从 EDOC 的业务过程建模到 J2EE 应用程序框架的 MDA 转换*)

吴 光 赵建华 李宣东 郑国梁
(南京大学计算机科学与技术系 南京 210093)

摘 要 OMG(Object Management Group,对象管理组织)提出的模型驱动架构(MDA,Model Driven Architecture)是解决不同中间件平台之间的集成以及技术升级所引起的问题的软件开发方法。MDA 的基本思想是将系统的模型作为软件开发过程的核心制品,并且将模型区分为描述系统业务功能的平台独立模型(PIM,Platform Independent Model)和描述系统在特定技术平台上实现细节的平台相关模型(PSM,Platform Specific Model)。模型转换是 MDA 开发的核心内容,MDA 的开发效率依赖于良好的支撑工具,特别是支持模型转换的工具的涌现。本文介绍一种对业务过程建模并将模型转换成程序代码的设计思想和实现方法。开发者可以用该工具对企业的业务过程建模。这个工具能将建立的模型自动转换成 J2EE 平台上的程序代码。

关键词 模型驱动架构,平台独立模型,平台相关模型,业务过程模型,会话 bean

A Tool Designed for MDA Transformation from EDOC Business Process Model to J2EE Applications

WU Guang ZHAO Jian-Hua LI Xuan-Dong ZHENG Guo-Liang
(Department of Computer Science and Technology, Nanjing University, Nanjing 210093)

Abstract In 2001, OMG proposed Model Driven Architecture (MDA) to improve the interoperability and portability of software on different platforms. The essential idea of MDA is to put the application system models on the critical path of software development. According to the abstraction levels, there are two categories of models: PIMs (Platform Independent Models) which describe functional specifications without considering the platform details, and PSMs (Platform Specific Models) which concern the implementation details of the PIMs on specific platforms. The efficacy of transformation from PIMs to PSMs is critical to a project adopting MDA method. Powerful tools for model transformation are required for efficient MDA development. In this paper, we introduce a tool designed for MDA modeling and model transforming of business applications. Developers can use this tool to model business process of enterprises. This tool can automatically transform the models built by developers into codes on J2EE platform.

Keywords Model driven architecture, Platform independent model, Platform specific model, Business process model, Session bean

1 引言

MDA 是 OMG 提出的一种软件开发的方法学,也是一种基于模型的软件开发框架。与传统的以代码为中心的软件开发方法相比,MDA 通过将系统功能规约和实现规约分离的途径,降低了系统在不同中间件平台上集成和移植的代价。同时,MDA 提升了系统模型在整个软件生命周期中的地位和作用,模型不仅在开发的初始阶段(如分析、设计、代码生成),而且在维护、复用以及进一步开发的过程中,都扮演着基础性的重要角色。MDA 的提出,对于提高软件的生产效率、可移植性、可复用性、可维护性、互操作性、易集成性等方面,都会产生积极的影响。

基于 MDA 的开发过程包括 3 个关键步骤:第一步,根据系统的功能需求创建 PIM;第二步,选择实现系统的技术平台,然后利用模型转换工具,由 PIM 生成所选平台上的 PSM;第三步,利用转换工具将上一步得到的 PSM 转换为程序代码。其中,第二、第三步有可能一次完成。这里,由第一步建立的 PIM 的好坏是最终开发的软件能否成功的关键。按照 MDA 的设想,模型转换的工作最终应该由 MDA 支撑工具自动完成。研究从 PIM 到 PSM 的自动转换,对于实现 MDA 支撑工具具有重要的意义。

本文介绍一种对企业的业务过程建立 PIM 模型的方法,以及将 PIM 模型转换成程序代码的技术。我们的目标是设计一个支撑 MDA 方法的模型转换工具。它能帮助开发人员根据用户需求建立一个平台独立的业务过程模型。该工具还可以从平台无关模型出发自动产生程序代码。在我们的研究中,EDOC(企业分布对象计算,Enterprise distributed object computing)的一个子集——业务过程 Profile 和 UML 状态图被用作 PIM 的描述方法,而 J2EE 被选为目标平台。

2 PIM 的目标

PIM(Platform Independent Model)是 MDA 体系中两大核心模型之一,其目标是抽象出系统的业务逻辑。PIM 所要关注的是系统中的业务逻辑,而且也只关注业务逻辑,任何与实现平台相关的部分都被摒弃掉。这样就可以保证在平台迁移的软件再造过程中,我们对于系统业务的分析和建模的工作成果不会浪费掉,这些劳动成果在 PIM 中被有效地保存了下来。

