从EDOC模型到J2EE程序:一个MDA工具的实现*)

林 嵩 赵建华 李宣东 郑国梁
(南京大学软件新技术国家重点实验室 南京210093)
(南京大学计算机科学与技术系 南京210093)

摘 要 模型驱动体系结构MDA是OMG组织推出的一种新的软件开发方法。根据MDA的框架,设计者首先建立平台无关模型PIM,然后遵循一定的转换规则转化成平台相关模型PSM,最后转化成目标平台上的代码。为了体现MDA低成本、高效率的优点,模型之间、模型代码之间的自动转换就显得尤为重要。本文描述了我们实现的一个MDA转换工具。该工具可以编辑用EDOC的Entities profile和Business Process profile构建的PIM,并且辅助自动转化成基于J2EE平台的PSM,最终转化成J2EE代码。

关键词 模型驱动的软件体系结构,EDOC,J2EE,自动转换工具

From EDOC Model to J2EE Program: an Implementation of a MDA Tool

LIN Song ZHAO Jian-Hua LI Xuan-Dong ZHENG Guo-Liang
(State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210093)
(Department of Computer Science and Technology, Nanjing University, Nanjing 210093)

Abstract Model Driven Architecture (MDA) is a new software development paradigm recently released by Object Management Group (OMG). According to MDA, the designer first builds a Platform Independent Model (PIM) of the system under construction, then transforms this model into a Platform Specific Model (PSM) according to some transforms rules, and/or generates code on the target platform. To increase the productivity and reduce the cost of MDA development, we should develop automatic tools for model transformation and code generation. A MDA transform tool is presented in this article. This tool can edit PIMs according to EDOC Entities profile and Business Process profile, and automatically transform the PIM to a PSM on J2EE. This tool can also directly generate J2EE code based on the PIM.

Keywords Model driven architecture, EDOC, J2EE, Automatic transform tool

1 引言

1.1 MDA的简单介绍

MDA(Model Driven Architecture)是OMG(对象管理组织)推出的一种新的软件开发方法。MDA的主要特点是将业务逻辑和应用逻辑与技术实现相分离,在体系结构上分离关注的焦点,从而提高软件产品的移植性、互操作性和重用性。在使用这个方法开发软件系统的时候,设计者首先建立一个基于UML(Unified Modeling Language,统一建模语言)的平台无关模型(PIM,Platform Independent Model)。PIM提供了关于系统的结构和功能的形式化规约。这个规约是独立于具体的实现平台的,并不涉及具体的实现细节。这里的平台可以指各种层次的平台,包括程序语言、操作系统、网络平台以及中间件平台,当然主要是征对中间件平台。软件系统最终需要在某个特定的平台上实现,与平台相关的实现细节必然要被加入到软件产品中去。在建立起系统的PIM之后,设计者可以遵循一定的转换规则将PIM转化成对应的、同样基于UML的平台相关模型(PSM,Platform Specific Model)。PSM

*)本文的研究工作得到国家863科技项目(2002AA116090),国家自然基金(6027036),江苏省自然基金(BK2002079)资助。林 嵩 硕士研究生,主要研究领域为软件工程。赵建华 副教授,主要研究领域为软件工程、形式化方法、模型检验。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。

参考文献

1 National Computer Security Center. Department of Defense Trusted Computer System Evaluation Criteria. DoD 5200. 28-STD, 1995. 12
2 The International Organization for Standardization. Information Technology Security Technology Evaluation Criteria for IT Security. ISO/IEC 15408-1,2,3,1999. 9
3 Common Criteria for Information Technology Security Evaluation. August 1999, Version 2. 1, CCIMB-99-031
4 中华人民共和国国家标准. 信息技术 安全技术 信息技术安全性评估准则. GB/T 18336. 2-2001
5 中华人民共和国国家标准. 计算机信息系统安全保护等级划分准则. GB 17859-1999
6 郭振民,胡学龙,姜会亮. 网络与信息系统安全性评估及其指标体系的研究. 现代电子技术,2003(9)
7 王志兰,赵怀勋,刘菲. 网络信息安全技术的研究. 现代电子技术,2003(8)

