模型驱动架构的应用

The application of MDA approach

李　丽　（宁波城市职业技术学院　宁波 315100）

摘　要： 模型驱动的软件体系结构是一种基于UML以及其他工业标准的框架，是一种由自动工具和服务所支持的组织和管理企业架构的方法。通过选择目标平台，能够在特定的运行时实现中执行模型驱动的架构，从而最终提高系统之间的互操作水平。本文描述了MDA的概念、构成及核心，阐述了建模的原理及模型驱动架构的某些应用。

关键词： 模型　建模　模型驱动的软件体系结构

1　引言

OMG 创建了一个概念性的框架，它将平台选择与独立的面向业务的决定分离开来以使在架构和演进这些系统时允许更大的灵活性，这个概念性框架和帮助实现它的标准就是 MDA。模型驱动的软件体系结构（Model Driven Architecture MDA）是一种应用系统设计

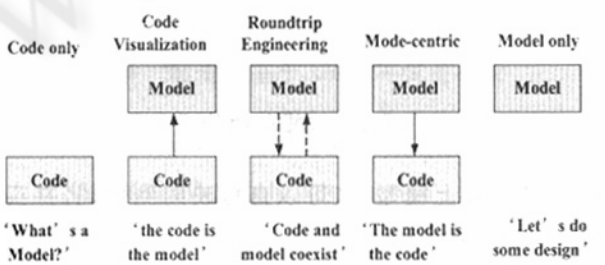

图 1　模型和代码间的关系

和实现的方法，是对象管理组织（OMG）提出的基于形式化模型的系统描述和互操作方法。MDA 从系统模型角度解决互操作问题，通过使用形式化模型，把系统描述与它的具体实现相分离，逻辑业务与实现技术被成功地解耦，二者相对独立变化，同时又具备到许多可能平台基础设施（如 Java、XML、SOAP 等）的形式映射。这样一来，应用程序就能够按照标准建模，也使得建模能够完全不受平台的限制，同时，模型的价值在包容已有技术的条件下被最大化。通过选择目标平台，能够在特定的运行时实现中执行模型驱动的架构，从而最终提高系统之间的互操作水平。

2　模型驱动架构的概念

2.1　建模及其对软件发展的作用

模型和模型驱动软件的发展是 MDA 方法的基础，也是它的核心。建模提供了对于一个实体系统的抽象方法，对于预测系统质量、当系统表面发生变化时定位特殊的属性等方面都非常有用，还可以让不同的开发者之间进行系统特性的交流[1]。

下图描述了模型是以不同的方式和源代码进行同步的，它展示了当前软件开发过程中进行建模的步骤。每一个类标识了一个软件开发者为一个特定的运行时间平台创建应用程序而使用的特定模型；同时，它也显示了模型和代码之间的关系。

架构设计模型中任意一个部分若被分离出来，都是非正式的，这种方法对于个人或是非常小的团队来说是够用的，但若要实现商业逻辑的细节，或是理解该系统的关键特性时，以及当系统的规模及复杂性发生变化时这种非正式的模型都会使得管理变得很困难。因此，可以使用某些合适的建模符号来提高代码的可读性，即代码模型，或执行模型。通过这种方法，图表可以用来直接表现出代码的作用，并且为视图和可能需要编辑的代码层提供了一种交互的方式[1]。

2.2　模型驱动架构的定义

MDA 是一种基于 UML 以及其他工业标准的框架，支持软件设计和模型的可视化、存储和交换技术；是一种组织和管理被自动化工具支持的企业体系架构和用

于定义模型和推动不同模型类型之间的转换的服务的方法。从狭义范围上来说，它是关于系统的不同抽象模型，在它们之间定义明确的模型进行相互转换；从广义的角度而言，它是从不同层次抽象出的模型，为通过不同技术最终实现的软件架构的基础而服务。

MDA 能够创建出机器可读和高度抽象的模型，这些模型以独立于实现的技术开发，以标准化的方式储存。MDA 以一种全新的方式将 IT 技术的一系列新的趋势性技术整合到一起。但 MDA 技术本身并不能完全结束编码，它的架构中包含了 PIM（平台独立模型）和 PSM（平台相关模型）两个重要的部分，对于实际的应用来说，从 PIM 生成 PSM 的工作是必不可少的。

2.3 模型驱动架构的战略意义

在传统的软件开发方法中，随着项目的进展，设计阶段产生的 UML 模型和代码之间的同步变得越来越困难 - -代码为了应付新增加的需求和新产生的想法而不断变化，模型却一直停留在原地不动，这使得模型在一段时间之后就失去了它的价值[2]。而 MDA 从根本上解决了这个问题—模型不再是一种辅助工具，而是开发过程的产品。MDA 使用编译器来自动生成和实现架构，如 J2EE、.Net、CORBA、COM 相关联的数据转换、持久存储、转换完整性、数据格式等。采取基于模型的方法，建立标准形式的系统模型之后，代码生成器可以从模型中自动生成出代码。