PIM 具有很高的抽象层次,但抽象层次的提高并不意味着精确性的降低。PIM 对系统功能特征的描述越完整,通过模型转换得到的 PSM 越完整,软件生成过程中需要的人为干预就越少,软件开发效率就越高。因此,PIM 的描述方法应

*)本文的研究工作受到国家自然科学基金(批准号 60203009,60233020)、江苏省自然科学基金(批准号 BK2003408)和国家 973 项目(批准号 2002CB312001)资助。吴 光 研究生,主要研究领域为软件工程;赵建华 博士,副教授,研究领域为形式化方法、软件工程、程序设计语言;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。

该具有精确和完备的语义。

具体到我们的研究中，我们将使用 EDOC 作为 PIM 模型中业务过程模型的描述方法。UML Profile for EDOC(UML Profile for Enterprise Distributed Object Computing，以下简称 EDOC Profile)是由澳大利亚的“分布式系统技术中心”的 Pegamento 项目组制定的，并且是已经被 OMG 所吸收的一个建模规范。我们选用它的一个子集——业务过程 Profile，为业务过程建模。

另外，在现在的软件开发过程中，开发用户界面的工作量占了总工作量的很大部分。虽然用户界面具有很大的平台相关性，我们发现很多商务信息软件的操作过程可以用一个独立于具体实现平台的方法来描述。在商务信息软件中，完成某个功能的操作过程基本上通过如下的模式进行：用户通过一系列操作页面完成软件所需数据的输入之后，软件触发相应的处理过程；处理过程按照业务逻辑对用户的输入数据进行处理。每个操作页面的功能就是向用户显示一些有用的信息和提供用户输入数据的接口。我们选用 UML 状态图来描述 PIM 中用户界面的数据输入过程。

下面将介绍 EDOC 业务过程模型(business process model，简称 BPM)和 UML 状态图的语义。

3 PIM 描述方法的语义

3.1 EDOC 中业务过程模型语义

业务过程是一组相互关联(由特定规则控制)的功能[7]。业务过程管理注重的是过程的自动化。业务过程建模是实现业务过程自动化的手段和策略，它通过对业务过程进行明确的定义和表示，使其能够被计算机所支持和运行，并最终实现业务过程管理与控制。

EDOC 业务过程模型提供了用于对业务过程进行建模的元素，其中主要的建模元素有 BusinessProcess、CompoundTask 和 activity。每个业务过程通过以下几个方面来描述[8]：一个业务过程是若干个业务活动(business activity，以下简称 activity)的组合；选择进行活动的实体；activities 之间的通信和协调。CompoundTask 描述如何把一组相关的 Activities 连接协调，使它们能够完成一个比较大的工作。它表示了使用这个 CompoundTask 的 Activities 的类型和端口的协议。同时，CompoundTask 也是 Activities、数据流和它包含的 Activities 使用的 ProcessRoles 的容器。这里，ProcessRoles 表示的是 Activities 要求的对象的绑定。每个 activity 表示一个业务过程中一部分的执行。它执行的机制主要是下面两者之一：创建 activity 和执行绑定到该 activity 上的实体对象的一个功能。一个 activity 的动作如图 1 所示。

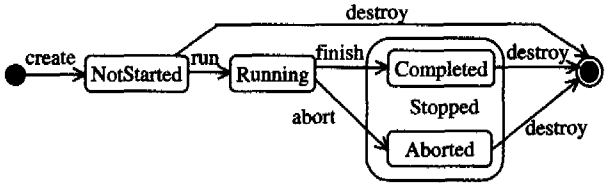

图 1 Activity 的动作

从图 1 中可以看到，一旦 activity 实例被创建，就进入 NotStarted 状态。等到适当的时候，如 activity 被调用且其执行的所有条件都已满足，activity 便进入 running 状态并立即处理数据。如果处理成功，就进入 completed 状态，否则进入 aborted 状态。最终，activity 会被撤销[8]。