用目标平台的规范术语描述了系统的模型，这些术语包括：特定平台的异常机制、参数类型和构件模型等。软件设计者最后的工作是将PSM转化成相应的实现代码或者代码框架。关于MDA方法更加详细的描述见文[1]。使用MDA来开发软件可以得到以下的好处[3]：

a)缩短开发时间，节省开发费用。通过生成代码取代手工编写每个文件，避免了编写同一文件的重复劳动。

b)体系结构上的优点。由于系统的建模采用UML，这就必然使开发人员真正地考虑隐藏于系统后的体系结构和对象模型，而不是像大多数开发人员一样一头扎入代码中。

c)提高代码的一致性和维护能力。通过使用MDA工具依据一致的算法生成代码，使所有的开发人员使用同一设计模式。从软件维护的角度来看，这是一个很大的优点。

d)提高在不同中间件销售商之间的移植能力。可以重用PIM，生成不同中间件平台上的应用。开发人员可以从PIM为一个新的目标平台重新生成代码。尽管并不是所有代码都可以自动生成，但重新生成一个应用的大部分代码还是可以节省大量时间的。

MDA的开发过程实际上是一个以模型为中心，将高抽象层次模型逐步转化成为低抽象层次模型，并最终转化成代码的过程。开发过程中，开发人员所设计的PIM、PSM与代码之间都有对应关系。各个抽象层次模型之间的转换都遵循一定的规则和模式，并且这些转换的规则和模式是相对固定的，也就是说转换过程可以实现一定程度上的自动化。同时，实现不同层次模型(代码)之间的自动转换也是提高MDA开发效率的必要手段。MDA辅助工具在MDA设计过程中的好处和重要性主要体现在以下几个方面。

a)由于PIM、PSM和代码之间的对应性，在遵循一定规则和模式下进行的转换在某种程度上来说，是一种重复性的劳动。这种重复性劳动完全可以、也完全应该通过计算机自动实现。

b)只有通过MDA工具辅助PIM到PSM、PSM到代码之间的转换，才可能实现MDA的预先构想——节约开发时间，减少人工编码的工作量。

c)MDA辅助工具可以帮助确保PIM、PSM以及代码之间的一致性。传统的开发方式下，往往以代码为中心，模型处于辅助开发的地位。这就造成模型与代码之间的不一致性。或者是代码并不完全按照模型来设计，或者是代码已经修改了，而模型并没有同步的更新。MDA工具的辅助开发使得转换过程自动化，确保了模型与代码的一致性。

d)MDA辅助工具便于验证模型的正确性。模型之间的转换除了PIM到PSM，还包括代码到PSM，PSM到PIM，以及PIM到PIM，PSM到PSM，这一系列转换为验证模型的正确性提供了便利。

1.2 EDOC和J2EE的介绍

EDOC(Enterprise Distributed Object Computing，企业分布式对象计算)是ISO RM-ODP(Reference Model for Open Distributed Processing，开放分布式处理的参考模型)运用于企业规模的系统所得到的规范。它提供了一套递归协作的建模方法。该方法可用于不同粒度层次和不同耦合度层次的业务及系统的建模。EDOC能够同时支持松散连接和紧密的连接系统的规约，同时支持容器管理的体系结构与基于信息的体系结构中的同步通信和异步通信[4]。

EDOC Profile主要包括ECA(Enterprise Collaboration Architecture)和模式[3]。EDOC的核心部分——ECA是用于规约EDOC系统的MDA方法，包括五大元素：CCA(Component Collaboration Architecture)、Entities profile、Events profile、Business Process profile、Relationships profile。ECA的五个元素联合起来，分别从五个不同侧面，也即RM-ODP定义的五个视点(企业视点、信息视点、计算视点、工程视点和技术视点)对一个分布式的计算系统进行刻画。

ECA关注的焦点是企业规约、计算规约、信息规约。这些规约构建了一个EDOC系统的平台无关模型PIM。使用从技术相关模型得到的技术概念，软件设计者可以将这些规约进一步转化成平台相关模型PSM的工程规约和技术规约。

