# MDA—新一代软件开发方法学的挑战与发展研究

陈 平 王 柏

(北京邮电大学通信软件工程中心 北京100876)

**摘 要** 本文首先概述了OMG的模型驱动对象(MDA)的体系结构,对MDA的发展前途的正反两方面进行了论述,同时给出了MDA的发展现状和最新动态、重点研究方向以及相关技术,并预测了其可能的发展模式及前景。

**关键词** 模型驱动的架构,元模型,软件体系结构

## The Challenge and Chance of MDA——A New Software Development Methodology

CHEN Ping WANG Bai

(Telecommunication Software Engineering Center of Beijing University of Posts and Telecommunication, Beijing 100876)

**Abstract** The paper presents an overview of MDA and its latest status. The two different views about MDA are discussed. The key technology and future direction of MDA are expatiated, several patterns of development are also described.

**Keywords** MDA, Meta-model, Software architecture

## 1 MDA综述

### 1.1 OMG/MDA体系结构简介

MDA指模型驱动的架构(Model Driven Architecture),它是一种对业务逻辑建立抽象模型,然后从抽象模型自动产生最终的完备的应用程序的方法论。MDA致力于提高软件开发行为的抽象级别,倡导将业务逻辑定义为精确的高层抽象模型,让开发人员从繁琐的重复的低级劳动中解脱出来,去更多地关注业务逻辑层面。它代表了OMG组织定义的互操作性规范的一个革命性进步。

MDA将软件系统的模型分为平台无关模型PIM(platform-independent model)和平台相关模型PSM(platform-specific model)[24],同时又能通过转换规则将它们统一起来,以这样的方式来解决需求变更所带来的问题。平台无关模型PIM是对工作流程的高层次抽象,其中不包括与实现技术相关的信息;平台相关模型PSM是跟特定平台相关的模型。

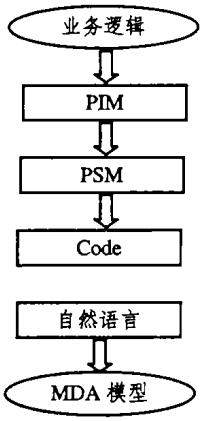


图1 遵循MDA的开发过程

遵循MDA的开发过程如图1所示。最上层的是业务逻辑,首先要使用平台无关的建模语言,来针对业务过程搭建平台无关的模型PIM,然后自动将PIM转换生成平台相关的模型PSM,之后根据特定平台和实现语言的映射规则,自动生成应用程序代码和测试框架[3]。

MDA的体系架构如图2所示,其核心是OMG的一系列标准[17,24]:

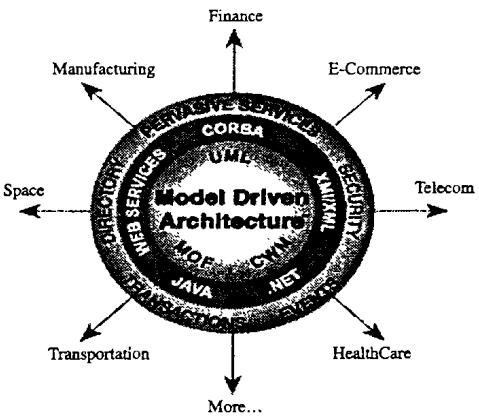


图2 OMG的MDA结构图[17]

·UML(Unified Modeling Language)—统一建模语言:UML被MDA用来对应用建模,它是MDA的基础。

·MOF(Meta Object Facility)—元对象设施:它是MDA的核心部分。和以往进行数据格式统一管理的思路不同,MDA允许使用不同格式(不同元数据描述)的数据同时存在,并为描述这些数据的语言提供了统一的定义语言(即MOF)。它是比UML更高层次的抽象,可以描述UML的扩展或者其它未来可能出现的类UML的建模语言,这体现了OMG的长远眼光。

·CWM(Common Warehouse Meta-model)—公共数据仓库元模型:提供了一种数据格式变换的手段,在各级的模型上都可以使用CWM来描述两种数据模型之间的映射规则,比如将数据实体从关系数据库变换为XML格式。在MOF的框架下,CWM使得通用的数据模型变换引擎成为可能。

陈 平 博士研究生,研究方向:通信软件.王 柏 博士生导师,研究方向:分布式计算.