我们在研究中假设每个 activity 通过如下的方式完成一个业务活动：用户通过一系列操作页面交互式输入数据，最后触发业务节点的处理过程，按照业务逻辑对用户的输入数据进行处理，最后完成业务的信息处理。我们选用 UML 状态图来描述用户操作页面间的迁移关系，以使得我们对 PIM 的语义描述更加完整。如果一个 activity 的完成方式不符合上面的假设，那么它对应的 PSM 就需要更多的人为干预。

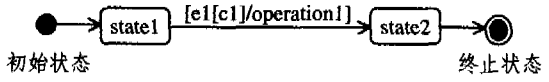

图 2 状态图

3.2 UML 状态图的语义

状态图用于对系统动态方面建模，强调从状态到状态的控制流[3]。我们使用状态图对用户界面建模，描述用户完成某个业务活动时的操作过程。这种图包含的元素有：状态和转换。

·状态

代表在对象生命周期间的某种状态。不同的状态代表了系统的不同行为模式。在我们的研究中，一个状态代表一个用户操作页面。

·转换

代表对象不同状态间的关系。从状态出发的转换定义了处于此状态的对象对外界发生的事件所做出的反应。通常，定义一个转换要有引起转换的触发器事件、监护条件、转换的动作、转换的源状态和目标状态。如图 2 的中间的转换，触发该转换的事件为 e1，c1 是该转换发生的条件。一旦 c1 满足，就执行操作 operation1。

1)触发器事件

触发器事件是引起转换的事件。事件可以包括信号、调用、时间段和状态的一个改变。在用状态图描述用户界面的时候，事件对应于用户和系统的交互过程中进入下一个操作画面的特定行为。

2)监护条件

转换可能具有一个监护条件，监护条件是一个布尔表达式。监护条件可以引用对象的属性值和触发事件的参数。当一个触发器事件被触发时，监护条件被赋值。如果布尔表达式的值为“真”，那么触发事件，即转换有效。反之，则不会引起转换。一个状态引出的多个转换可以有同样的触发器事件，但每个转换必须具有不同的监护条件。当其中的一个监护条件满足时，触发器事件会引发相应的转换。在用状态图描述用户界面的时候，监护条件是针对用户输入数据和会话数据的一个布尔表达式。

3)动作

当转换被引起时，它对应的动作被执行。常见的动作有赋值操作、算术计算、给另一个对象发送消息、调用一个操作、设置返回值、创建和销毁对象等等。动作也可以是一个动作序列，即一系列简单的动作。当状态图用于描述用户界面时，动作对应于对用户输入数据的保存/处理等动作。

有关 UML 状态图的详细资料可见参考文[3~5]。

4 业务过程模型到 J2EE Session Bean 程序框架的转换

实现模型自动转换需要两个前提：①作为转换输入的模型具有丰富语义；②高质量的转换算法。EDOC 领域建模语

言和 UML 状态图已经为 PIM 提供了准确和丰富的语义。下面我们通过业务过程模型到 J2EE 应用程序框架的转换规则来说明如何实现基于模型语义的模型转换，在这之前首先简单介绍一下 J2EE 应用程序框架。

4.1 J2EE 概述

J2EE 旨在为支持 Java 语言服务器部署而提供平台无关的、可移植的、多用户的、安全的和标准的企业级平台[1]。它的整体架构是一个多层结构，包括：用户层、Web 层、业务层和 EIS 层[2]。

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，包括 HTML 用户、Java Applet 或 Java 应用；Web 层产生表示逻辑，并接受客户端的用户反馈，这些用户通常是 HTML 客户端。该层由 jsp 或 servlet 来实现。业务层处理系统的业务规则，一般由 EJB(Enterprise JavaBeans)来实现；EIS 层主要包括：后台数据库系统、遗产系统、企业资源计划系统等[2]。

J2EE 的核心是 EJB[1]。EJB 组件可以用于封装业务逻辑，并使应用程序开发者不需要担心许多系统级问题，如事务、安全性、可扩展性、并发性、通信、资源管理、持续性、出错处理。一个 EJB 组件由一组 Java 类和 XML 配置文件组成。其中 Java 类包括：一个 Home 接口、一个 Remote 接口和一个实现接口的 Bean 类。EJB 组件主要分为 3 种类型：会话 Bean、实体 Bean 和消息驱动 Bean。会话 Bean 用于表示工作流、业务逻辑和应用程序状态，每次由单个的客户使用。会话 Bean 又细分为两种类型：有状态的(stateful)和无状态的(stateless)。有状态会话 Bean 可以在客户访问期间保存数据，无状态会话 Bean 不能保存这些数据。实体 Bean 可以看作是数据库的面向对象的数据表示。与数据库相似，它可以同时由多个客户访问。一个实体 Bean 可以代表一组数据，比如一个顾客或一种产品。消息驱动 Bean 是被异步调用的，可以通过 Java Messaging Service(Java 消息服务，JMS)提供者来接受和处理 JMS 消息。典型情况下，客户会向一个专门的 JMS Queue 或者 Destination 发出一条消息，而所有订阅此目标的消息驱动 Bean 都将接受此消息。

