模型驱动体系结构研究综述*)

刘亚军　康建初　吕卫锋

(北京航空航天大学计算机学院 软件开发环境国家重点实验室　北京 100083)

摘　要　模型驱动体系结构(MDA)是由OMG提出的应用模型技术进行软件开发的方法和标准体系,其核心技术是平台无关模型建模和平台特定模型转换。MDA代表了现代软件开发理论与方法发展的主流趋势,可以预见到它会成为继面向对象技术后软件工程史上的又一个里程碑。本文对MDA进行了系统性的阐述,包括MDA产生背景、总体构成、模型体系、软件方法、应用与评价以及最新研究与趋势等。

关键词　模型驱动体系结构,平台无关模型,平台特定模型,模型映射,模型转换

Overview of Model-Driven Architecture

LIU Ya-Jun　KANG Jian-Chu　LU Wei-Feng

(School of Computer Science & Engineering, Beihang University, National Lab of Software Development Environment, Beijing 100083)

Abstract　Model-Driven Architecture(MDA), which takes platform independent modeling and platform specific transforming as its core technologies, is a software development method and standards system using modeling newly issued by OMG. As MDA stands for a general orientation of the evolution of modern software development theory and methodology, we can predict that it will definitely become the next milestone in the SE history following OOT. This paper presents a systematic introduction of MDA including its birth background, constitution, software method, application & remarks, and latest researches as well.

Keywords　Model-driven architecture, Platform independent model, Platform specific model, Model mapping, Model transformation

1　引言

近一段时期,模型驱动体系结构(Model-Driven Architecture,以下简称MDA),这个由OMG组织全新提出的应用模型技术进行软件系统开发的方法论和标准体系,正成为软件工程界的研究焦点,同时正在引起各个相关产业领域的高度关注。基于MDA标准的软件开发技术和方法,目前已经广泛应用于电信、金融、电子商务等众多行业的信息系统的建设。

20世纪90年代,面向对象技术的发展为软件开发领域带来了革命性的飞跃,成为软件工程史上的里程碑。而MDA的提出,也必将对未来软件系统,尤其是将对大规模分布式软件系统的设计、开发、测试、部署、集成、评估、演化等方方面面技术发展产生深远的影响。本文首先介绍MDA的产生背景,进而重点系统地阐述MDA的总体构成、模型体系以及软件开发方法,随后分别介绍MDA的应用、评价及MDA相关研究趋势,最后进行总结。

2　MDA产生背景

上个世纪90年代是软件中间件高速发展的一个时期,包括CORBA、J2EE、COM/DCOM、XML/SOAP等在内的一系列平台技术诞生并得到广泛应用。在近十余年的市场竞争中,支持不同技术标准的中间件产品并存,没有而且也不可能产生一个"最终赢家"。这种中间件分化(Proliferation)现象所带来的中间件平台之间的互操作障碍,直接导致了企业必将为不同中间件应用系统的集成付出昂贵代价。同时,由于企业商业逻辑与某种平台实现技术的"绑定",提升了信息系统的平台移植难度,加大了企业业务发展受制于某种平台技术发展的风险。因此,在这种背景下,如何解决企业信息系统建设的互操作性、可移植性、可重用性等问题,成为软件开发领域的重要课题。

从1997年起,OMG在继续推动CORBA平台技术及标准的同时,将其工作范围进行了重大的拓展,陆续颁布了几个重要的技术无关(非限于CORBA)建模标准,包括统一建模语言UML、元对象设施MOF、XML元数据交换XMI和公共仓库元模型CWM等。这4个建模标准面向各种软件平台技术提供通用的模型设计规范,实现系统设计过程中企业商业逻辑与实现技术相分离,在系统设计阶段保证企业的商业应用在不同实现平台上的互操作性、可移植性、可重用性。然而,这种要求实际上不仅仅限于系统设计阶段,而是贯穿于软件系统整个生命周期:从商业建模,到系统设计,到组件构造,到组装、集成、部署、演化。因此,更需要一套完整的规范和方法体系,在软件系统生命周期的各个阶段,提供互操作性、可移植性、可重用性保证。

