从 Executable UML 模型到 J2ME 程序*)

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

摘 要 嵌入式系统由于功能的特定性和底层硬件的多样性,使得其代码的重用和易移植性一直是困扰开发者的一大难题。模型驱动体系结构(MDA)是OMG组织提出的一种新的软件开发方法。MDA将关注点集中于业务模型,把平台相关内容和不同的实现技术从中剥离,并利用工具遵循一定的转换规则,实现其到特定平台的自动转换,最终得到目标平台上的代码。MDA针对解决复用和移植问题提出了新的解决途径,在嵌入式系统开发中有广泛的应用前景。本文在研究基于MDA的嵌入式系统开发途径的基础上,提出了一个从Executable UML模型到J2ME平台下JAVA代码的转换框架。

关键词 模型驱动的软件体系结构,J2ME,嵌入式系统,可执行UML,转换规则

From Executable UML Model to J2ME Program

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

Abstract Because of the specific function and the variety of hardwares, the reusability and portability of the code has been a big problem in the embedded system development. Model Driven Architecture (MDA) is a new software development method released by Object Manage Group (OMG). MDA focuses on the business model which is independent of any platforms and implementations, and transforms this model to specific platform's model automatically, then to generate the code of the target platform finally by using tools according to the transform rules. Because MDA proposes a new approach to solve the problem of reuse and portability, it has a wide application perspective in the embedded system development. This article introduces a transform frame from Executable UML model to J2ME platform and presents how to generate JAVA code.

Keywords Model driven architecture, J2ME, Embedded system, Executable UML, Transform rules

1 引言

近年来,随着微处理器和各种硬件成本的大幅下降,以及消费类电子产品的普及,基于并发和实时的嵌入式系统得到了大规模的发展。但由于产品更新速度加快,产品的研制周期缩短以及新硬件的层出不穷,在系统的开发和研制上却遇到了一系列的瓶颈。传统的分析设计方法,如实时结构化分析和设计方法(RTSAD)、实时系统的设计方法(DARTS)以及Jackson系统开发方法(JSD)等已经很难直接适应这种现状。嵌入式系统对特定硬件平台的依赖性及其实现功能的特定性,使得可移植性和代码的复用成了开发过程中的突出矛盾。而开发周期却越来越短,开发者迫切需要一种新的开发方法和过程来解决这一些问题。

MDA以一种全新的方法来整合这一系列开发技术和趋势,它并不排斥传统技术,而是扩展和推动这些技术进行更好的协作,来面对全新的挑战。MDA将关注点集中于业务模型,把平台相关内容和不同的实现技术从中剥离,并利用工具遵循一定的转换规则,实现其到特定平台的自动转换,最终得到目标平台上的代码。MDA针对解决复用和移植问题提出了新的解决途径,在嵌入式系统开发中有广泛的应用前景。

模型转换的自动化和标准化是MDA得以普及和体现其巨大生产力的关键。问题域的一次建模、由各种自动化工具将其转换到各个实现平台的方法,展现了诱人的前景。但只有通过研究各个目标平台的特性和相应的平台转换规则,并最终实现转换规则的标准化,才能实现各种模型自动转换工具。因此,模型转换的研究是当前MDA发展中的重点和热门问题。

目前的MDA模型转换研究多集中在静态结构模型转换方面,并取得了长足的发展,但在行为模型的转换方面还有很多欠缺。Executable UML是UML的一个Profile,由于UML最初是作为一种建模语言而存在,在动态精确建模等方面存在缺陷,不能直接由模型生成完整的程序,因此一些团体和公司(如Kennedy carter和Nucleus)在UML1.4的基础上进行剪裁,并加入了精确行为语义,形成可执行UML。它可以用于精确的建模系统,并可直接将模型转换成完备的目标平台代码。因为Executable UML能够详细和完整地定义系统的静态和动态模型,所以我们这里选用它作为建模语言。J2ME是SUN公司推出的一个面向消费类电子产品和嵌入式设备的应用程序开发平台,可动态扩展、代码的可移植性、安全的网络传输等是它的特点,目前在各种通讯产品上有广

*)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)的资助。袁 梁 硕士研究生,主要研究领域为软件工程;李宣东 教授,博士生导师,研究领域为形式化方法、模型检验;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;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。