在我们的工具中，我们选用EDOC Profile中的Entities profile和Business Process profile作为软件系统的PIM的描述工具。Entities profile描述了用于对应用领域的实体对象建模的一组UML扩展，并且这些实体对象被定义为可组合的组件。Entities profile的目的是在技术无关的层次上，用实体的属性、关系、操作、约束和依赖来定义实体。Business Process profile描述了一组UML扩展。它与其他的EDOC profile联合起来后，可以对业务环境中的系统行为进行建模。Business Process profile提供了一组描述业务处理过程的建模概念，如业务动作的合成、实体执行某些动作的条件和实体间的通信与协作。Entities profile和Business Process profile联合起来，分别从静态和动态两个方面，共同定义了业务流程的结构和逻辑。设计者使用Business profile建立起业务流程模型。在业务流程模型中，当有一个系统输入的时候，处理流决定作为结果的动作什么时候发生，某个业务事件是否发生，以及参与的人将有什么动作。关于EDOC profile更加详细的描述见文[3]中。鉴于Entities profile和Business Process profile足以完整地刻画一个中小规模系统的静态特征和动态行为，我们只选用这两个profiles来描述PIM，分别用其中的Entities profile来描述Entities Model，用Business Process profile来描述Business Process Model。

我们目前选用的是J2EE作为我们的工具的目标平台。将来我们将考虑将工具的目标平台集合扩展到CORBA、J2EE、.NET等中间件平台。我们目前选用的是J2EE平台。J2EE(Java 2 Platform Enterprise Edition)是由SUN公司领导、各厂商共同制定并得到广泛认可的工业标准。J2EE为支持Java语言服务器端部署提供了一个操作系统平台无关的、可移植的、多用户、安全和标准的企业级平台。J2EE平台实际上是一个分布式的应用程序—服务器环境，提供了一个基于组件的方法来设计、开发、装配及部署企业应用程序。J2EE平台提供了多层的分布式应用模型、组件再用、一致化的安全模型和灵活的事务控制。J2EE规范定义了以下种类的组件：EJB组件、Web组件(包括Servlet和JSP组件)、应用客户组件和Applet。鉴于J2EE的这些优秀特性，以及它在工业界的广泛应用，我们选择J2EE作为我们工具生成的代码的目标平台。

2 EDOC模型到J2EE程序的转换过程

将平台无关的模型映射到具体的实现平台上的平台相关模型大体有两个方法[4]：

a)将一个模型转化成一个声明的集合。这些声明用选定技术平台的声明性语言来表述。比如说映射到CORBA平台的转换将会产生用CORBA IDL表述的声明。映射到Java平

台，将生成声明性的 Java 代码，也就是 Java 接口和抽象类。

b)将一个模型转化成另一个 UML 模型。转换得到的 UML 模型用选定技术平台的 UML profile 来表述。比如说 CORBA 的 UML profile 和 EJB 的 UML profile。

MDA 目前采用的是第2种方法。我们的工具也将采用这样的方法，即把一个按照 UML profile for EDOC 构建的 PIM (包括 Entity Model 和 Business Process Model)映射成一个按照 J2EE UML profile 构建的 PSM。映射的基本方式为：将 Entity Model 中描述的实体数据模型转化成为相应的数据库表格和 Entity Bean 类，并将 Business Process Model 中的每个业务处理节点转化成为相应的 JSP、Session Bean。

2.1 由 Entities Model 转换得到 Entity Bean 和数据库结构的规则

Entities Model 包含一个或者多个 Entity，每个 Entity 中分别包括实体名、属性、属性的类型以及主键。这里的 Entity 类似于面向对象中的类，但是 Entity 没有方法。例如，作为实体的邮件，包括邮件名、内容、回信的内容、关键字 id 号等属性，那么在 Entities Model 中邮件大致表示如图1所示。