以上的内容详见参考文[1,2]。

4.2 模型转换规则简述

当我们用 EDOC 中的业务过程建模元素和 UML 状态图建好 PIM 后，就可以借助于 BPM(业务过程模型)到 J2EE 应用程序框架的转换规则实现模型的自动转换。为了保证源模型和目标模型在语义上一致，我们在元模型层次定义基于语义的 BPM 到 J2EE 的转换规则，在执行转换时根据转换规则建立 BPM 和 J2EE 的语法映射。这里主要有两种转换规则：BPM 中每个 activity 到 J2EE 的会话 bean 的转换和描述 activity 行为的状态图到用户界面的转换。这里的用户界面包括多个负责显示信息或接受用户数据的操作页面，而用户界面中每个页面的布局由另外的工具定义。

4.2.1 activity 到会话 bean 的转换

一个 activity 生成一个对应的 session bean，包括 home 接口、remote 接口、bean 类。其中，home 接口中至少包含一个 create 函数。该 sessoin bean 负责维护用户界面的操作过程和业务处理逻辑的实现。根据 activity 的状态图进行如下转换：

• session bean 实现状态图中的每个转换。

1)如果从状态 A 出发有一个或者多个(内部/外部)转换的触发事件为 event，那么生成一个函数 A_event，同时在 remote 接口中生成一个跟该函数对应的接口。假设从 A 出发，触发事件为 event 的转换为：event[$condition_1$]/$operation_1$，event[$condition_2$]/$operation_2$，…，event[$condition_n$]/$operation_n$，那么函数 A_event 的功能为首先判断 $condition_i$ 中那个条件是否成立。若成立，则执行相应的操作，并完成状态的转换。

2)对于状态 A 的入口操作 entry/operation，则在 bean 类中生成一个函数 A_entry，并在函数内实现 operation，同时在 remote 接口中生成一个跟该函数对应的接口。入口操作通常用来进行状态所需要的内部初始化。我们规定一个状态只能有一个入口操作。

3)对于状态 A 的出口操作 exit/operation，则在 bean 类中生成一个函数 A_exit，并在函数内实现 operation，同时在 remote 接口中生成一个跟该函数对应的接口。出口操作通常是从一个状态离开时进行的善后处理工作。我们也规定一个状态只能有一个出口操作。

• 我们可以在状态图的初始状态中定义入口操作，负责完成整个 session bean 的初始化操作。同样，我们也可以在状态图的终止状态中定义出口操作，负责进行整个 session bean 的善后处理。

• 对于状态转换，其实现方法如下：session bean 中有一个变量存放 session bean 的当前状态，变量的值集包括状态图中所有状态和一个 accepted 接受状态的集合。当该变量的取值为 accepted 时，表明本次 activity 的用户界面操作部分结束，软件将调用相应的过程来完成业务逻辑处理。

根据上面的转换规则进行转换后，只是生成一个 PSM(代码)框架，因为没有对 activity 中间的业务逻辑进行精确定义。开发者必须手工编写完成业务逻辑处理的程序。

5 用户界面状态图到操作页面的转换

5.1 有关 action 信息的描述

需要说明的是：我们的状态图只是定义了用户操作页面间的迁移关系和在进行相应的迁移时所要进行的业务处理。用户操作页面中元素的布局和显示方式由另一个操作页面定义工具定义。该操作页面定义工具根据 activity 的状态图信息自动产生若干个用户操作页面原型并分别进行设计。该工具还将生成每个页面的摘要信息。一个摘要信息包括：名称、入口参数、要显示的信息、待输入的信息和一组 action 按钮描述信息。其中，action 按钮的格式为：

action 按钮＝(ID 号、名称、一组提交参数，触发器事件，提交的数据)