泛的应用。本文在研究基于 MDA 的嵌入式系统开发途径的基础上，提出了一个从 Executable UML 模型到 J2ME 平台下 JAVA 代码的转换框架。本文的第 2 部分介绍了嵌入式系统的特征和嵌入式开发平台 J2ME；第 3 部分给出了 MDA 和 Executable UML 的一些基本概念，并说明了 Executable UML 在精确建模方面的优势以及嵌入式建模的特殊性；第 4 部分给出了从 Executable UML 建立的 PIM 模型到 J2ME 平台模型转换步骤，并描述了状态机和类到 JAVA 代码的生成方法。

2 嵌入式系统

2.1 嵌入式系统的特征

嵌入式系统是以应用为中心，软硬件可剪裁，对功能、可靠性、体积、功耗、成本有严格要求的计算机系统。实时性是嵌入式系统的基本特征，而大多数嵌入式系统都具有并发处理的要求，所以嵌入式系统也可称为带有时间约束的并发系统。

实时系统也可分为硬实时系统和软实时系统。硬实时系统在时间上具有严格的最后执行期限，这个约束必须遵守，否则会有灾难性的系统错误。而软实时系统虽然要求时间约束，但偶尔超过时限也不会导致灾难性的后果。

由于嵌入式系统的上述特征，在构建系统的时候就会遇到如下问题[4]：

a) 硬件的多样性：比桌面系统更多的 CPU 种类和系统架构、以及各种不同底层硬件，带来了软件可移植性问题。

b) 安全性：越来越多的设备要求接入网络，这要求考虑安全性因素。

c) 系统正在逐步变成分布式的，要求更灵活的体系结构。

d) 消费类产品的生命周期有很长的跨度，其中要应对不断的发展变化，需要有灵活的可扩充性。

这些因素促进了嵌入式实时操作系统和各种嵌入式开发平台的发展。目前流行的通用型嵌入式操作系统如 Windows CE、VxWorks、嵌入式 Linux 等，专用型嵌入式操作系统如 Palm OS、Symbian 等。

2.2 J2ME 开发平台

J2ME 是 SUN 公司推出的一个面向消费类电子产品和嵌入式设备的应用程序开发平台。它的目的[4]，是提供一个易于理解的应用程序开发平台，用来创建可动态扩展的、网络化的设备和应用程序。它主要针对两大类产品：高端消费类产品和低端消费类电子产品。

J2ME 架构定义了三个基本概念：

a) Configuration 定义了一个最精简的公共平台，以及在这个平台上设备都适用的 JAVA 语言设施和虚拟机特性以及最基本的类库。

b) Profile 是处于 Configuration 之上的扩展。它是一组相似设备特定需求的细化，如手机家族等。Profile 包含一些类库以及一系列的标准接口，任何支持 J2ME 的设备厂商都要为其设备实现定义的接口。

c) Optional Package 是在 Profile 层之上的一系列 API，它的目的是让这些功能包可以按照需要应用在各种 Profile 之上。

MIDP(Mobile information device profile)是位于 CLDC (Connected, Limited Device Configuration)之上一个为低端消费类电子产品定制的 Profile，典型的 MIDP 的目标设备如手机和寻呼机等。MIDP 没有实现 Applet 程序模型，而是采用新的应用程序模型 MIDlet，是一种可执行和通信的应用程序。MIDlet 是 MIDP 的基本执行单位，一个或多个打包在一起的 MIDlet 构成了 MIDlet Suite，它是部署的基本单位。

3 MDA 和嵌入式系统建模

3.1 MDA 简介

模型驱动体系结构(MDA)是对象管理组织(OMG)提出的一种基于平台无关模型的、新的描述系统规约的方法。它希望以一种开放式的架构，用新的方式来整合一系列软件开发新技术以及处理企业集成问题。

MDA 关注的焦点集中在描述一个系统功能和行为的模型，而不关注这个系统在何种平台下开发以及用何种技术来实现系统的功能。MDA 通过关注点转移的方法将实现的技术细节从业务模型中剥离开来，从而提高了业务模型的抽象层次。因为与实现细节无关，所以当出现新的技术和新的平台的时候，开发者就无需对功能和行为部分再次建模，减少了重复劳动。