在这种背景下,OMG于2001年正式提出了第二个框架规范:模型驱动体系结构MDA。不同于OMG颁布的第一个框架规范OMA,MDA不是一个实现分布式系统的软件体系结构,而是一个模型技术进行软件开发的方法。MDA的核心思想是首先为企业商业应用建立独立于实现技术的平台无关模型,再通过映射方法将平台无关模型转换为与实现技术

*)基金项目:国家"863"高技术研究发展计划项目(2002AA113090)。刘亚军　博士研究生,研究方向为软件集成、软件体系结构、网络管理;康建初　教授,研究方向为人工智能、软件体系结构、中间件技术;吕卫锋　副教授,研究方向为网络管理、下一代网络。

特性相关的平台特定模型，进而生成可执行代码并在目标平台上部署和实现。

3 MDA 概观

MDA 提供了一个开放的、独立于供应商的应对商业和技术变化挑战的方法。MDA 以 OMG 建立的各种标准为基础，实现将商业或应用逻辑与支撑平台技术相分离。通过 MDA 及相关标准建立的平台独立应用可以被实现于包括 CORBA、J2EE、.NET、Web 服务和其他基于 Web 的平台等在内的一系列开放和私有平台之上……"[1]。这是 OMG 对 MDA 的概括性定义和描述，也准确地阐述了该组织推出 MDA 的初衷和期望。

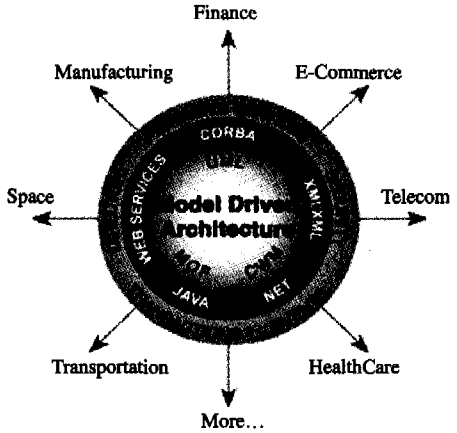

图 1 模型驱动体系结构

图 1 所示为 MDA 的总体结构[1]。这个结构不仅描绘出了 MDA 的标准体系的构成与相互关联，其从内环向外环的过渡性也体现了 MDA 软件开发方法的主导思想。以下分别介绍 MDA 的各个构成部分。

3.1 MDA 核心

MDA 的核心是 OMG 的建模标准，包括 UML、CWM 和 MOF。MDA 依据这些标准为企业应用建立独立于实现技术的平台无关模型。

UML(Unified Modeling Language)是一套标准的面向对象分析和设计的图形化模型语言，用于实现软件系统可视化(Visualizing)、规范定义(Specifying)、构造(Constructing)和文档化(Documenting)建模。MDA 的各种模型均采用 UML 进行描述。

CWM(Common Warehouse Metamodel)为数据仓库和业务分析领域最为常见的业务与技术相关元数据的表示定义了元模型[5]。CWM 实际上提供了一个基于模型的方法来实现异构软件系统之间的元数据交换。这样，依据 CWM 建立的数据模型，尽管它们存储于不同的软件系统中，但可以很便利地被整合和集成，进而确保数据挖掘等应用可以跨越企业数据库的边界。

MOF(Meta Object Facility)是 OMG 提出的一个对元模型进行描述的规范的公共抽象定义语言。MOF 是一种元-元模型，即元模型的元模型。MDA 中的 UML、CWM 元模型均以 MOF 为基础。MOF 标准的建立确保了不同元模型之间的交换。作为一个描述建模语言的标准语言，MOF 标准避免了将来由于建模语言不同而引起建模语言间相互理解与转换障碍。

