

# 模型驱动的软件开发模式研究

薛振伟, 吴志杰

(中国工程物理研究院 计算机应用研究所, 四川 绵阳 621900)

**摘要:**文中详细论述了模型驱动的软件开发模式。阐述了以代码为中心的传统软件开发模式的缺点,并通过对模型驱动架构(MDA)体系结构的讨论,分析出模型各个层次描述语言的要求。分别讨论了MDA软件开发生命周期、MDA软件开发参与者及活动的变更和MDA开发工具的更新,并对MDA软件开发模式进行了评价。在模型驱动的软件开发模式下,软件开发生命周期是由软件系统建模行为驱动的,其开发过程围绕模型的创建和变换开展,其最终目的就是要通过架构性的分离来实现软件开发的轻便性、互操作性和可重用性。

**关键词:**模型驱动架构;对象管理组织;计算无关模型;平台无关模型;平台相关模型

**中图分类号:**TP311.52

**文献标识码:**A

**文章编号:**1673-629X(2008)02-0024-03

## Study on Model Driven Pattern of Software Development

XUE Zhen-wei, WU Zhi-jie

(Institute of Computer Application, Chinese Academy of Engineering Physics, Mianyang 621900, China)

**Abstract:** The software development pattern driven by model has been discussed in detail in this paper. The shortcomings of traditional software developing pattern central on codes were presented firstly. The requirements of language for every model level were analyzed via the discussion about model driven architecture. Subsequently, the software developing lifecycle based on MDA, the participants and activities in the MDA process, and the update of MDA developing tools were discussed separately. Meanwhile, the software-developing pattern based on MDA was evaluated. Under the software development pattern driven by model, modeling behavior drove software development lifecycle. Model building and model translating surrounded the development process. The destination of MDA was to achieve the portability, interactivity and reusability of software developing through separating the architecture into many layers.

**Key words:** MDA; OMG; CIM; PIM; PSM

## 0 引言

模型驱动架构(Model Driven Architecture, MDA)是由国际对象管理组织(Object Management Group, OMG)于2001年7月正式发布的<sup>[1]</sup>。它是一种基于UML以及相关工业标准的框架,将软件系统建立在各种模型的基础上,通过模型的变换来驱动系统的开发。在传统的软件开发模式当中,大部分开发过程都是以代码为中心的,这种开发模式随着软件规模和复杂度的不断增加以及频繁变更的出现而逐渐暴露出越来越多的缺点和问题。相比而言,以模型为中心的基于MDA的软件开发模式在众多大型软件项目中越来越展现出卓越的问题解决能力和强大的生命力。

## 1 以代码为中心传统软件开发模式的缺点

目前许多开发过程都是传统的以代码为中心,即整个项目的开发以代码生产为主要任务,“编写软件仍是劳动力密集型的活儿”<sup>[2]</sup>。这种以代码为中心的开发模式主要存在以下三方面的缺点:

第一,混淆了解决问题的内在本质和外在手段,在分析问题本身的同时过多地去考虑运行环境与代码实现等因素,而放松了对问题根本解决方法的研究;

第二,大型应用系统往往涉及诸多领域的专业知识,而软件设计人员往往无法在短期内涉猎所有的专业技术领域,需要相关的领域专家参与需求分析工作。然而领域专家却只能用自己的专业术语对复杂的需求加以描述,在与软件开发人员的沟通过程中存在较多的障碍,影响软件人员对问题域本质的把握。

第三,软件技术的发展日新月异,开发语言、开发工具和开发平台一直在变化当中。如果还沿用以代码为中心的系统开发模式,既无法保障已经投入的项目投资,无法最大限度地复用原有系统的资源,又无法应

收稿日期:2007-05-27

基金项目:国家基金资助项目(20050656)

作者简介:薛振伟(1978-),男,河南濮阳人,硕士,工程师,研究方向为软件体系结构、应用软件开发、软件测试;吴志杰,研究员,研究方向为软件工程。

对变化多端的诸多环境因素。再加上系统的软件需求也会随着用户理解的不断深入而发生许多变化。即使采用原型法,也不能根本有效地解决随时都可能发生的变更问题。

## 2 MDA 体系结构

在 MDA 中,模型(Model)不再仅仅是描绘系统,辅助沟通的工具,而是软件开发的核心和主干。图 1 描绘了 MDA 的模型架构。

从图 1 可见,在 MDA 中,一个系统从不同的视角可以被不同的模型所描述。这些模型分为 3 个层次:

(1) 计算无关模型(Computational Independent Model, CIM):仅仅表述商务知识和商务过程,不涉及如何用软件来实现商务过程;

(2) 平台无关模型(Platform Independent Model, PIM):表述如何用软件系统来实现商务过程,去除了与基础功能无关的技术细节;

(3) 平台相关模型(Platform Specific Model, PSM):表述特定技术对软件系统功能的具体实现。模型之间通过模型映射机制相互映射,从而保证了模型的可追溯性<sup>[3]</sup>。