其中的 ID 号在该用户操作页面内唯一。在进行模型转换之前，我们还需要描述这些参数是如何进行传递和处理的。这些描述不是 PIM 的一部分，而是由模型转换中开发人员的设计决定。最后，所有有关 action 的描述信息都以 action 按钮为单位保存到多个 XML 文件中。每个 XML 文件的名称为用户页面名＋action 的 ID 号。图 4 和图 5 分别给出了生成 XML 文件的算法和 XML 文件中元素的说明。

```
输入：业务模型中所有 activity
输出：action 按钮对应的 XML 文件
For 每个 activity DO
  For activity 中的每个用户界面 DO
    For 用户界面中的每个 action DO
      创建并打开一个 XML 文件，文件名＝界面名＋action 的 ID 号
      将该 action 按钮提交的参数写入 XML 文件
```

将转向的目标 form 以及其入口参数和 action 按钮提交参数的对应关系写入 XML 文件
将 action 按钮触发时要调用的操作以及操作所在的 session bean 写入 XML 文件

图 3　生成 XML 文件的算法

Ⅰ 〈actualParam〉元素——action 按钮提交的参数，即实参，相关属性说明如下：
　Ⅰ.1 get：取值为 url，表示该参数从通过 url 传递过来；session，表示该参数从 session 中传递过来
　Ⅰ.2 pass：yes，表示该参要放到 session 中；否则，取 no
　Ⅰ.3 function：若该参数对应的值要暂放到 session bean 中，该 function 取值为 session bean 中某个函数的名称，注意，该 session bean 不是 activity 对应的 session bean，而是专门用来存放用户数据的，由设计者制定；该属性可以不出现。
　Ⅰ.4 key：将参数对应的值放到 session bean 时对应的关键字，其与 function 同时出现。该属性可以不出现。这里的 session bean 和上面的是同一个 session bean。
Ⅱ 〈targetForm〉元素——action 按钮的目标操作界面，包括：一个属性 tgtFormName，保存目标操作界面的名称；若个子元素〈formalParam〉
Ⅲ 〈formalParam〉元素——目标操作界面的形参，包括一个属性 actualParamByRef，指向实参
Ⅳ 〈operation〉元素——action 按钮被触发时要调用的 session bean 中的操作，属性 name 指向操作名称，属性 sessionBean 指向操作所在的 session bean。若干个子元素〈funcParam〉
Ⅴ 〈funcParam〉元素——操作的实参，实参名与 actualParam 元素的实参名对应。

图 4　XML 文件中元素的说明

5.2　状态图到 jsp，HTML 页面的转换

我们使用一个解释器来完成用户界面中操作页面之间的迁移关系。在我们的研究中，描述用户界面的状态图中所有的状态都假设为简单状态，既不包含子状态，也不考虑延迟事件。状态的入口/出口操作会在进入/离开对应操作页面的时候自动调用。首先说明一下状态图和用户操作页面之间的对应关系。

1)描述用户界面的 UML 状态图中每个状态对应一个操作页面，状态的名称对应于一个同名的 jsp 操作页面。这个操作页面由我们开发的另外一个工具生成。同时生成的还包括有关这个操作页面的摘要信息：操作页面入口参数、要显示的信息、需要用户输入的信息和一组提交信息 action 按钮。每个 action 按钮在操作页面内有唯一的 ID。

2)每个离开 S 状态的转换，包括内部转换，其上都标记有：event[condition]/operation。在这些转换中出现的每个 event 和 S 操作页面上的一个 action 按钮相对应。这里是一个 event 而不是一个转换对应到操作页面上的一个 action 按钮。当用户按下某个按钮时，系统将根据当前情况和转换上的[condition]标记，触发不同的转换而进入到不同的画面。

我们使用一个解释器来完成用户界面中操作页面之间的迁移关系。实现的基本思想如图 5 所示。所有 action 按钮的请求都提交给一个专门的用 jsp 编写的解释器。该解释器的主要任务包括：首先，根据 action 按钮的 ID 号和所在操作页面的名称找到特定的 XML 文件，并从该文件读出业务规则信息；jsp 解释器根据这些信息作出相应处理：首先调用相应 session bean 中的某个操作，然后再转向目标操作页面并按照约定传递相应的参数。这里，每个 action 按钮至少要向解释器提交两个参数：action 按钮的 ID 和 action 按钮所在操作界面的名称。另外，XML 文件中保存的是界面转换规则信息，而 XML 文件的名字由操作界面和 action 的 ID 号按照特定规则组合而得到。有关 XML 文件的内容的详细说明见图 4。解释器处理 action 请求的算法如图 5，其中的一些符号的说明见图 4。