MDA 的核心思想以模型为中心，根据参考平台的不同，模型可以分为平台无关模型(PIM)和平台相关模型(PSM)。当开发者应用 MDA 来构建一个应用系统时，开发者首先选择合适的 UML Profile 来创建平台无关模型 PIM，然后由平台专家遵循一定的转换规则把 PIM 转化成平台相关模型 PSM。这一部分在两个平台间建立标准化的映射规则之后，可以生成映射工具，由工具自动在两个模型间自动转换，最后转化成目标平台上的代码。

MDA 应用的两个关键环节是 PIM 的生成和平台间转换。如果有成熟的工具来实现模型间的自动映射，MDA 将爆发出巨大的生产力。目前 OMG 致力于平台间映射的标准化工作。这项工作需要有适合系统不同方面的、不同抽象层次的语言定义能力，这是 MDA 的应用基础。OMG 提出了元对象设施(MOF)和 UML Profile，这是 MDA 的语言定义机制。MOF 用一种统一方式来描述不同类型的建模结构，可以定义不同抽象层次模型的元模型，这使平台间映射的标准化有了坚实的基础。OMG 在此基础上提出了一系列的标准化映射，如 MOF-CORBA，JMI，XMI 等。

MDA 中模型间映射可分成如下几类：

a) PIM 到 PIM。如分析模型到设计模型的转换。通常，PIM 到 PIM 的映射与模型的精化相关联。

b) PIM 到 PSM。当 PIM 需要在某个特定平台或特定技术实现时，就需要这种映射，如组件的逻辑模型到 J2EE 下的 EJB 映射。

c) PSM 到 PSM。通常在特定平台下模型的精化时，需要这种映射。

d) PSM 到 PIM。当需要从特定平台下的模型抽象出业务模型时，需要这种映射。

用模型驱动架构来开发系统有很多优点，如提高软件的复用和移植性、减少重复的编码工作、保证开发过程中代码和文档的一致性等。但 MDA 目前还处于发展的初级阶段，在映射的标准化和自动化方面还有一些问题需要解决。

3.2 Executable UML 介绍

UML 不是一个完备的可执行建模语言，其最初的产生是为了面向对象建模，所以主要用于描述开发系统的两个方

面[5]：

a) 描述系统需求。用例图和顺序图；

b) 描述系统软件模。类、关联、属性、操作、状态机等。

因此，UML 用来描述系统的静态模型是充分的，但要生成一个可执行系统却是不足的。

因为，a) 缺少描述操作内部实现的机制；b) 状态机内部语义部分缺乏详细的实现机制，如没有属性、关联的操作功能；c) 关联的部分没有动态机制，只考虑了静态关联。比如两个类学校和学生，它们之间可以有两种关联：一种是所有在学校上过学的学生，一种是现在正在学校上学的学生。这两种关联一个是静态的，一个是动态的，UML 缺少这部分表达动态关联的机制；d) UML 是广泛的建模语言，它为了支持开发过程的各个方面，融入了各种元素。而建模过程中的建模元素的选择和不同阶段开发目的的不一致性，会带来模型本身的歧义。

由于 UML 的上述不足，Executable UML 作为 UML 的一个 Profile 引入，用来创建一个明确的、完善的可执行系统。目前有多家厂商都推出了自己的版本，如 Kennedy carter 公司的 iUMLite 和 Nucleus 公司的 BridgePoint。这些建模系统各有其特点，但整体上而言，各个 Executable UML 版本都采纳了 UML1.4 的一个子集并加入了精确行为语义相容的行为语言(Action language)作为自己的建模系统。

Executable UML 从三个方面来刻画系统：用静态的类图来描述事物的数据，包括类、属性、关联和约束；用动态的状态图来描绘事物的生命周期，包括状态、事件、状态转移和过程；用行为语言来描述事物各个阶段的算法和实现细节。

类图中的类只包含了其属性和方法，却没有方法的具体实现细节。OMG 已经认识到这方面的不足，所以在 2001 年已经采纳了精确行为语义作为 UML 的一个组成部分。这些语义包括对象的创建和删除、属性和联接的操纵、逻辑和数据计算、数据传输和迭代等等内容。精确行为语义提供了动作语言的元模型，提供了其抽象语法，但是没有定义动作语言的具体语法，所以现在有很多与其相容的行为语言，如 Kennedy carter 公司的 ASL，Nucleus 公司的 Object action language。