3.2 平台支持

在 MDA 核心的外层，是 MDA 对各种实现技术平台(CORBA、J2EE、.NET、XML/SOAP 等)的支持。在这个层次上，平台无关模型被转换成为与各个平台技术特性相关的平台特定模型，并进而在平台上实现。在这个层次上，除了 CORBA 之外，MDA 的重要标准包括 UML Profile 和 XMI。

UML Profile 是对 UML 的扩展，通过对 UML 添加新的语言元素增强该语言的描述能力，以支持特定的计算环境(如企业分布对象计算 EDOC)和特定的平台技术(如 EJB、CORBA 等)。

XMI(XML Metadata Interchange)通过标准化 XML 文档格式和 DTD，为 UML 元模型和模型(元模型可以视为模型的特例)定义了一个基于 XML 的交换格式，随之也即定义了一个从 UML 到 XML 的映射[6]。更直接地说，XMI 定义了如何用 XML 对 UML 模型进行描述。UML 是一个图形化的模型描述语言，不便于被计算机所"理解"。而 XMI 提供了一个"文本"形式的模型描述方法，加以 XML 标记语言自身的技术无关性，为各个平台间模型共享和交换提供了一个便于计算机处理的途径。

3.3 普适服务

企业计算需要一系列的公共基础服务为其应用程序提供运行环境，如目录服务、事件处理服务、事务服务、持久性服务、安全服务等。如果这些服务仅被定义在特定平台之上，必然为了能够发挥该平台的最大优势而使这些服务局限于该平台的某些自有特性，从而限制了服务的通用性能以及整体系统的互操作性。为了避免这种情况的发生，MDA 通过高层次抽象，建立平台无关的公共通用服务模型，称之为普适服务(Pervasive Services)。

普适服务位于一个平台无关的高抽象层次，适用于所有的计算环境和应用。通过映射，普适服务可在 MDA 所支持的平台上得以具体实现，确保其不受某一特定平台特性的限制。OMG 将以其提出的 CORBA 服务为蓝本进行 MDA 普适服务的定义，包括目录服务、事务服务、安全服务、分布式事件与通知服务等。

3.4 领域应用

图 1 中，最外围的箭头代表了 MDA 对应用领域的支持。这种支持体现在两个方面：为领域应用提供了建模标准和公共模型，实现领域应用的可重用性、可移植性和互操作性；为各行业领域软件系统的开发和集成提供通用方法论。

OMG 下属的领域技术委员会 DTC 负责为包括电信、金融、制造业等在内的多个应用领域的服务与设施标准化。在 DTC 的领导下，有 10 个领域任务组正分别致力于采用 MDA 相关标准为各自的应用领域建立框架和功能规范，即建立标准化的领域应用平台无关模型、平台特定模型等。

4 MDA 模型体系

4.1 四层元模型架构

依据模型之间的描述与定义关系，MDA 中的模型组织为四层结构，图 2 是对 MDA 模型体系的层次划分和组织架构的描述。

从 M3 层到 M0 层：上一个层次是下一个层次的定义基础，即元模型；下一个层次是上一个层次的应用，即描述对象。M3 层为元-元模型层，仅包括 MOF，元-元模型为模型语言(元模型)定义提供公共标准；M2 层为元模型层，代表建于 MOF 之上的各种模型语言；M1 层为应用模型层，即采用模

型语言为企业应用建立的描述模型，MDA 中的 CIM、PIM、PSM 模型均位于这个层次；M0 层代表了企业应用真实的系统，或企业计算，或信息，也代表客观世界在人们头脑中的概念或认知模型，是模型描述的终极目标。