MDA 提升了软件开发的抽象层次，软件工业的最终产品将不再是代码，而是独立于计算平台的模型。产品的变化必然带来工程化方法的变化，可以说，MDA 改变了软件开发的方式。

3 模型驱动架构的构成及核心

3.1 模型驱动架构的构成

一个完整的 MDA 应用程序包含：

- CIM（Computation - independent Model）这个模型来自于业务领域，通过对业务领域建模产生；
- 一个权威的 PDM（Platform - Independent Model）这个模型通过 CIM 映射过来，经过修改 PIM 进行平台无关建模；
- 一个或者多个 PSM（Platform - Specific Model）这个模型通过 PIM 映射过来，经过修改 PSM 对平台相关的部分进行建模；
- 一个或者多个完整的实现 - 所有平台上的应用程序实现。

如图 2 所示，MDA 方法提出用 PEM 描述系统需求，然后通过规则将 PIM 转换成 PSM，从而可保证构造的软件系统能够适应新的硬件和软件平台。

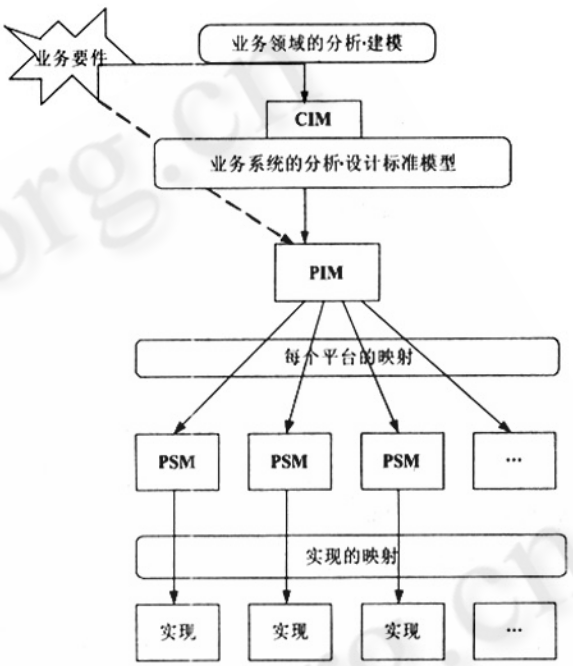

图 2　MDA 的构成

有了以上模型，就可以直接映射到代码，并实现模型与代码的同步。

对于 MDA 来说，当模型没有规定对于中间件技术的特定选择时，一个模型可以自由选择相关的通信中间件，此时它是 PIM；然而，当模型规定了必须使用特定的中间件，比如 CORBA，则模型被转换成 CORBA 型的 PSM。新的模型也可能仍然是和选择相关的 PIM，而且肯定和目标操作系统及硬件相关。如图 3 所示。

因此，MDA 工具可以支持分几个步骤进行模型的转换，从最初的分析模型到可执行代码[3]。这种转换可以应用于对于系统外形的抽象描述添加细节，使得描述更为具体，或者在描述方式之间进行转换。

3.2 模型驱动架构方法的核心

MDA 方法的核心是一系列重要的 OMG 标准：

- UML（Unified Modeling Language，统一建模语言）

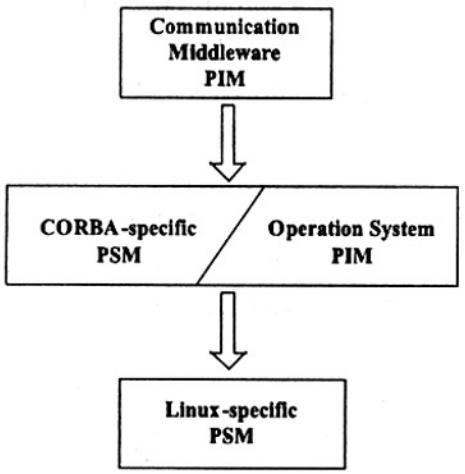

图 3　同一模型的不同形式

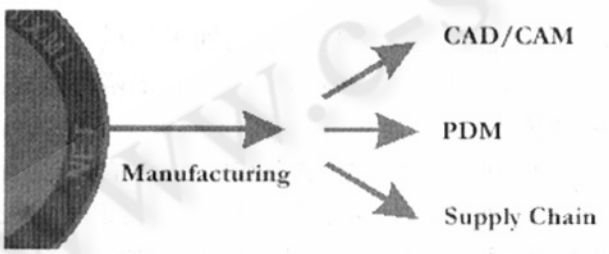

图 4　制造业基于 UML 的模型框架

- MOF(Meta－Object Facility,元对象设施)
- XMI(XML Metadata Interchange ,XML 元数据交换)
- CWM(Common Warehouse Metamodel,公共仓库元模型)