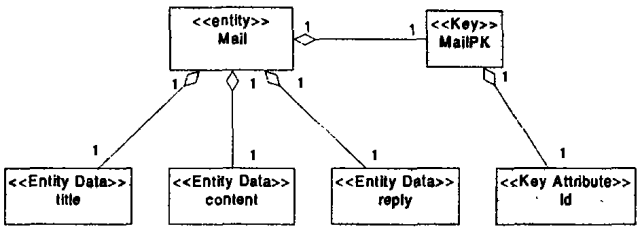

图1 邮件的实体模型

我们将每个 Entity 转化成一个相应的数据库表格，生成相应的 Entity Bean，并将 Entities 之间的关系转化成相应表格。下面的转换规则比较适合小型应用程序。一些复杂的应用需要更加灵活的转换方法，这是我们将来需要研究的目标。a)一个 Entity 转化成一个对应的数据库表格，实体名对应生成表名，属性对应生成字段名，属性的类型对应生成字段的属性；b)一个 Entity 同时也转化成一个对应的 Entity Bean，包括其 Home 接口，Remote 接口，Bean 类，Primary Key 类等；c)Home 接口定义生成、回收、查找 EJB 对象的方法，分别据 Entity 的属性对应生成 create 方法、remove 方法、findByPrimaryKey 方法和其他的 find 方法；d)Remote 接口复制 Bean 的业务方法，并根据 Entity 中每项属性对应生成 get/set 方法；e)Bean 类是 Entity Bean 的实现类，具体实现 Home 接口、Remote 接口中定义的方法。Bean 类中定义一些业务方法、每项属性的 get/set 方法以及一些 EJB 所必需的方法，如 ejbCreate、ejbFind、ejbLoad、ejbStore、ejbRemove 方法等。

2.2 由 Business Process Model 到 JSP，Session Bean 的转换规则

软件设计者依据业务的流程，将业务分解成一系列的业务处理节点，这一系列业务处理节点就组成了 Business Process Model。每个业务节点描述了整个业务流程中的一个环节。每个业务处理节点包含了节点名、输入实体、参与角色，以及处理逻辑等信息。工具在作自动转换的时候，还需要一些其他的信息才能够使得转换得到比较详细的 PSM 并最终生成代码。这些附加的信息包括：角色可以执行的处理动作、用户输入数据的方式、用户可见约束条件等。由于我们假设每个业务节点都由一个 Web 页面中完成，因此我们限制每个业务处理节点最多只有一个角色参与。对于没有参与角色的业务处理节点，其对应的处理过程将由计算机自动完成处理。如果原来的模型中的某个业务处理节点有多个角色参与，那么设计人员应当对该业务处理节点进行精化，进一步分解成一系列只有一个角色参与的业务处理节点。例如，一个市长信箱系统中，市民查看信件回复情况的节点大致如图2所示。

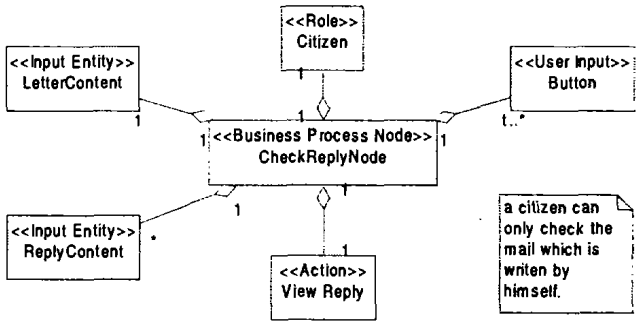

图2 市民查看回复结果的业务处理节点

对每个业务处理节点，我们将对应生成以下的 J2EE 组件：作为用户界面以及执行用户动作的 JSP 和作为用户控制的 Session Bean。一个业务处理节点对应生成一个 JSP 页面，大致的转换规则描述如下：

a)根据业务处理节点的输入实体 Input Entity，确定有哪些数据作为该节点的输入数据，并根据用户可见约束条件进一步确定哪些数据该用户是可见的；b)根据用户输入格式 User Input，生成对应的输入界面，如文本框、下拉框、复选框、按钮等。这些输入界面就决定了角色在该 JSP 页面上可以输入什么信息，以及以什么方式输入；c)根据动作生成相应的按钮，并根据动作的描述语言，解析其动作语义，生成相应的 Session Bean 或者 Java 代码。然后再用生成的按钮调用相应的 Session Bean 或者 Java 代码，实现动作的语义。