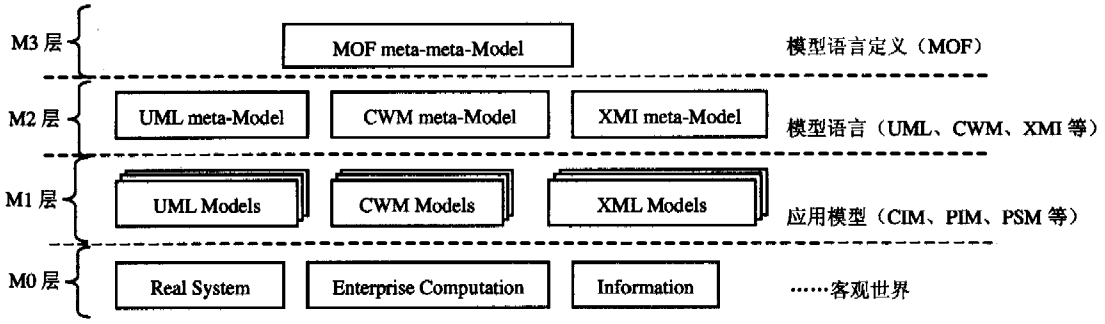

图 2　MDA 四层元模型体系结构

4.2　MDA 应用模型

4.2.1　CIM

计算无关模型 CIM(Computation Independent Model)是 MDA 基于计算无关视角(CIV)建立的系统模型，用于描述系统需求、功能、行为和运行环境，也称为业务模型。之所以被称为计算无关，主要是 CIM 侧重于表述系统的外部行为和运行环境，而不表现系统的内部结构和实现细节等相关内容。

MDA 中，CIM 为领域专家与系统设计专家之间关于领域需求的沟通和交流提供了桥梁，并直接支持 PIM、PSM 模型的构造和实现。

4.2.2　PIM

平台无关模型 PIM(Platform Independent Model)是 MDA 基于平台无关视角(PIV)建立的系统模型。PIM 是抽象出的业务逻辑。之所以被称为平台无关，主要是 PIM 不包含与实现平台和技术相关的特定信息。PIM 所表现出的平台无关性，使其能够在任何技术平台上得以实现。

4.2.3　PSM

平台特定模型 PSM(Platform Specific Model)是 MDA 基于平台特定视角(PSV)建立的系统模型。PSM 从相应 PIM 转换而来，它既包含了 PIM 中所定义的业务逻辑规范，也包含了与选定平台和技术相关的特定实现信息细节。

特别地，如果 PSM 提供了足够的用于构建一个系统并使之运行的信息细节，如程序代码、程序链接和装载规格、部署描述说明以及其他形式的配置规范等，则该 PSM 被称为一个“实现”(Implementation)。可以说“实现”是 PSM 的特例。

4.3　模型映射

模型映射(Mapping)是模型转换时所需定义的模型元素间映射关系，它提供转换规则及规格标准。MDA 中，模型转换的核心是 PIM 到 PSM 的转换。MDA 提供了两种模型映射方法：类型映射和实例映射。

4.3.1　类型映射

类型映射提供了从 PIM 采用的模型语言类型到 PSM 采用的模型语言类型的映射。PIM 采用某种平台无关的建模语言，利用该语言的模型元素对模型进行描述。同样，PSM 采用另一种平台特定的模型语言，利用该语言的模型元素进行描述。类型映射建立两种模型语言元素之间的映射关系，并据此形成转换规则，指示模型语言元素之间进行转换。语言元素的转换完成，相应地，PIM 被转换为 PSM。通过类型映射进行模型转换的过程可以理解为一个模型语言的“翻译”过程。

4.3.2　实例映射

实例映射的方法是通过对 PIM 模型元素加以标记，来标识该元素以某种特定方式转换为 PSM 模型元素。标记是元素转换的规则和规格标准的集合。当得到一个 PIM，需要对该模型进行标记，标记后的 PIM 是进一步生成 PSM 的直接基础。

实例映射中的标记(Marks)是平台特定的，其内容体现着 PSM 所选定实现平台的相应特定要求。特别当模型转换需要满足一些特殊应用要求时，需要采用实例映射的方法。