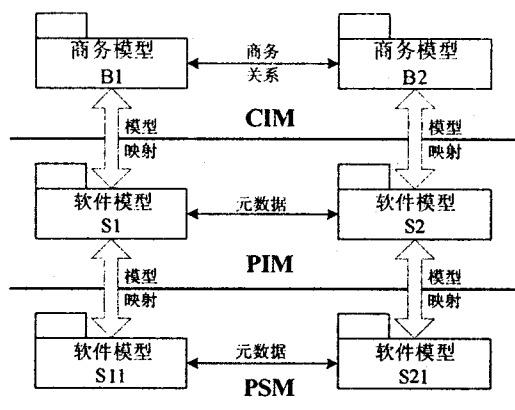


图 1 MDA 的模型架构

其中, CIM 就是通常意义上的“业务模型”,它描述系统的业务知识和业务过程,不涉及软件实现,因此用于构造 CIM 的语言就需要包含用来规定业务过程、参与者、相关部门、过程间依赖等方面的术语。而 PIM 主要用来表述如何用软件系统来实现业务过程,在模型中去除了与基础功能无关的技术细节,因此用于构造 PIM 的语言就需要具有较高的抽象,且能够与具体的技术细节脱离,同时也必须能够对系统的静态结构和动态行为进行精确的建模。最后 PSM 是用来表述某种特定技术对软件系统功能的具体实现,这是最贴近代码一级的模型,要求构造 PSM 的语言必须足够精确,且具备一定的扩展性,能够满足与各种实现技术紧

密挂钩的要求。

## 3 基于 MDA 的软件开发模式

### 3.1 MDA 软件开发生命周期

基于 MDA 的软件开发生命周期是由软件系统的建模行为驱动的,整个开发过程都是围绕模型的创建和变换开展的,如图 2 所示。

从整体来看,MDA 软件开发生命周期同传统软件开发生命周期并没有很大不同。关键之处在于,当完整实现 MDA 后,对生命周期的关注点从“代码”转移到了“模型”。目前,开发人员的关注焦点是代码,一般是使用面向对象编程语言书写。在使用 MDA 方法后,关注焦点就会逐渐转移到更高的层次上:先是 PSM 上,然后再转移到 PIM 上。开发人员会逐步忘记 PSM 需要被转换成代码,因为代码生成的过程将会被完全自动化,如图 2 所示。再过些年,部分开发人员甚至会忘记 PIM 需要被转换成 PSM<sup>[2]</sup>。

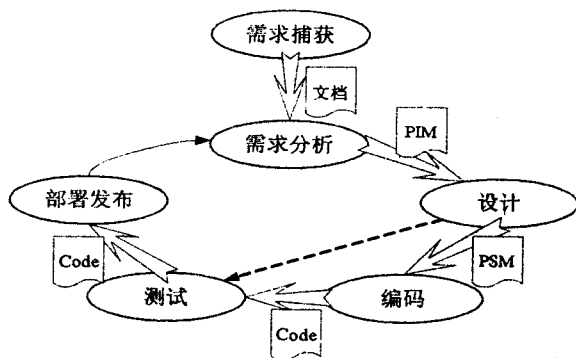


图 2 MDA 软件开发生命周期

### 3.2 MDA 软件开发参与者及活动的变更

与传统开发过程相比,MDA 开发过程有许多地方没有改变,仍需先找出需求并加以表示,需要测试系统、部署系统。改变较大的是分析、设计和编码这三项活动,以及参与 MDA 开发过程的三类参与者角色<sup>[2]</sup>。

首先,在分析阶段,需要由 PIM 分析员来开发一个 PIM,他们主要由一组领域专业人士组成,主要关注问题域的业务规范及需要实现的功能,并且由业务需要和业务模型来驱动。同时,在本阶段完成之后, PIM 应该得到充分的验证,以保证该模型的完整性和精确性。

其次,在设计阶段,需要由 PSM 创建者来将 PIM 变换成一个或多个 PSM。他们需要对不同的平台、不同的系统构架和所使用的变换工具的变换定义非常了解,从而决定使用什么样的变换参数,并在不同的目标平台和目标系统架构之间做出选择。同时,也是由 PSM 创建者来确定要开发的系统的架构(三层、两层

还是单机)和目标平台(J2EE 还是 .Net),也要承担起确保实现服务质量(QoS)的责任。PSM 创建者的另一个任务是对 PIM 和变换定义的改变做出响应。PIM 和变换定义都可能会独立改变。如果业务需求变了,那么受影响的只有 PIM。如果目标平台改变了,那么需要更新的只有变换定义。PSM 创建者必须及时响应这些改变,并将这些改变反映到生成的 PSM 中去。与此同时,已经部署的系统前一版本可能会包含很多历史数据,PSM 创建者还要借助工具,负责把这些数据迁移到新版本的系统中去。

最后,在整个 PSM 开发过程中,PSM 创建者都要用到变换定义,所以变换定义开发者的工作就显得尤为重要。这个角色也是在以往的开发过程中没有的。他们的主要工作就是结合具体的变换工具来开发变换定义,而 PSM 创建者通过购买或者修改变换定义来进行工作<sup>[2]</sup>。