3.3 嵌入式系统建模的特殊性

嵌入式面向对象建模需要考虑并发和实时问题，因此引入了主动对象和被动对象。一般企业建模较多地关注信息隐藏和分布，一般的类都是被动类。而嵌入式应用一般由多个任务协作，引入主动类，主动对象拥有自己的控制线程，能够影响被动类。企业建模偏重于持久性对象，而嵌入式开发中大量对象是“暂时”的、有状态的，更注重其动态特性，因此建模中注重对象的状态分析。国外大量研究应用表明，用状态机可以精确地刻画一个嵌入式系统。

UML 建模中，状态图一般用来揭示复杂对象的活动和寻求其内在方法，作为一种分析和设计的辅助模型(很多系统分析完全不用状态图)，包括了嵌套状态、历史等复杂表示法，并且消息传递的一些表示也不是形式化的，带有随意性。因此，Executable UML 定义了一个精确状态机模型，加上行为语言，可以完整地描述一个对象的生命周期和其相关行为。因此嵌入式建模中，其动态模型是以状态图为其中心的。

状态机由四个部分组成：状态、事件、状态转移、过程。并且在任一特定时刻，只有一个状态对应一个过程在执行。每个状态机对应一个状态转移表，控制事件接受和相应的状态转移。状态机的应用主要在以下方面：一个类如果存在生命周期，就可以用状态机来表述。当两个类之间存在多对多关系时，一般引入一个关联类和关联对象的状态机，来控制相关联对象的联接创建和删除。对竞争资源，引入关联类状态机，这种类状态机只有一个实例。

嵌入式建模过程也和传统企业级建模过程有所不同。企业应用建模[3]从需求分析到找寻分析类过程中，倾向于将核心抽象类往三个方向扩张成边界类、控制类和实体类，得到分析类，再通过分析阶段设计模式和设计机制的实施，得到最后的类模型。但这样的效果就是业务流程集中到控制类中，控制类往往比较复杂。这样的后果就是控制类的状态机非常复杂，不可控制。所以嵌入式建模不采用这种方法，甚至提倡不要一个控制类[1]，通过忘掘类之间的关联、多对多的关联引入关联类，通过“pulled”和“pushed”方法将其控制在相关类之间流动，分散职责。MDA[2]的开发过程中也注重这点，其中的订餐系统的 PIM 模型只反映了其实体模型和实体间关系。

4 从 Executable UML 模型到 J2ME 代码

用模型驱动的方法进行嵌入式软件开发，在国外已经有了一些应用，也出现了一些比较成熟的软件，如 Kennedy Carter 公司的 iUmlite 和 Nucleus 公司的 BridgePoint。它们可以支持嵌入式软件开发的整个生命周期，并且能生成完整的可执行的 C 或 C++ 代码。下面将介绍从 Executable UML 建立的 PIM 模型到 J2ME 平台下 JAVA 代码生成的一种方法。

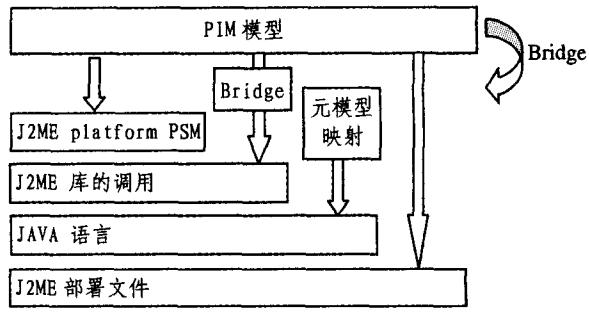

图 1

在开发过程中，通过关注点转移、去除不相关细节的方法，我们可以得到不同的业务域(Domain)，然后通过这些域的交互实现用户的需求。域之间是相互依赖的，而它们内部却是高度耦合、自治的，它构成了系统重用的基本模块。不同的业务域用 Executable UML 建模后，就得到了我们需要的 PIM。为了构建完整的系统，我们还要引入各种实现相关的域，如 J2ME 中的类库可以作为 realized Domain 引入。要使这些模型能够成为一个可执行的系统，我们需要做如下的工作，如图 1 所示。可以分成五个部分。首先，我们需要将域和域之间的依赖性通过某种方法实现，表现在 PIM 模型中各业务域用桥接器进行互联。其次，PIM 到 J2ME 平台的变换需要处理 PIM 类到 J2ME 类之间的关联、泛化等关系，如主动类需要继承线程类以及 MIDlet 类的生成。第三，诸如对象的持久化，以及 UI 等需要用 J2ME 的类库接口或类来实现，需要在模型和 J2ME 类库间建立桥接器。第四，我们需要在 Excutable UML 模型和 Java 语言之间进行模型映射。最后，我们需要生成 J2ME 的部署文件。下面将讨论与前四个部分相关的一些生成技术：