4.3.3　映射的组合

大多数情况下，模型映射是类型映射和实例映射的组合使用。

类型映射提供了两种模型语言之间一个“通用”的转换规则。一旦规则确定，从 PIM 转换的 PSM 结果也是确定的。

实例映射采用模型标记的方法进行模型转换，提供了一个通过加入附加信息指导和约束模型转换的途径。可以在标记中附加相应的类型约束条件，确保模型转换的“语义”正确性。此外，可以在标记中加入一些满足应用环境的特殊要求，甚至体现设计人员自身设计经验的约束条件，使模型转换更准确和更具效率。而这些，类型映射难以做到。

4.4　模型转换

模型转换是应用模型映射将描述同一系统的某种模型转换为另一种模型的过程。其中，从 PIM 到 PSM 的转换是 MDA 软件过程的核心。

对于模型转换来说，其输入是一个已存在的 PIM 和选定的某个映射，输出的结果是相应的 PSM 和转换记录。转换记录记载了模型转换的过程和内容，包括 PIM 元素和 PSM 元素之间的转换对应关系、映射的应用对象和范围等，用于检查模型转换的正确性和有效性。

4.4.1　模型转换方法

分别对应于类型映射和实例映射两种模型映射方法，MDA 提供两种基本的模型转换方法。

图 3(a)为采用类型映射的模型转换方法。当一个采用某种平台无关模型语言描述的 PIM 已经准备好，目标平台已经选定，类型映射已经定义，该 PIM 依据类型映射定义的转换规则被转换为采用另一种平台特定模型语言描述的 PSM。

图 3(b)为采用实例映射的模型转换方法。当一个 PIM 准备好，目标平台已经选定，映射已经定义，首先对该 PIM 进行标记，然后依据映射将已标记的 PIM 进一步转换为相应的 PSM。

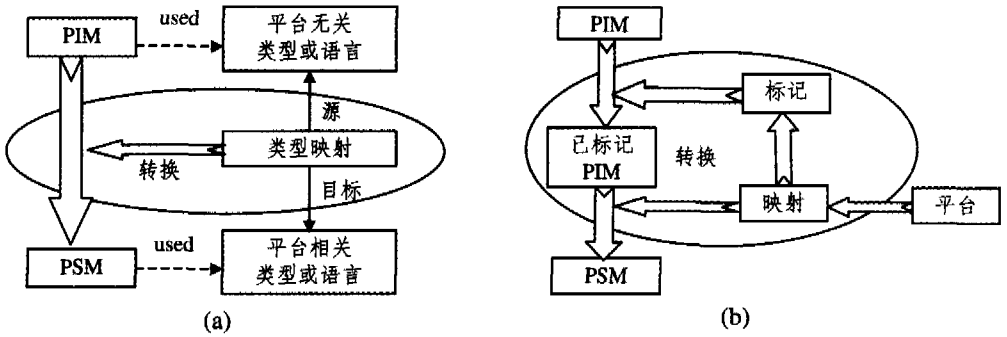

图 3　两种模型转换方法

4.4.2　模型转换方式

模型转换可以由设计人员手工完成。这需要设计人员参与和控制转换的整个过程，包括 PIM 的准备、目标平台的选定、模型映射的定义和设置、转换过程的记录、转换结果的保存等。

模型转换也可以由软件工具自动完成。但这需要前提条件：准备好的 PIM 和相应的映射包含了满足 PSM 自动生成所需要的全部信息。如果信息不够完备，例如转换规则漏定义，或标记信息不全，在转换过程中仍然需要设计人员的人工参与并进行转换决策。借助于软件工具，自动转换可以直接生成程序代码。