3 工具的实现与结构

我们的工具具有一个图形用户界面，主要是编辑、生成基于 UML 的平台无关模型 PIM，目前我们支持的是实体模型 Entities Model 和业务流程模型 Business Process Model。工具的主要部分是转换模块。该模块可以将包含了 Entity Model 和 Business Process Model 的 PIM 转化成基于 J2EE 平台的各功能组件。这个部分是我们将要在这一节中详细讨论的。这部分又按转换结果的功能划分成：生成用户管理代码的模块、生成 EntityBean 的模块和生成 JSP 的模块。为了便于讨论，我们作一些约定：MDA 工具生成的系统我们称为系统，其对应的用户称为用户。而 MDA 工具的用户就称为工具的使用者。

3.1 生成用户管理代码的模块

大多数 J2EE 上的系统都有用户管理功能，用户通过登录之后才可以使用这些系统的功能。同时在登录过程中，系统还将用户的有关信息保存在 Session 中供其他的系统功能使用。我们的工具可以根据工具使用者定制的信息，自动生成这些用户管理模块的代码。这些代码的功能包括：用户登录界面、添加用户、修改用户信息、删除用户等。当用户登录后，系统将在用户的 Session 中记录有关的用户信息。除了默认的用户名、密码、用户在系统中的角色这一属性外，MDA 工具的使用者还可以在工具中制定一些附加的信息。

软件设计人员可以使用自己手工编写的代码来替代由这

个模块生成的代码。当然，手工编写的代码必须能够在用户登录的 Session 中按照相应的格式存放必要的信息。

3.2 生成 Entity Bean 的模块

Entities Model 由一个或多个 Entity 组成，每个 Entity 对应生成一个 Entity Bean，主要是生成 Remote 接口、Home 接口和 Bean 类以及 Primary Key 类。例如对于2.1节中的实体——邮件 Mail，就生成 Mail 接口、MailHome 接口、MailBean 类以及 MailPK 类。下面对各接口及类的生成作具体讨论。

Remote 接口的生成：工具根据 Entity 的属性，对应于每一个属性，生成相应的 set 和 get 方法，并生成其他的业务方法。例如在 Mail 接口中，生成 gettitle()方法，返回类型为一个表示 title 属性的字符串；同时生成 settitle(String para)，返回类型为空。

Home 接口的生成：工具根据 Entity 的属性，将每个属性作为 create 方法的参数生成 create 方法，并为关键字属性生成 findByPrimaryKey 方法。例如 MailHome 接口中，生成 create(String title,String content,String reply,String id)，返回类型为 Mail 接口，以及 findByPrimaryKey(String id)，返回类型为 Mail 接口。

Bean 类的生成：前面 Remote 接口、Home 接口生成的只是方法的声明。具体的实现需要由 Bean 类中相应的方法来完成。首先，工具根据 Entity 的属性，将每个属性作为 ejbCreate 方法的参数生成 ejbCreate 方法，并在方法体中完成向数据库表格中添加一条记录的操作，插入的数据即 ejbCreate 方法中的参数，亦即 Entity 的属性；工具自动生成 ejbRemove 方法，并在方法体中完成删除一条记录的操作；根据实体模型中的关键字，工具生成 findByPrimaryKey 方法，并在方法体中实现查找关键字为方法所定义的参数的记录的操作；同样，对应每个属性，工具生成对应的 set 和 get 方法及其实现；然后，工具同样根据 Entity 的属性，生成 ejbLoad、ejbStore 方法；最后，生成其他的诸如 postCreate、setEntityContext、unsetEntityContext、ejbActivate、ejbPassivate 等方法。