·XMI(XML Metadata Interchange)—XML 元数据交换:它是基于 XML 的元数据交换。它通过标准化的 XML 文档格式和文档类型定义 DTD(Document Type Definitions),为各种模型定义了一种基于 XML 的数据交换格式。这使得模型可以在各种不同的工具中传递,能够实现不同模型之间的相互理解。

从图2中还可以具体地看到,从同心环中心部分的核心标准到外层环的 JAVA、.NET、Web Service 等平台模型再到外一层的 TRANSACTIONGS、SECURITY 等公共服务,最后到更外层的 Telecom、Space 等领域的应用代码,实际上就是从计算无关元模型[24]CIM(computation-independent model)通过模型转换到平台无关模型 PIM,再到平台相关模型 PSM,最后到各个领域的应用程序的过程,这就展现了模型驱动的开发过程。

有必要详细介绍一下 MOF,它是一种面向对象的元-元模型,为面向对象元模型定义了一种公共的抽象语言。MOF 为模型的转换和映射提供了基础支持,使得 MDA 可以处理由风格迥异的语言所描述的模型,只要有了这种语言的 MOF 元模型。MOF 用来定义面向对象元模型的基本元素、语法和结构[17]。元模型可以通过交换元数据来实现互操作。图3表示了四层元数据结构,数据的语义由模型来描述,模型的语义由元模型来描述,元模型的语义由 MOF 来描述,MOF 则有自描述的能力[17],所以可以使用它来建立一种建模语言的模型。

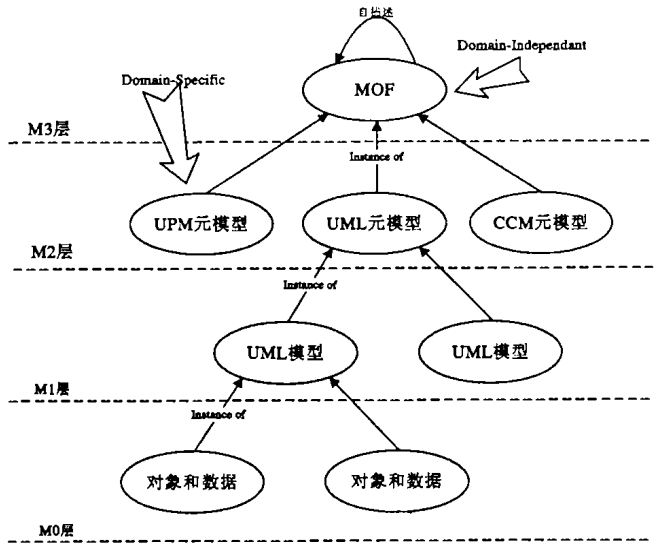


图3 MOF 的元层次[1]

综合而言,MDA 有如下的特点:

MDA 是一个开放的、中立于软件商的架构,能支持异构系统之间的互操作。它从系统模型层次上来解决互操作性的问题。互操作性能够实现的一个重要的原因就是设立了元模型的规则,元模型在规范中、建模技术中和元数据中都是主要的活跃因素。互操作性最终都是通过共享的元数据和理解元数据的策略来实现的。

MDA 是需求和技术之间的杠杆和纽带。PIM 是对需求的建模,其意义就是使用平台无关的语言进行业务流程描述,使得它和具体的平台以及实现技术分离,PSM 是应用具体技术后的模型,同时可以根据各种具体平台的映射关系来生成各种实现代码。它们各自的改变都可以是相互独立的,商业逻辑和实现技术是松耦合的,同时 MDA 又可以通过模型建立和模型转换来沟通它们之间的鸿沟。MDA 开发方式使我们的系统能够灵活地实现、集成、维护和测试。互操作性和可重用性是长期保持的,能够应对变化的形势。