这些标准形成了创建、发布和管理模型的基础。企业解决方案的描述可以建立在使用这些建模标准并且转化成一个主体开放或私有的平台,包括 CORBA,J2EE,.NET,以及基于 Web 的平台。

(1) UML — MDA 实现的基础是基于 UML 2.0 的,UML 被 MDA 用来描述各种模型。UML 是一种通用的可视化建模语言,用于对软件进行描述、可视化处理、构造和建立软件系统的文档。UML 适用于各种软件开发方法、软件生命周期的各个阶段、各种应用领域以及各种开发工具。UML 能够描述系统的静态结构和动态行为。UML 不是一种程序设计语言,但可以用代码生成器将 UML 模型转换为多种程序设计语言代码,或使用反向生成器工具将程序源代码转换为 UML 模型。UML 与程序设计语言无关,而且,UML 符号集只是一种语言而不是一种方法学。

例如:对于制造业,DTF 可以为 CAD/CAM 互操作性、PDM(Product Data Management,产品数据管理)和供应链集成——如图 4 所示,生成规范的 MDA UML 模型、IDL 接口、Java 接口等。一旦这些模型完成并被采用,它们的实现可以在任何 MDA 支持的中间件平台上部分自动化。

(2) MOF — 是比 UML 更高层次的抽象,它的目的是为了描述 UML 的扩展或者其它未来可能出现的类 UML 的建模语言。MOF 规范是为了统一定义各种建模语言及模型间的转换关系而制定的,是各建模语言(元模型)的公共抽象(元元模型),用来统一建模语言的表达,继而可以统一各模型之间转换规则的表达,此规范借用 UML 的图示方法,基于面向对象思想的概念定义元建模元素。

在 MOF 思想中,用面向对象的实例化的概念将模型描述抽象成四层,即 M_0、M_1、M_2 和 M_3 层,M_i 层中的元素是 M_{i+1} 层中元素的实例,i=0,1,2。并且 MOF 可以做到自解释,因此不会再有 M_4、M_5...层[4]。

(3) XMI — 是基于 XML 的元数据交换。它通过标准化的 XML 文档格式和 DTDs(Document Type Definitions)为各种模型定义了一种基于 XML 的数据交换格式。这使得作为最终产品的模型可以在各种不同的工具中传递,它保证了 MDA 不会在打破了一种束缚之后再被加上一层新的束缚。

XMI 是一种工业标准,得到所有主要建模工具的支持,任何建立在 XMI 之上的模型都可以使用这些建模工具。XMI 可用于生成文档类型定义来描述图元素的语法,该文档类型定义可以在一个 XML 文件中使用。

(4) CWM — 提供了一种数据格式变换的手段,在任意级别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。在 MOF 的框架下,CWM 使得通用的数据模型变换引擎成为可能。

CWM 定义一个描述数据源、数据目的、转换、分析的元数据框架,提供使用信息的继承。上述三个标准:UML、MOF、XMI 是 CWM 的基础。尤其是 UML 在 CWM 中得到充分的应用,担任 3 个不同的角色:

· UML 用来做为与 MOF 对应的 meta – metamodel;

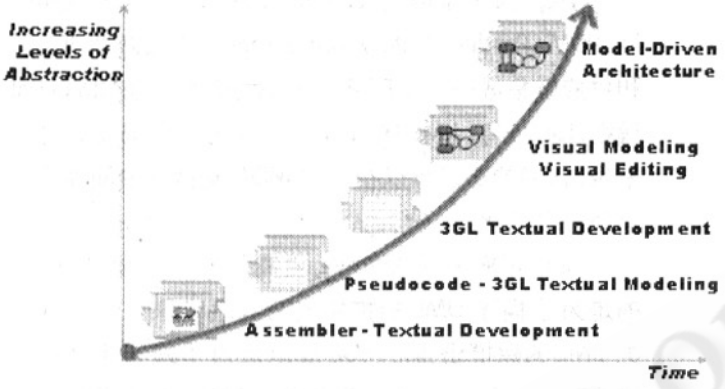

图 5　对于软件从业者渐增的抽象级别

· UML 用来创建元模型。

· UML 做为面向对象元模型(object – oriented metamodel)

CWM 元模型使用包(package)(对象模型包、基础包、资源包、分析包和管理包)和包等级结构来控制复杂性、提高理解性、支持重用。

4　模型驱动架构方法的应用

(1) Web services。使用 MDA 方法,可以为应用建模系统自动生成和底层架构、数据格式以及数据传输之间的连接,而开发者只需要关心程序的业务逻辑,当新的事物出现的时候,也可以生成相应的应用。MDA 对 Web services 的完全支持体现在:可以由一个 UML 模型自动生成相应的 WSDL 定义和 SOAP 文件头,从而避开那些繁琐的协议细节。另外,还有对应 CORBA 的映射。因此,如果已经有了一个 CORBA 架构,就可以用它来实现 Web service—bim、bam、boom。

