通过模型定义语言之间的变换来实现的。

2.2 元模型

元模型是陈述建模的模型。元模型定义了一种建模语言的抽象语法(abstract syntax)和静态语义(static semantics)。元模型是特殊的模型, 它的建模对象是语言。不同的建模目的和描述角度产生不同的语言。模型管理自动化要求元模型显式化, 模型必须能像数据一样处理。模型既是模板也是数据。

2.3 基于模式的 PIM 到 PSM 模型转换方法

模型转换的本质是读取源模型, 按照转换规则将其转换为目标模型。MOF 将模型分为 4 个层次, 如果源模型和目标模型都处于 M1 层, 那么这种模型转换常被称为模型进化, 这是最常见的一种模型转换, 通常认为代码是 M0 层。下文的平台独立模型(PIM)到平台相关模型(PSM)的转换是 M1 层上的转换。图 1 描述一种基于模式的模型转换定义方法。

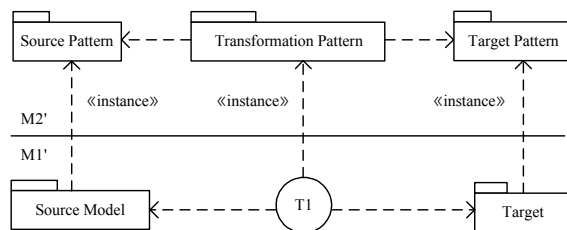


图 1 基于模式的模型转换方法

其中, M1' 层是对 UML 模型层 M1 层的扩展, 从而支持 UML 模型转换的表示法, M2' 层是对 UML 元模型层 M2 层的扩展, 从而支持转换的元建模。本文使用扩展机制实现上

基金项目: 国家自然科学基金资助项目(70671035)

作者简介: 王永涛(1984—), 男, 硕士研究生, 主研方向: 模型驱动构架技术; 刘 勇, 教授

收稿日期: 2011-01-26 **E-mail:** ytw719@163.com

述扩展。可以使用 UML Profile 这种轻量级的扩展, 通过定义一组构造型、一组相关约束以及一组标记值实现。

根据上述对转换模式的定义, 从 PIM 到 PSM 的模型变换过程中, 在模型变换定义中可以借助相应的模式实现平台独立模型到平台相关模型的变换。

3 PIM 层的 UML 模型到 PSM 层的 EJB 模型转换

本文以企业人力资源管理系统为例, 利用 OptimalJ 工具来开发其中的销售管理模块的 PIM 建立和 PIM 如何转换为相对复杂的 PSM 过程。

3.1 PIM 模型的建立

PIM 是 MDA 框架开发的基础, 对系统结构和功能进行抽象的规约^[2], 不考虑实现的技术细节。在 OptimalJ 中, 与之对应的是域模型。域模型又分为域类模型和域服务模型。域类模型侧重于描述系统的静态结构, 域服务模型主要侧重于业务规则的实现, 展现应用程序的事务、信息的动作类型, 描述系统的动态特征。

(1) Domain 模型

对应 MDA 中的 PIM, 它是 MDA 开发的核心部分。Domain 模型中包含 Class 和 Service 2 种模型, 使用 UML 表示模型。允许导入基于 XMI 的 XML 文件。

Domain Class 模型定义了应用程序使用的信息的静态结构, 包括类(classes)、关联(associations)、泛化(generalizations)、操作(operations)等。

(2) Application 应用模型

Application 模型对应于 MDA 中的平台特定模型 PSM, 关注的是 J2EE 平台。它包含 development 部件和 Integration 部件。把 PIM 模型作为输入, 然后根据基于模式的变换规则自动输出 PSM 模型, 它对应于 PIM 到 EJB PSM 的变换: 使用 UML 创建的 PIM 模型作为输入, 生成 Web 程序的中间层, 输出模型是通过使用特殊 EJB 构造型的 UML 变体表示的。