### 1.2 当前 MDA 应用水平

MDA 的基本方法其实已经被使用很多年了,在"MDA"这个词出现之前它应用于嵌入式和实时系统。基于 Schlaer-Mellor 的系统用于为多种的电子设备生成嵌入式代码,从抽象模型产生数百万行 C/C++代码来完成复杂的电信交换[31]。目前 MDA 的应用目前已经进入众多 IT 组织中,在全球1000家 IT 组织中许多企业级的 MDA 工程正在开展[27]。

模型驱动的方法是下一代软件开发的方法,核心应该是工具的发展,而不仅仅是方法的发展。现有工具如 AndroMDA[27]、ArcStyler[27]和 OptimalJ[28]这些工具声称可以生成遵循 MDA 的设计方法。但是多用于数据库应用程序,这些应用的特征是 ACID 事务,处理对象是大量的数据,但是在数据上只进行相对简单的计算。有一些工具可以用在企业应用中,从业务逻辑建立模型,然后可以把绝大多数的业务逻辑转换为实体类的规则。另有些工具只产生结构代码,作为常规的操作都被包含在实体模型中,但是对于商务逻辑的事务操作,这些工具只产生空的方法体。

## 2 MDA 面临的挑战

上面介绍了 MDA 的体系和优点以及应用状况,很多人对 MDA 的发展前景持乐观态度,甚至还有人提出软件蓝领失业的想法。下面将介绍关于 MDA 的负面观点[27],使我们可以反省对 MDA 的认识和理解,因为 MDA 发展前景并不明朗,借此可以帮助我们找到正确的发展方向。

(1)总体上看,MDA 的发展还需要受到很多的磨砺。历史上当 CORBA 提出来的时候,人们以为它是解决软件互操作性的圆满解决方法,甚至其创始人 Mike 在一次演讲中说了 CORBA 会经历10 年的艰苦的发展过程,都导致了大会组织者的不满。不过事实证明 CORBA 确实不能担此重任。同样,MDA 是否能担当起万能解决方案的角色,仍是有待实践证实的。

(2)MDA 标准体系本身的问题。MDA 展现了一个全面的体系结构,但是在各个模块之间是脱节的[2,3],或者说没有实际上的连接手段,至少目前的 OMG 的体系还是如此。从必要性看,在某些技术细节方面,一些曾经被认为是 MDA 的独特功能的特性,也可以用已有的技术实现;从标准的执行性方面看,MDA 的 OMG 方案实际上并没有得到一些厂家的完全遵守,这将为未来埋下隐患。另外,目前厂家宣称的 MDA 工具实际上并不是完全实现了 OMG 的目标,更多的是部分的解决和商业炒作[16]。

(3)MDA 开发方法本身也有局限性。从适用范围看,MDA 的开发方法不能完全取代编码方法,比如它就不适用于系统软件和操作系统开发;从复杂性方面,应充分认识建模工具的局限性,以及业务建模和流程建模的难度。工具是否具有足够的职能或者说是否人类具有足够的建模能力,这都是很大的挑战。MDA 的发展可能会更加类似于一种框架生成工具,并不能完全取代编码。现有所谓的最好的 MDA 工具如 ArcStyler 等,目前实际上主要用于数据库应用程序的设计。

(4) MDA 作为比 UML 更高一层的开发方法学,在实践中,会不会变成仅仅作为一个工作流的高层建模工具而已,还难以确定。因为与 MDA 密切相关的 UML,经过了多年的发展,大部分人只是把它当作一个写文档时画图的工具,一旦进入详细设计阶段就被抛在了一边。

(5)方法和工具的推广是一个难题,因为现在很多的大企

业都有自己的一套开发模式和自己熟悉的开发工具。让他们放弃经验积累，去学习一个全新的开发模式，去使用一个全新的开发工具是困难的。

(6)关于编译器和虚拟机的争论问题。实现 MDA 的目标主要是采用编译方法还是虚拟机方法有一定的争论，目前的趋势是二者同时在发展。Java 将源代码编译成中间语言，中间语言就是一种动态的模型，它在运行期间被解释引擎(Java 虚拟机)解释。MS. Net 中所有的. Net 语言也是类似。虚拟机方式的坏处是显然的，增加了一个解释层，运行速度降低了；这样做的好处是程序是可以跨越平台运行的，并且让程序员专心于业务的实现而不必考虑平台的差别。