4.4.3　从 PIM 到“实现”的转换

多数情况下，从 PIM 到“实现”的模型转换并非一次完成，而需要经过多次。模型转换的次数取决于对每次转换结果 PSM 的可用性程度要求。例如：一个 PIM 首先被转换为特定于组件平台技术的 PSM1，以实现对各种组件平台技术进行支持（CORBA、EJB、COM/DCOM 等）；为了采用 CORBA 技术，PSM1 被进一步转换为特定于 CORBA 平台的 PSM2；为了采用 IONA 公司的 CORBA 产品进行实现，PSM2 又被转换为特定于 Orbix 平台的 PSM3。在本例中，进行了 3 次模型转换，每次转换都归因于相应的模型可用性程度要求。可以看出，PIM 和 PSM 是相对概念，比如上例中 PSM2 既是 PSM1 的 PSM，又是 PSM3 的 PIM。

5　MDA 软件开发方法

软件开发方法包括 3 个基本要素：符号（Notation）、过程（Process）与工具（Tool）。UML、CWM、XMI、MOF 等建模标准为 MDA 软件开发方法定义了符号，而众多软件商支持 MDA 的 CASE 产品为 MDA 软件开发方法提供工具，如 ArcStyler，Rational Rose，OptimalJ 等。

软件过程有很多种模型，如迭代型、瀑布型等。然而，无论采用哪一种，软件开发必然要经过需求收集、分析、设计、实现、测试、部署等多个阶段。

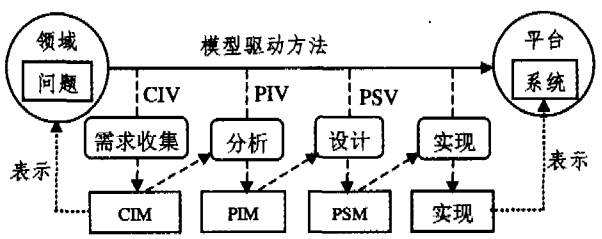

图 4　MDA 软件过程

图 4 是 MDA 软件过程模型的示意图[11]。图中，两个圆形框分别代表领域和平台，其内的方框分别代表了领域问题和目标平台系统，圆角方框代表软件开发活动，下部的方框代表了软件过程各个阶段的模型。

MDA 软件过程采用了模型驱动方法，即采用模型技术来“制导”软件开发每一个步骤或活动的进行。MDA 从 CIV、PIV、PSV 三个不同的视角，驱动系统需求收集、分析、设计、实现等活动的进行，采用建模和模型转换技术，逐步建立系统 CIM、PIM、PSM 模型，直至目标系统的实现。

MDA 的模型驱动软件开发方法极大程度地保证了系统的可移植性、可重用性和互操作性。与 CORBA、EJB、COM/DCOM 等软件技术不同，MDA 提供的不仅仅是可执行代码级软件重用，而是面向整个软件过程的可重用性。CIM、PIM、PSM 作为系统分析与设计阶段的独立性成果，可以作为以后相关系统开发中系统分析与设计工作的重用基础。尤其，PIM 以其平台无关特性，使得应用逻辑与实现代码分离，系统可以很便捷地移植到新的平台之上，从而新系统的建设不致于重新从零开始。PIM 采用通用的模型描述语言，可以在一个平台无关的高度上对系统整体的互操作性进行分析和设计，并对系统实现阶段的互操作接口和协议提供设计指导规范。

从企业角度来看，MDA 中的平台无关模型是商业逻辑，是领域知识，它超越并独立于系统的实现技术。这种超越性使得企业商业逻辑的演化不受制于系统平台；这种独立性确保每当一个新的平台技术出现，企业应用可以快速地移植到新的平台之上，已建立的领域知识模型得到最大程度重用。