输入：源操作界面名称和 action 按钮的 ID 号，以及其他的一些参数
输出：由 action 语义决定，通常是转向目标操作界面
将源操作界面名＋action 的 ID 号组合成 XML 文件名；
If 该文件不存在
　　报错，如何处理可由设计者决定；
else
　　解析该 XML 文件，获取与该 action 相关的业务规则信息，包括：
　　1. 将所有实参 actualParam 的信息放入一个实参列表 actualParamList；
　　2. operation 的值放入一个 operation 对象，其有一个列表 opParamList 保存该 operation 的所有形参；
　　3. targetForm 的值放入一个 targetForm 对象，其有一个列表 tgParamList 保存该 targetForm 的所有形参；
　　for 每个实参 actualParam in actualParamList Do
　　//从 url 或 session 数据中获取 action 按钮提交的参数的值
　　if get＝＝“url”按实能名从 url 获取它的值；
　　else if get＝＝“session”
　　按实参名从 session 获取它的值；
　　if pass＝＝“yes”
　　将该实参以名/值对的形式放入 session；
　　//将提交的参数放入用户设计的 session bean 中。
　　if function！＝null&&function.length()！＝0
　　获取 key 的值，然后将该实参以 key 的值为关键字，
　　并用 function 函数将其值放入某个特定的 session bean；
　　if operation！＝null
　　for operation 的每个形参 opParam in opParamList Do
　　根据 opParam 的取值，从实参列表中找到对应的实参，
　　将找到的实参的取值作为该形参的实参值；
　　根据上面得到的形参值，调用 operation 的 sessionBean 指向的 session bean 的 operation 的操作；
for targetForm 的每个形参 formalParam in tgParamList Do
　　根据其 actualParamByRef 的取值，从实参列表中找到对应的实参，将找到的实参的取值作为该形参的实参值；
根据目标操作界面的名称和上面得到的目标操作界面形参值，转向目标操作界面；

图 5　jsp 解释器处理 action 请求的算法

结束语　本文介绍了一种对企业的业务过程进行建模并将模型转换成程序代码的设计。用户可以使用我们的工具根据需求建立一个平台独立的业务过程模型。然后，我们的工具能根据该模型自动产生程序代码。我们的工具可以使开发者在开发时把精力集中于对需求的满足，而无需考虑最终的实现平台和技术。使用该工具的好处有两个方面：一方面，使开发出来的软件能更好地满足用户需求；另一方面，用户不需要太多的有关平台技术的知识就可以完成开发。

我们知道，最初提出 MDA 的目的是解决不同中间件平台间的互操作问题。目前，我们的工具仅仅以 J2EE 为目标平台。所以，我们的下一步工作是：一方面，实现到其他目标平台和实现技术的转换；另一方面，研究并解决不同中间件平台间的互操作性问题。到时我们就真正达到了“一次建模，多次使用”，从而达到缩短开发周期、降低开发成本、简化实现难度的目标。

事实上，我们的工作是作为我们的研究小组研究的 MDA 系列工具中的一个有机组成部分而存在的。与 MDA 中的实体定义工具以及操作界面定义工具结合，我们的工具可以为用户提供更加强大的功能。

参 考 文 献

1　刘晓华，等(译). 精通 EJB(第二版). 北京：电子工业出版社，2002
2　马树奇(译). J2EE 编程指南(1.3 版). 北京：电子工业出版社，2002
3　常晓波(译). UML 技术手册. 北京：中国电力出版社，2002
4　邵维忠，等(译). UML 用户指南. 北京：机械工业出版社，2001
5　夏昕，何克清(译). UML 业务建模. 北京：机械工业出版社，2004
6　鲍志云(译). 应用 MDA. 北京：人民邮电出版社，2003
7　企业内部业务流程管理. http://www-900.ibm.com/developerWorks/cn/wsdd/support/redbook/SG246173/index.shtml
8　UML Profile for Enterprise Distributed Object Computing Specification. http://www.omg.org
9　MDA Guide. http://www.omg.org/docs/omg/03-06-01.pdf
10　MDA Specification. http://www.omg.org