## 3 远期发展核心目标

前面两部分介绍了 MDA 的优势与不足。下面我们将先回顾软件开发方法的发展历程，然后展望未来 MDA 软件开发方法学发展的方向。

### 3.1 发展历史的回顾

(1)从软件开发方法自身的发展历程分析

从图4中可以看到，从机器语言发展到 MDA，软件系统的发展是在不断地提升灵活性和提升系统的可伸缩性，所以我们现在所进行的软件开发都可以看作是一种模型。可以看到，模型的不断提升，其结果是让开发人员更加接近于想要表达的业务逻辑。而运行期间的动态模型更是增加了其中的灵活性，更少的代码改变换来更多的对业务的关注。由此，我们设想一下，演进到一定的时期，只需要领域内的业务知识就可以设计软件(实质是设计业务流程模型)。

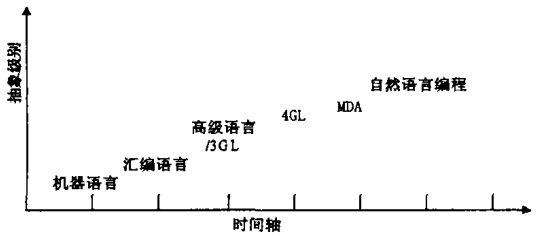


图4 软件开发语言的发展历程

(2)从参与程序开发人员组成角度分析

软件开发参与人员的变化状态可以用一个状态转换图(图5)表示。可以看到，从当前领域专家和计算机专家共同参与软件的设计，发展到没有计算机专家的参与[19]或者发展到没有领域专家的参与是一个阶段。之后可能一个最终的发展方向就是把计算机专家和领域专家也给抛弃了。这说明软件的开发是个自我否定的过程，软件开发不需计算机专家和领域专家的参与就是对软件开发作为一个专业工作的否定和超越。这个目标的实现，必须建立在人类知识库的完备建立和领域知识的模块化和形式化的坚实基础之上，也必须有高度发展的开发方法和工具的支持才可能实现。

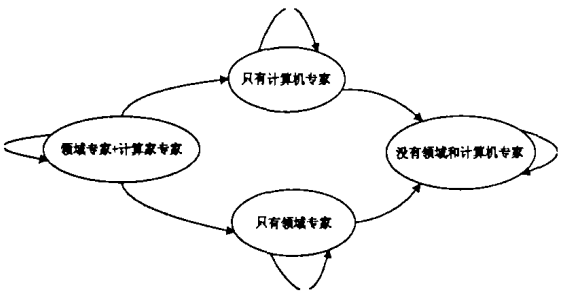


图5 软件开发的参与专家变化图

### 3.2 软件开发方法的战略远景展望

通过对历史的回顾，我们得到一些启示，从而可以向前进一步展望软件开发方法学的发展方向。

(1)人工智能和软件工程工具的结合 从以前的 DOS 系统的文本编程到 PASCAL 的集成界面，又到 VS. NET 的上下文敏感的智能编辑集成环境，体现了智能技术和界面技术以及编程知识库的及时咨询功能，大大提高了编程的效率。把软件开发知识库引入和集成到软件开发工具中去，将面向契约[15]、面向方面[20]、基于模式[21]、基于构件等方法囊括其中是一个趋势，这一点从 OMG(http://www.omg.org)的研究方向中可以得到体现。这样一个主要的好处是，能提供一个集成的环境，使得软件的设计和程序的生成可以在一个统一的环境中来完成，而不需人为地在一个个的分别的程序中来人工实现。这一步看似简单，但却是模型驱动得以实现的一个基础，有利于在软件开发流程中，高层模型转化时过程的无缝集成，它也是一定程度上的 MDA 的实现。比如设计了一个数据库的模型(这里体现为字段的特性的设计)不需要人为地再去理解设计意图，然后再进入数据库环境手动设计或输入 SQL 命令，来生成数据表及完成各字段的特性设置。