6　MDA 应用与评价

目前，MDA 技术已经得到了企业界的广泛认同和应用。各行各业的众多企业或组织已经把 MDA 作为软件系统开发的指导框架，把 MDA 软件方法和标准应用于系统开发中。这方面的成功案例有：国际电信管理领域的权威组织 TMF 利用 MDA 方法和标准提出新一代运营支撑系统 NGOSS 的标准框架[12]；洛克希德-马丁公司采用 MDA 方法成功开发 F-16 组合任务计算机应用软件；瑞典最大支付服务提供商 Postgirot Bank AB 采用 MDA 技术对支付系统进行重建，重建后系统的运行费用较原来降低 80%，事务处理能力提高了 10 倍等[13]。

MDA 的提出，从方法论和标准方面解决了当前软件开发领域面临的困境和难题。为此，MDA 赢得了来自软件业和企业界的多方面高度评价。

OMG 主席 Richard Soley 博士认为：“应用 MDA 会带给

企业根本的适应能力，即在系统底层架构不断变化时而从稳定的模型衍生代码的能力。投资回报体现于贯穿整个软件生命周期的应用和领域模型的重用。"[14]

著名的面向对象技术大师，Rational Software 的首席科学家 Grady Booch 谈及 MDA 时说："……，针对那些已经相当了解的问题领域，我们可以甚至更经济地和技术化地固化系统的体系结构，而这样做的更好的途径就是模型驱动方法。"[16]

Borland 公司的首席技术设计师 Mike Rozlog 说："通过定义了一个应用设计中的高层次抽象标准，MDA 给我们带来了大幅度降低不断变化中的 IT 世界之复杂性的光明前景。"[16]

EDS 的资深设计师 Steven Witkop 这样评价 MDA："MDA 是目前最令人激动的提高复杂软件系统质量和加速代码开发的方法之一。通过计算机辅助验证和机器智能的结合，MDA 使得可以在代码检查和测试之前的建模阶段发现和排除设计错误。结果是：企业既节省了大量的开发时间和成本，又获得了所期望的业务敏捷性(Agility)。"[15]

7 MDA 相关研究趋势

目前 MDA 的相关研究集中在标准演进、MDA 方法与工具、MDA 与软件新技术的结合等方面。MDA 标准体系演进一直由 OMG 的技术委员会推动和主导。目前，该委员会正致力于 UML2.1、MOF 2.1 XMI 等新标准的制定以及为多个应用领域制定领域标准。

关于 MDA 相关研究，以下几个方面值得我们特别关注。

7.1 Agile MDA

Agile MDA 是 Agile 方法与 MDA 的结合。Agile 方法是 Agile Alliance 提出的一种迭代增量式的软件开发方法。Agile MDA 的中心思想是采用可执行 UML(executable UML)建立系统的可执行 UML 模型，并对这些模型编译、运行和测试，使得在设计阶段就能快速验证系统模型的功能正确性，排除各种设计缺陷，确保后期生成的代码的质量和效率。目前关于 Agile MDA 的研究集中于可执行 UML 和 Agile MDA 工具两个方面。前者主要是现有 UML 标准的扩展和完善，后者重点在 Agile MDA 工具的集成化，包括可执行模型的建模、转换、编译、运行、测试以及代码自动生成等。

7.2 领域特定建模语言(Domain-Specific Modeling Language)

为了使领域专家能够更容易地采用模型技术分析和解决问题，需要一种能够清晰完整描述领域问题的建模语言，即领域特定建模语言。UML 是一种通用的建模语言，可应用于任何领域。然而 UML 中的语言元素定位于从软件设计角度对事物的抽象描述，不容易为领域专家所理解和掌握。领域特定建模语言的符号和关系则紧密关联于领域问题和知识，更便利于领域专家建立和分析应用逻辑模型。