4.1 域(Domain)间的依赖实现

桥接器(Bridge)在设计模式中用来分离规约和实现,也可以作为在运行时连结不同技术的适配器,如 CORBA-COM 桥接器。而在 MDA 中,我们用它来实现不同组件部分的依赖,即用来在两个不同的接口或模型间建立连接,使它们能够互相访问。桥接器可以看成 MDA 中各个模型之间的胶合剂,是 MDA 的实现核心,有广泛的应用。各个平台模型间的转换工具可以看成是一个 Bridge。模型执行虚拟机可以在执行 PIM 的过程中,通过桥接的方法在 PIM 和本地代码和资源间动态建立连接。

域之间不同依赖关系的桥接器处理方法[1]是:

a) 域需要其他域提供某种服务,这可以通过消息传递或者方法调用实现。例如 ATM 系统中 ATM 域向银行系统的账号管理域请求银行卡认证。则 ATM 只需向一个预定义的桥接器发送请求认证的消息,消息包含卡号和密码。由桥接器处理和账号域的认证工作。如果同步地操作,则可以直接调用桥接器中定义的方法,由桥接器处理两个域之间接口的协调。

b) 域之间的类存在一对一的映射,称之为类类映射。如一个商店,其销售系统中有一个销售商品类,库存系统有一个库存货物类,它们之间存在映射关系。销售出几件商品,意味着对应库存商品的减少。这种映射可能是广泛的,用显式调用相关方法或消息比较繁琐,不利于解耦。两个系统间可以建立隐式的桥接器,自动处理这种关联。

c) 一个域的某个类也可以和另一个域中多个对象建立映射关系。这种关系广泛存在于界面系统中。界面系统一个显示页面可以包括模型中多个对象的内容。可以为每个页面生成桥接器。如 VC++ 开发环境中,对话框就采用了这种技术,对话框的控件内容可以自动映射到实体模型中的属性。

桥接器可以用各种语言编写,但在我们的转换方法中用 Executable UML 的 Action language 编写,这样可以保证整个模型的一致性。

4.2 PIM 到 J2ME 平台 PSM 的转换

桥接器可以处理 PIM 域之间的依赖性以及 PIM 域和 J2ME 库之间的调用关系。PIM 到 J2ME 平台 PSM 的转换主要处理两个部分之间一些相关类的泛化关系。

MIDP 应用程序采用的是 MIDlet 应用程序模型,MIDlet 是 MIDP 的基本执行单位,所以从 PIM 应该生成一个 MIDlet 应用程序框架。这需要在 PIM 加入两个 tag,一个表示主动类,一个表示系统主动类。系统主动类表示主线程类,而一般的主动类表示线程类。转换的规则如下:

a) 将每一个主动类扩展为 J2ME 的 Thread 类,并加入 run 方法。

b) 将每个系统主动类扩展为 J2ME 的 MIDlet 类,并实现 CommandListener 接口,生成接口实现方法。生成 MIDlet 类中用于生命期控制的抽象方法。对应于 startApp, destroyApp, pauseApp 三个方法。

4.3 J2ME 平台 PSM 到 Java 代码的生成

通过上面两个步骤,我们得到了 J2ME 平台下的 PSM。这个 PSM 还是完全用 Executable UML 表述的,包括了 J2ME 扩展了的 PIM 模型和桥接器。模型可以分成三部分,一部分是静态类图,另一部分是表述动态模型的状态机,最后是表达语义的 Action Language。MOF 具有四个元层次。我们的 PSM 和所需要产生的 JAVA 代码处于 M1 层。一个典型的转换方法可以先构建 Executable UML 的元模型和 JAVA 语言的元模型,然后对应其元模型的转换规则,用工具将这个过程实例化,产生 UML 模型到 JAVA 模型的转换,最后得到 JAVA 语言下的 PSM,最后由这个 PSM 产生代码。在这里我们通过原型语言(ArcheType language)在生成代码。原型语言可以取得 PIM 的模型数据,并根据元模型和原模型间转换规则,生成相应的代码,这样可以略去产生 JAVA 语言下 PSM 这一个步骤。原型语言包括操作部分和文本处理部分。操作部分可以用任意语言实现,这里我们还是用 Action Language 来实现。