(2) 模型驱动开发(MDD)。受 MDA 影响所产生的一个典型的开发技术——模型驱动开发(Model – Driven Development MDD),它支持快速创建含有新技术但经过验证和测试的模型。任何实现 MMD 的工具,都是在模型的实现和模型的映射上[5]。现在 IBM 已经实现了 MDD 的基于具有开放性的 Eclipse 平台的产品 Rational Software Architect(RSA),并在其中集成了过程管理、版本控制、代码复审等功能。RSA 主要是通过对 Eclipse Wizard、View、Editor 的贡献来实现的,具体到 MDA 上又与 RUP 的几个模型视图对应起来,每个模型都有相应的框架,在创立模型的时候就对模型之间进行了映射,所以不管在哪个模型上做了修改,其它模型也会相应地改变。采用 RSA,只要做了相应的支持工作,就可以更加专注于业务领域的模型。IBM Rational 工具一直强调建模作为一种提升系统抽象级别的方法的重要性,软件的从业者在这个抽象的级别上理解和构建软件解决方案。软件开发行业不断的意识到更高级别抽象的价值 — 从在机器语言级别上的代码描述到 MDA 的出现,如图 5 所示。

5　支持模型驱动架构方法的建模工具

基本上,当前可视化和开发工具以两种方式支持 MDA:

(1) 通过在特定的方案领域提供高度自动化。

(2) 通过允许组织容易的为自己的特定领域构建定制的模型驱动的方法来提供综合目的的能力。

除了上面已经提及的 IBM 的产品 RSA 外,还有一些其它支持 MDA 技术的建模工具:

· ArcStyler — 提供了对于 J2EE 和. NET 平台的 MDA 支持

· Optimal — 也是一种优秀的 MDA 建模平台

· AndroMDA 的开放源码 MDA 工具 — 使用 XML 元数据交换文件格式的输出,采用开放源码工具:Maven 或 Ant 管理安装和一般性应用。使用这个工具,开发人员要做的只是设计和建模应用程序,以及少量手工编码。

国内以及国外的几种产品对于"类 MDA"开发思想的贡献:

· Magic 系统工具/平台 — 涵盖了分析、设计、开发、部署这几个最为关键的软件构建阶段,提供了快速的跨平台的软件生成功能;

· Genux — 将整个软件开发过程通过自己的平台重新定义为:需求捕获、建模、自动化实现/部署几个阶段。

(下转第 112 页)

(上接第 100 页)

不论是否是 MDA 模式,或者"类 MDA 模式",所要达到的目的其实是一样的:让开发人员在更高层次上审视整个软件构建过程[2]。

6 结束语

模型驱动架构最大的好处就是业务模型的持久价值。在模型驱动架构开创的时代,代码将被认为是重复而机械的工件,各种各样的模型翻译工具可以在极短的时间内产生大量的代码。但是它付出的代价是增加了抽象层,因此,它也存在着不足之处,但是新的思想意味着新的机遇和新的挑战,模型驱动架构将帮助新一代的程序员摆脱编码的桎梏,模型驱动架构会遍及从客户端到服务器端的每一个角落。利用 MDA,OMG 继续寻求异构系统间所有层次上的集成和互操作性。模型驱动架构的近期目标是在共享元数据的基础上实现在 PIM - PSM 之间的形式化转换,使得软件系统之间能够进行接近无缝的互操作,它的长远目标是能广泛部署作为 MDA 的一种演进的自适应对象模型 AOM (Adaptive Object Manager)。总之,MDA 还在一个发展的过程中;完整的 MDA 定义还在不断的演进。

参考文献

1 Davud S. Frankel, 应用 MDA, 人民邮电出版社, 2003. 15 ~ 36.

2 Anneke G. Kleppe, Jos Warmer, Wim Bast MDA Explained: The Model Driven Architecture: Practice and Promise? Pearson Press, 2003. 26 ~ 45.

3 (美)Chris Raistrick, Paul Francis, John Wright, 赵建华译, MDA 与可执行 UML ? 机械工业出版社, 2006.

4 Anneke Kleppe, Jos Warmer, Wim Bast 著, 鲍志云译, 解析 MDA, 人民邮电出版社, 2004. 110 ~ 132.

5 David Frankel, Model Driven Architecture: Applying MDA to Enterprise Computing John Wiley & Sons OMG Press, 2003. 78 ~ 95.

6 S. Mellor et al., MDA Distilled. Forthcoming from Addison Wesley, 2004.