(2)进一步展望，把领域知识库引入和集成到软件开发方法和工具中 这是更进一步的发展，实现它首先要有一个精确的领域知识的模型集合和语义集，这涉及到知识的表达和推理的实现，目前还是人工智能的领域中的难点。一个思路是可以借助于向导和专家系统先把领域知识模块化。这样在工作流建模或需求建模过程中可以帮助进行特定领域的建模，最终的目的是不要业务专家也不要编程员。众所周知，知识的表达形式是一个有待解决的问题，如果知识无法形式化表示，那么抛弃专家的努力就没有实现的基础，所以从该角度看，要实现 MDA 各层模型间的自动转换，首先要实现模型和知识库模型的形式化表达和形式化操作，于是就提出了基于知识的方向[19]。

(3)自适应系统发展方向 更长远地来看，面向 MDA 的系统结构还要包括软件自动发现环境属性的能力，以及软件以各种方式对环境的适应力，包括自我行为的动态修改的能力。这是一个雄心勃勃的设想，它建立在对实现当前一些基本研究的基础上。设想将有如下的系统：该系统可以直接解释模型并且根据运行情况而修改自己的行为，它不需要显式地映射共享模型的外部视图到执行模型[19]。当修改模型时就直接修改了软件的行为，实现完整的运行时的扩展性。例如，发布一个新版本的模型到知识库里，使得环境中的所有的软件系统自动地调整它们的行为来反映这些变化。

高度相关的元数据是这种体系结构的关键之处，元数据(比如自适应的对象模型)在系统执行时会得到更新，当正在运行的系统的元数据变化时，系统行为和结构的改变就会马上体现出来。

这是一个高度动态的自组织系统，并且不须告诉具体如何做，它就能够直接按照领域知识来行动。这样的系统可以容易地适应那些不可预测的变化，并且不需程序员的参与就能够做出正确的反应。当系统确实需要修改时，只需要修改系统的模型。这时只需不熟悉软件专业知识的领域专家参加，或很多时候由系统自己完成。这一点与反射技术相关[29]，可借用反射技术的成果在模型驱动架构中应用。

(4)超越 MDA：在 MDA 的基础上的再发展 虽然 MDA 尚处于发展的早期阶段，但是还是可以基于软件方法学的发

展历程来更进一步展望MDA之后的发展状况。MDA之后，一个大方向就是对自然语言的开发方法的支持。可以设想，在MDA前端，先进行自然语言的描述，然后通过语音识别来识别语言，再通过人工智能的技术，来分析出自然语言中的语义，之后进行自然语言的建模，最后再映射到精确的领域模型上，往后的事就是MDA的工作了。这个模型比MDA模型还高一级，如图6所示。它的价值是不需要人们直接去设计业务的高层模型了，因为高层模型如果是精确的，那必定还得学习和精通复杂的建模工具，也不是任何人都能用的。这个展望几乎是目前可能预测的最高境界了。不过当前人工智能的发展水平还没有大的进展，可借用的是语音识别技术，然而，语音识别技术只不过是把语音文字化了，还应该在MDA之上有一个层次来把文字语言的非形式化的文本化为一个形式化的模型，形式化不是自动实现的，必须有一个人工参与和求精的过程，还有语义分析，歧义的消除，明确化语义，寓意的理解等等，这是更大的挑战。

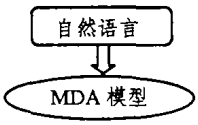


图6 人工智能对MDA的影响

## 4 MDA的当前研究领域

### 4.1 目前的重点研究方向

目前，总体上MDA相关的研究重点有：MDA的支撑技术、模型语言、模型转换、模型建立、模型运行、模型应用等。下面我们具体地描述一下：

(1)*标准化* 为了保证所有有关的组件能理解共享的元数据，基于MDA的系统需要在以下几个方面进一步标准化[19]：一是用形式化的语言(含语法和语义)来表示元数据。二是用一种交换格式来交换和发布数据。三是用一种供元数据访问和发现的编程模型，它必须包含通用的编程能力来处理具有位置性能的元数据。四是一个可选的某种形式的元数据服务，可以用于发布存储的元数据。五是扩展以上四条的机制。