桥接器和状态机内过程的代码生成:

桥接器和状态机内过程是由 Action Language 实现的,因为 Action Language 是一种类似于 SQL 语言的数据操纵语言,它的抽象层次比较高,它的下面抽象了数据结构,所以较难直接翻译成目标语言。因为 Executable UML 本身是一种建模语言,Action Language 可以看成是一种平台,所以可以用 Executable UML 建模 Action Language,得到 Action Language 的元模型,再用这个元模型和原型语言可以生成 JAVA 的代码。具体的实现过程这里不再赘述,"Configurable Code Generation in MDA using iCCG"里有个小例子可以参考。

4.3.1 *状态机的转换* 图 2 是一个状态机的简化元模型图。这个状态机是以 Executable UML 表示的,是不完善的,这里创建它只是为了演示一下状态机到 JAVA 的映射规则。下面是映射的基本步骤,因为 JAVA 中只允许单一继承,所以这里状态机的实现没有采用生成一个状态机基类的方法,而是直接将状态机整合到拥有状态机的类中:

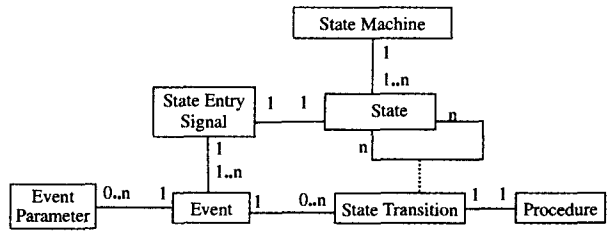

图 2

a) 在状态机所在的类建立一个状态转移表,里面有事件、当前状态、下一状态等内容。

b) 根据状态机内容,将每一个事件对应的状态等信息加入到状态转移表内。

c) 在状态机所在类中加入一个当前状态的属性。

d) 在状态机所在类中加入一个私有函数 GetNextState。

e) 为每个状态增加一个私有函数,封装其对应的过程,输入参数是一个事件类。

f) 为每一个输入事件建立一个公有函数,将事件的参数依次填入此函数的参数表中。

g) 在输入事件所在函数中增加如下内容:调用 GetNextState 函数,取得下一个状态,如果返回一个有效状态,就将当前状态设置为返回状态,并且将输入参数打包成事件类,传入当前状态所对应的过程。

4.3.2 *JAVA 类的生成* 静态结构的映射,在 MDA 中已经有了比较多的应用,OMG 也有了一些标准产生了,如 XMI 和 JMI。以下是 JAVA 类的生成规则:

a) 根据类的原模型,为每个元模型中类的实例,用类名生成一个 JAVA 文件。

(下转第 127 页)

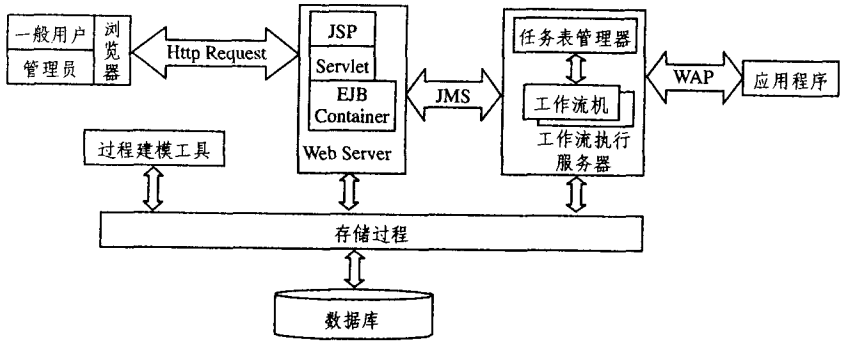

图2 系统结构

5)客户端:客户端是基于浏览器方式的瘦客户端,方便管理员管理整个工作流管理系统的运行过程,和一般用户管理和执行分配给自己的任务。

6)数据库接口:实现了底层的数据存储,包括过程定义,工作流控制数据,工作流相关数据,企业组织模型等工作流管理系统运行过程中必须的信息。

4 系统的一些实现技术

4.1 任务的自动分配和触发机制

可以根据模型定义自动地分配任务,当一个过程实例运行的时候,活动可以根据模型定义自动分配到指定接收者,并且,有关完成此活动所需要的数据也会传递给相应的接收者,从而提高业务过程执行效率。模型中使用角色机制,不指定具体人员,这样,人员变更不至于引起模型的变动。系统支持迟后绑定,即可以在活动运行的时刻才确定此活动由谁来完成。