这里只使用销售管理模块中销售人员和客户订单之间关系的片断。直接在 OptimalJ 中创建 PIM, 如图 2 所示。

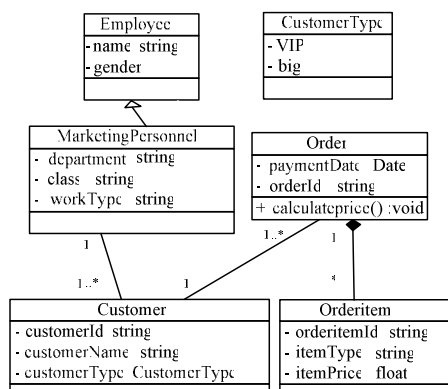


图2 系统功能片段的 PIM 模型

3.2 UML 到 EJB 模型的转换

模型转换定义了如何从一种源模型转换成另外一种目标模型^[3]。要实现这样的变换, 首先需要源语言和目标语言的元模型以便给出形式化的定义, 在变换规则中需要引用这些元模型中的元素^[4]。因此, 需要 UML 元模型和 EJB 元模型。基于源语言和目标语言的元模型, 转换规则采用基于 OCL 的规则约束, 采用基于模式的模型转换方法来完成, 从而实现 UML 到 EJB 模型的变换。

通过比较 UML 元模型和 EJB 元模型可以得到以下转换规则:

(1)UML 中的每个类都有一个到 EJB 模型中的主键类的变换。UML 模型中的每个类都会生成一个单独的主键类。

```
Transformation ClassToKeyClass(UML,EJB){
source
class:UML::Class;
target
keyClass:EJB::EJBKeyClass;
target condition
id.class=keyClass and
unidirectional;
mapping
class.name+'Key'<=>keyClass.name;
class.name+'ID'<=>id.name;
}
```

(2)每个关联都被变换为一个 EJB 关联。

```
Transformation
AssociationToDataAssociation(UML,EJB){
target dataAssoc:EJB::EJBDataAssociation;
source condition
assoc.end->exists(composition);
unidirectional;
mapping
assoc.end<=>dataAssoc.end;
assoc.getOuterMostContainer()<=>dataAssoc.packag;
}
```

(3)UML 属性到 EJB 属性的变换。

```
Transformation
UMLAttributeToEJBAttribute(UML,EJB){
source umlAttribute:UML::Attribute;
target ejbAttribute:EJB::EJBAttribute;
unidirectional;
mapping
umlAttribute.name<=>ejbAttribute.name;
umlAttribute.type<=>ejbAttribute.type;
}
```

4 实现

上述一系列的变换规则经过整理非形式化表达如下:

(1)对每个 PIM 类, 生成一个 EJB 主键类。

(2)每个 PIM 关联都被变换成一个 EJB 关联, 归入一个 EJB 数据模式, 这个 EJB 数据模式是从被变换的 PIM 类的最外组合 PIM 类生成的。

(3)每个 PIM 关联都被变换成为 2 个 EJB 关联和一个 EJB 数据类, 这些 EJB 关联和 EJB 数据类归入一个 EJB 数据模式, 这个 EJB 数据模式生成一个 PIM 类, 该 PSM 是从可以经过变换的 PIM 关联追溯的 PIM 类的最外组合。

上述每条转换规则都被建模成为一个转换模式, 比如与第(1)条转换规则 PIM 类到 EJB 主键类的转换, 对应的转换模式如图 3 所示, 其余的此处省略。

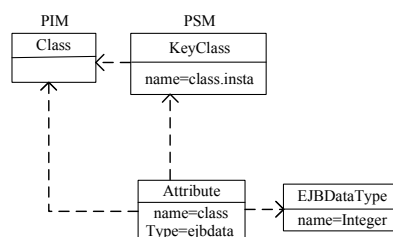


图3 与第(1)条规则对应的转换模式

(下转第 103 页)