(2)*模型语义的研究与模型语言的完善* 设计和建立精确高效的模型定义语言是当前的重点[14]，这需要对方法论的进一步深入研究[12]和对象约束语言OCL(Object Constrain Language)的进一步发展。目前UML2.0的RFP(Request For Paper)在模型语义方面做了不少的改进和提高[17]。但是目前在该方面的研究还在继续中。UML1.x的不足在UML2.0中有所改善，但是元模型元素之间仍旧还有太多的交叉依赖性，使得即使只需要一个其中的子集，都必须引入太多的元素。而且用标准的方法让工具之间交换图表的工作还没有完全解决。

对象约束语言OCL可以对模型的精确化定义和模型转换提供保证，虽然荷兰的Klasse Objecten公司已经发布了Octopus支持OCL2.0的Eclipse[32]开发环境的插件[30]。不过OCL本身和符合它的工具都还不是很完善。UML也没有提供OCL的元模型，当多个特定工具在MDA环境中协作时，目前不具有扩展性[3]。所以目前对UML语言及其对象约束语言的研究和完善是一个值得研究的重点。

(3)*建立业务领域的子建模语言* 针对各种不同的应用领域，需要建立大量的基于MOF的元模型、定义和建立领域相关的子模型语言[8]、定义专用的抽象语法。这些元模型要包含应用领域的特殊规则和语义，既要严格和没有歧义，又要容易为非计算机专业的领域专家所掌握。因为元模型之间必须要互相理解，所以各种不同的语言又必须是基于MOF元模型的。这方面的工作也有待研究。

(4)*平台间模型映射的研究* MDA是从系统模型层次上来解决互操作性问题的方法，它和具体的平台以及实现技术分离，同时根据不同的具体平台的映射关系来生成各种实现模型，最后映射到代码，例如Java、XML、SOAP，如图7所示。目前已有MOF-CORBA的映射、MOF-XML的映射XMI，MOF-Java的映射JMI，正在研究的内容包括MOF到Web-Service的WSDL和SOAP的映射。

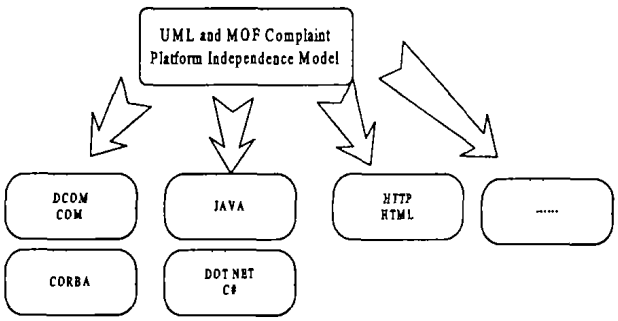


图7 平台独立映射到平台相关模型[1]

(5)*MDA工具中知识仓库的组织和研究* MDA工具的实现中一个重要的内容就是计算机软件开发知识的有效组织、开发和模式化。其中比如模式就是前人们成功的经验总结与成功的定式，包括算法的模式，设计的模式，建模的模式，把这些知识组织为模型转换中或代码转换中。所以基于模式[21]、面向方面[20]、面向契约[9]、面向对象等技术，和各种体系结构和平台架构的独立发展，以及以上技术和架构的面向MDA的整合，都是很有应用价值的，也是MDA开发工具实现的一条主要途径。最终需要建立一个系统，它把大量的软件开发知识和领域知识提高到更高的抽象层次[1]。系统会具有理解如何有效地抽取信息和操作信息的能力，并在模型开发过程中给与知识的支撑。

(6)*编译技术的进一步发展* 从模型到可执行代码，需要一次编译或者多层次的编译。或者结合虚拟机协同工作。目前编译技术发展得很成熟了，但是要对MDA的模型进行各级编译或称映射技术仍是一个新的挑战。