流程从使能到运行的控制,采用触发机制,分为人工触发、自动触发、消息触发和时间触发。人工触发一般是用户从任务表中选取其中一项任务来完成,自动触发是一些通过程序自动执行的过程,一旦使能就被触发,消息触发是指系统外部的消息到达触发,如 Email,时间触发是由定时器来触发。

4.2 活动信息的统计

系统可以通过对活动信息统计,并将活动的运行状况和统计信息存储在数据库内。通过提供有关工作量的信息,可以在建模的时候预测所需要的时间,并且在活动结束时计算任务完成情况,与初始模型进行对比,生成相应的图表以判断工作效率,辅助决策经营。除系统提供的几个基本统计模型之外,用户也可以利用系统提供的工具,自行扩展新的模型来完成工作量信息统计和生成对比图表。

结论 根据软件过程管理的需求,以工作流技术为核心,J2EE 技术为支撑,结合 SPP 模型,文章给出了一个软件管理系统的体系结构和其中的一些技术实现。但是,为了更好地实施软件过程控制和度量,我们发现,还有一些问题需要进行深入的研究。

首先,软件过程模型的建立就要结合具体的实际情况,需要深入了解整个软件过程,并根据不同的需要修改模型来完成资源的动态配置和管理。另外,关于分布式工作流机之间的通讯和一致性问题也是相当重要的问题,需要拟定合适的策略来实现资源优化调度。

参 考 文 献

1 范玉顺主编. 工作流管理技术基础——实现企业业务过程重组、过程管理与过程自动化的核心技术. 北京:清华大学出版社,2001
2 Workflow Management Coalition. The workflow reference model. WFMC TC00-1003, 1994
3 林锐,王慧文,董军. CMMI3 级软件过程改进方法与规范. 电子工业出版社,2003.1
4 杨东,王英林,张申生,傅谦. 工作流过程建模方法及模型的形式化验证. 计算机科学,2003,30(9)
5 van der Aalst W N P. Structural characterization of sound workflow nets. Computing Science Reports 96/23, Eindhoven University of Technology, Eindhoven, 1996

(上接第 115 页)

b) 根据类名生成一个 JAVA 类,如果这个类有超类,就加入扩展超类的代码。

c) 查看类是否有实现的接口,如果有,将这些接口加入到类的声明中。

d) 为类中每一个属性,根据转换规则得到属性类型的实现类型,并生成属性声明。

e) 如果这个类有状态机,将上面状态机部分生成代码加入类的实体中。

f) 对对应类的接口中每一个方法,生成相应的代码。

g) 为类的每一个方法生成相应的方法。

结束语 MDA 是软件开发的一种趋势,它的各种应用方兴未艾。Executable UML 在消除建模语言和编程语言之间的界限方面迈出了坚实的一步,但要从平台无关模型到特定平台转换的工具化、自动化,还有很长一段路要走。本文主要提出了一种从 Executable UML 模型到 J2ME 平台间变换的基本思路,但要具体地应用,还有许多工作要做,例如当 PIM 到 PSM 映射时,存在多种可能映射途径时处理的细节等方面。

参 考 文 献

1 Mellor S J, Blcer M J. Executable UML A foundation for Model-Driven Architecture. ADDISON-WESLEY
2 Warmer J. MDA explained ADDISON-WESLEY
3 尤克滨. UML 应用建模实践过程
4 Riggs R, Van Peursem J, Patel M. Programming Wireless Devices with the J2ME
5 Carter K. ASL Reference Guide
6 Carter K. Configurable Code Generation in MDA using iCCG
7 Model Driven Architecture (MDA) Document number ormsc/2001-07-01 Architecture Board RMSC1 July 9, 2001