3.3 生成 JSP 的模块

这个模块主要实现这个功能：生成用户进入某个业务处理界面的 JSP。

Business Process Model 由一系列业务处理节点组成，每个业务处理节点的结构如2.2所描述。根据2.2中的转换规则，MDA 工具遍历每个业务处理节点，将每个节点对应生成一个入口 JSP 和一个业务处理 JSP 页面：

a)工具从业务处理节点中，检索出输入实体的名字和该节点所涉及数据记录的关键字。对于这个输入实体，工具在这之前根据该实体的实体模型，已经在后台数据库生成了一张与输入实体同名的数据库表格。工具就在该数据库表格中，根据该节点所涉及数据记录的关键字查询出一个结果集，作为该 JSP 页面的输入数据。

b)工具根据用户可见约束条件，生成入口 JSP 页面。工具根据用户可见约束条件，在入口 JSP 页面中，主要生成一系列条件查询的 SQL 语句，实现业务处理节点中所定义的用户可见约束条件。工具生成的 JSP 可以对输入数据进行筛选、屏蔽，选取输入数据中的部分或全部数据，在浏览器上显示给用户。而用户可以点击他要处理的数据而进入业务处理 JSP 页面。

c)工具根据业务处理节点中的信息，生成对应的业务处理 JSP 页面。业务处理的完成，是通过完成业务处理节点中定义的一系列动作来实现的。工具将业务处理节点中的每个动作对应生成一个业务处理 JSP 页面上的按钮。并且，工具解析业务处理节点中的动作的语义，生成相应的 Session Bean 或者 Java 代码。用户在使用工具生成的应用系统时，点击按钮，通过按钮生成的 Session Bean 或者 Java 代码，完成业务处理节点中定义的动作，从而实现相应的业务处理。

d)同时，在工具生成的业务处理 JSP 页面中，工具可以根据定义的用户输入格式，在生成的业务处理 JSP 页面中加入文本框、下拉框、复选框、表单等。在工具生成的应用系统中，这些输入组件可以接收用户输入的参数，向系统提交数据。

在用户使用由 MDA 工具生成的系统的时候，用户登录后将进入第一个功能生成的登录页面。用户在登录后，系统罗列出该用户可以处理的业务处理节点。这些节点对应的 URL 指向处理这些业务的 JSP 页面。当用户选择了具体的某个业务之后，系统将进入该业务的入口 JSP 页面。用户选择相应的列表项就可以进入具体处理这个数据的 JSP 页面，也就是第三个功能生成的页面。

结束语 MDA 是未来软件开发技术发展的一个重要方向。MDA 的推广将会极大地提高软件开发的效率、降低开发成本，提高软件业的自动化程度。当然，在目前 MDA 的研究中，还存在不少难题有待于我们解决。比如，模型与模型之间、模型与代码之间的自动转换就是一个最大的难点，也是一个关键的重要环节。这些自动转换不仅包括 PIM 到 PSM、PSM 到代码的转换，而且包括 PSM 到 PIM、代码到 PSM，以及 PIM 到 PIM、PSM 到 PSM 等的转换。

我们的 MDA 工具是对 MDA 自动转换的一个尝试，已经可以生成一些较为简单的 J2EE 应用。但是，工具本身还存在一些不足，有待进一步的完善和改进。比如，目前工具使用的规则比较简单，所能够处理的系统的规模比较小。同时，规则的简单使得工具的灵活性受到一定限制。我们在下一步的研究中将重点考虑这些问题。最后，动作语义的解析，尤其是生成 Session Bean 这一块我们还需要进一步研究。

参考文献

1 Architecture Board ORMSC, Model Driven Architecture(MDA), OMG document number ormsc/2001-07-01. (July 2001). http://www.omg.org/cgi-bin/ doc?ormsc/2001-07-01.pdf

2 The Middelware Company. Model Driven Development for J2EE Utilizing a Model Driven Architecture(MDA) Approach Productivity Analysis. http://www.omg,org/mda/mda_files/ MDA_Comparison_TMC_final.pdf

3 UML Profile for Enterprise Distributed Object Computing Specification, OMG document

4 EDOC Part II v1.0, Annex E-Technology mappings from EDOC to Distributed Component and Message Flow Platform Specific Models