(7)*超级虚拟机和专用CPU的研究和开发* 开发虚拟机实质是运行平台的开发。可以借鉴Java虚拟机的经验，并加以外推，使得模型可以直接运行在虚拟机上[5,19]。还有专用CPU的开发也是一条思路，参考Java CPU[22,23](可直接运行Java的CPU)，是否可以开发出直接运行高层MDA模型的CPU也是一个挑战。

(8)*应用方面* 在业务领域中进行工作流建模，在MOF元模型的指导下，比如对通信运营商的CRM系统的建模或宏框架的架构[25]。企业应用集成中的各不同系统的集成中可以用高层模型的转换来集成。另外在CWM的指导下，进行各种数据模型的同层水平转换的元模型设计[26]，也是当前各大企业，尤其是电信运营企业的重点需要解决的问题。

### 4.2 实施MDA的技术路线

上一节从研究内容的角度给出了目前重点的研究方面，

下面接着介绍实施 MDA 的技术路线。

MDA 实施路线有一纵一横的两个主要发展方向：完全符合 MDA 的方法[7]和部分符合 MDA 的方法学。后者又有两个分方向，分领域的纵向[13]和领域无关的水平同层间的映射，某层间的映射又可以在 PIM(平台无关模型)和 PSM(平台相关模型)层间研究，会形成如图8示意图的三个分方向。

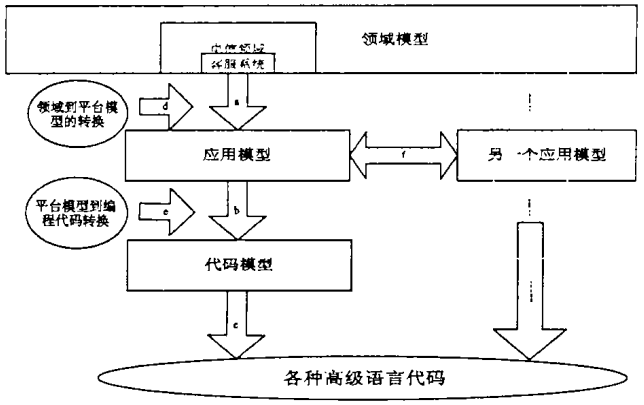


图8 实施 MDA 的技术路线示意图

路径1是选择一个范围狭窄但是有意义的垂直域，可以是整个垂直域中的某些子集，然后使用 MDA 方法去一层一层地实现它(a→b→c)，或者是面向某个领域的，比如面向电信领域的体系或者是面向电信领域的 ERP 的系统。做这件事的意义就是，生成的体系是一个可实际应用的系统，比如前述的几个工具就主要是面向数据库设计的。

路径2就是在整个体系结构中的水平层中完成某种层间的映射如图8中 d、e，即是说映射或叫转换，其中重点的核心就是规约，而且是层间的带普遍意义的抽象规约。比如说，定义语义完备的模型系统，并把它映射为一个平台无关的结构。各个领域的模型设计好了，它们将如何统一呢？一个解决方法是仿照 Eclipse 软件开发环境[32]，把各方开发的领域模型和自己的向导集成到某个系统中，各领域模型系统之间可以是兼容的也可以是独立的。

路径3就是同级模型之间的模型转换如图8中 f。比如从一种类型的数据库转换成另一种类型的数据库，这需要比数据库模型更高层的元模型来指导其转换。

**结论** 综上所述，MDA 提供了一个战略性的方法学，在它的理论里，一切都是模型。MDA 的发展目标就是要尽可能减少重复工作，尽可能地提升复用的层次和范围，甚至在模型一级复用，并且实现模型的直接运行或是代码的自动生成。这是一个远大的目标。但是有较多的因素制约着 MDA 的进一步发展，有自身体系的不完备的原因，又有外部对 MDA 的容纳难度，表现了 MDA 在曲折中发展的必然性。但是，MDA 将是未来软件体系研究的重点，其中有一些挑战性的工作和研究热点，这些方面包含 MDA 本身的问题研究和 MDA 的应用问题，二者的协调发展将使 MDA 在未来走向真正的全面的应用。