领域特定建模语言被业界认为是 MDA 进一步推向应用前沿的重要途径，众多的研究组织和软件商也已把对领域特定建模语言的支持列为研究和开发的重要方向。

7.3 MDA、Web 服务与 SOA

Web 服务作为新兴的基于 Internet 的软件技术，同时解决了数据层的信息表示以及业务层的互操作和松耦合协作的问题，其涉及的关键技术包括 XML、SOAP、WSDL、UDDI 等。面向服务体系结构 SOA(Service-Oriented Architecture)以其平台独立的接口、松耦合、业务级集成粒度以及可发现的服务等特点被称为下一代 Web 服务的系统基础架构。MDA、Web 服务与 SOA 三者的结合(MDA 的软件方法和标准、Web 服务的实现技术、SOA 的系统架构)将是未来企业 IT 系统建设的主流趋势，也是现代应用开发领域最重要的研究课题之一，重点研究内容包括 PIM 与 Web 服务 PSM 的映射与转换、UML Profile for SOA 等。

结束语 面向对象方法给我们提供了认知和表示客观世界的技术手段，而 MDA 则站在面向整个软件生命周期的高度上给我们带来了全新的软件开发方法论和标准体系。"未来的软件工程师不再需要编程，而只需要建模，剩下的工作交由计算机完成"，这是人们对 MDA 发展终极目标的期望。虽然目前离这个目标尚有一段较长的距离，但 MDA 代表了现代软件开发理论与方法发展的主流趋势，我们的软件开发正在逐步从"对象组合"时代进入到"模型转换"时代。

本文介绍了 MDA 的方方面面，由于篇幅原因，并未对某个方面进行详细阐述，主要是期望能呈现给读者一个 MDA 的完整概貌。

参 考 文 献

1 How System Will Be Built. http://www.omg.org/mda/mda.htm, OMG
2 Model Driven Architecture(MDA), Document Number: rmsc/2111-17-11, Architecture Board ORMSC, July 2001
3 Object Management Architecture Guide, Revision 3.1, June 1995, OMG
4 MDA Guide Version 1.1.1, Document Number: omg/2113-16-11, OMG
5 Poole J D, Hyperion Solutions Corporation. Model-Driven Architecture: Vision, Standards And Emerging Technologies, April 2001
6 XML Metadata Interchange. http://www.omg.org/mda/specifications.htm, OMG
7 Meta Object Facility(MOF), Version 1.4, April 2002, OMG
8 OMG Unified Modeling Language Specification, Version 1.5, March 2003, OMG
9 XML Metadata Interchange(XMI) Specification, Version 2.1, May 2003, OMG
10 Common Warehouse Metamodel(CWM) Specification, Version 1.0, Feb. 2001, OMG
11 Understanding the Model Driven Architecture(MDA) http://www.methodandtool.com/archive/understandingMDA.htm, Sinan Si Alhir
12 http://www.tmforum.org/, TeleManagement Forum
13 http://www.omg.org/mda/products_success.htm, OMG
14 Model Driven Architecture, by Richard Soley and the OMG Staff Strategy Group, Object Management Group, White Paper, Draft 3.2, Nov. 2000
15 Driving business agility with Model Driven Architecture, Accelerated development, by Steven Witkop, Information Specialist, EDS
16 http://www.omg.org/mda/mda_files/Member_and_Analyst_Quotes.pdf, OMG Members and Industry Analysts Support the MDA, OMG
17 http://www.omg.org/schedule, OMG
18 Agile MDA, Stephen J. Mellor, steve@projtech.com, Project Technology, Inc
19 http://www.agilemodeling.com/essays/agileMDA.htm, A Roadmap for Agile MDA, by Scott W. Ambler, Copyright 2004-2005
20 Domain-Specific Modeling and Model Driven Architecture, Steve Cook, MDA Journal, January 2004
21 Using OMG's Model Driven Architecture(MDA) to Integrate Web Services, by Jon Siegel, Vice President, Technology Transfer, OMG
22 Bezivin J, Hammoudi S, Lopes D, et al. Applying MDA Approach for Web Service Platform. In: Proceedings of the 8th IEEE Intl Enterprise Distributed Object Computing Conf(EDOC 2004), 2004. 1541~7719