## 参 考 文 献


1 Model-Driven Architecture: Vision, Standards And Emerging Technologies. Position Paper Submitted to ECOOP 2001. Workshop on Metamodeling and Adaptive Object Models
2 Kleppe A, Warmer J, Bast W. MDA Explained: The Model Driven Architecture™: Practice and Promise
3 Frankel D S 著，鲍志云译. 应用 MDA. 电子工业出版社，2003
4 Alhir S S. Understanding the Model Driven Architecture (MDA). Salhi_r@ earthlink. net
5 Mellor S, Balcer M. Execable UML: A foundation for Model Driven Achitectures. Addison-Wesley, 2002
6 Wammer, Klepper A. The Object Constraint Language: Precise Modelong with UML. Addison Wesley, 1998
7 Gervais M P. Towards an MDA-oriented methodology. Lab. d'Informatique de Paris 6. In: Computer Software and Applications Conf. 2002. COMPSAC 2002. Proc. 26th Annual Intl.
8 Wang Hong, Zhang Dong, Zhou Jun. MDA-based development of e-learning system. In: Computer Software and Applications Conf. 2003, COMPSAC 2003, Proc. 27th Annual Intl. 2003. 684～689
9 Fuentes L, Pinto M, Vallecillo A. How MDA can help designing component and aspect-based applications. In: Enterprise Distributed Object Computing Conf. 2003. Proc. Seventh IEEE Intl. 2003. 124～135
10 Wegmann A, Preiss O. MDA in enterprise architecture? The living system theory to the rescue. In: Enterprise Distributed Object Computing Conf. 2003. Proc. Seventh IEEE Intl. 2003. 2～13
11 Silaghi R, Strohmeier A. Integrating CBSE, SoC, MDA, and AOP in a software development method. In: Enterprise Distributed Object Computing Conf. 2003. Proc. Seventh IEEE Intl. 2003. 136～146
12 Gervais M P. Towards an MDA-oriented methodology. In: Computer Software and Applications Conf. 2002. COMPSAC 2002. Proc. 26th Annual Intl. 2002. 265～270
13 Anido L, Santos J, Rodriguez J, Caeiro M. A MDA-based model for vertical application task forces an experience report. In: Enterprise Distributed Object Computing Conf. 2002. EDOC '02. Proc. SixthInt. 2002. 167～178
14 Bezivin J, Gerbe O. Towards a precise definition of the OMG/MDA framework. In: Automated Software Engineering, 2001. (ASE 2001). Proc. 16th Annual Intl. Conf. on, 2001. 273
15 Mitchell R, Mckim J 著，孟岩译. Design by contract 原则与实践. 人民邮电出版社，2003
16 http://martinfowler. com/bliki/ModelDrivenArchitecture. html
17 http://www. omg. org/mda
18 http://www. csdn. net/MDA 论坛
19 Poole J D. Model-Driven Architecture: Vision, Standards And Emerging Technologies. Position Paper Submitted to ECOOP 2001 Workshop on Metamodeling and Adaptive Object Models Hyperion Solutions Corporation, April 2001
20 高海洋，等. AOP 综述. 计算机科学，2002
21 GoF 著. 设计模式-面向对象的可重用设计. 机械工业出版社，1998
22 http://larc. ee. nthu. edu. tw/dtc/doc/0824. pdf
23 http://cn. sun. com/microelectronics/pro-article/feature3. html
24 MDA Guide Version 1. 0. http://www. omg. org/cgi-bin/doc?mda-guide
25 王君珂. CRM 宏框架的形式化描述. 中国联合通信有限公司，2003
26 王贞喆. 基于 CWM 规范的数据仓库建模及管理——MDA 方法在数据仓库领域的应用：[硕士论文]. 北京邮电大学通信软件工程中心，2003
27 www. mdachina. net/ M-Programmer1. pdf
28 OptimalJ White Paper: How model-driven development enhances productivity
29 赵建伟，刘东升，等. 基于 CORBA 的反射工作流模型，计算机工程与应用，2002. 2
30 http://www. klasse. nl/ocl/octopus-intro. html
31 http://www. embedded. com/2000/0008/0008feat1. htm
32 www. ccw. com. cn/htm/news1/tech/02_12_17_12. asp