图 3 展示了 MDA 过程中不同参与者以及他们使用的工具和创建的工件。

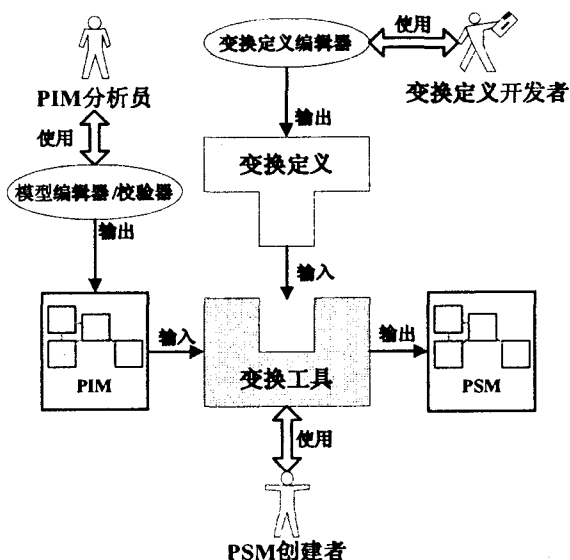


图 3 MDA 过程的参与者、工具和工件

### 3.3 MDA 开发工具的更新

应用 MDA 也将会对软件开发工具造成一定的影响。相对于传统的软件开发过程,主要会有 3 组人需要新的或更好的工具,即 MDA 过程中的变换定义开发者、PSM 创建者和 PIM 分析员<sup>[4]</sup>。变换定义开发者需要一种新型的 IDE 来创建、编辑和测试变换定义。由于 MDA 的目标是复用变换定义,且规范主要是由国际组织统一进行管理,所以参与变换定义开发的总人数不会太多,因此相应的变换定义开发工具也不会有太多的选择。

PSM 创建者主要使用变换工具进行工作,而这些工具需要具备以下功能:

- 1)能够在不同实现平台间进行选择;
- 2)能够灵活地在一个平台上转换不同实现策略、应用程序构架和编码模式;
- 3)具备插入多种建模语言和变换定义的开发性;
- 4)对标准的领域特定模型或蓝图提供支持;
- 5)能够同其他软件开发和维护工具(代码维护、版本控制、需求管理、工作流自动化、测试工具、性能调整工具等)自动集成<sup>[2]</sup>。同时,PSM 创建者需要对正在使用的变换定义进行少部分的调整,以便生成更高效的系统。而变换定义开发者需要调试和测试自己开发出来的变换定义,会频繁地调用一种或几种变换工具。因此,这些工具之间的标准性就非常重要,起码可以对模型和变换定义使用相同的标准交换格式。

PIM 分析员这组人是在传统软件开发过程中就已经存在的,已经有很多建模工具为他们提供服务。目前大多数工具都能以标准交换格式读写 UML 模型,但是对于 PIM 分析员来说还需要支持模型检查、模型各部分整合以及各种 UML 图表之间信息流的工具组。但 PIM 分析员最需要的还是更好的建模语言<sup>[2]</sup>。

### 3.4 MDA 软件开发模式评价

MDA 架构最核心的思想就是把软件设计的复杂性分散到不同的层面上,在每一个层面上只关心一类问题,使从简单性进化来的复杂性重新还原成多个层次上的简单性。其最终目的就是要通过架构性的分离来实现软件开发的轻便性、互操作性和可重用性<sup>[5]</sup>。

在 MDA 中将平台无关模型 PIM 和平台相关模型 PSM 显式地分开有下述三个主要的优点:

- 1)PIM 可以被多种实现技术复用,当技术平台发生迁变的时候 PIM 不必做改动;
- 2)由于和具体实现技术无关,PIM 可以更加精确地体现系统的本质特征,因此可以跨越具体的技术圈子进行交流和共享;
- 3)由于使用了简单的、通用的模型,可以更加容易地对 PIM 进行验证。

整个基于 MDA 的软件开发过程是一个以模型为载体、由模型映射所驱动的过程。这种开发模式采用一组基于 MOF 的元模型为系统建模。利用模型映射技术完成软件的逐步求精过程,并保证了这个过程的可追溯性。这种开发模式对于传统的软件开发模式在许多方面都有了根本性的改进。

相比而言,在传统的以代码为中心的软件开发模式中也具有“建模”的活动和相应的参与者,但是其主要的的问题在于很多软件团体对于建模只是“纸上谈

(下转第 30 页)

假定用户的请求是  $a$ , 则计算出比对算子  $\text{ExaC} = \{1, 0, 0, 0, 1, 1\}$ , 使用上述算法得到兼容子背景  $K' = (G', M', R')$ , 如图 4 所示, 其中由于对象  $e$  的对象相关度低于阈值未被加入到  $K'$ , 属性 3 和 6 也被约简掉了。

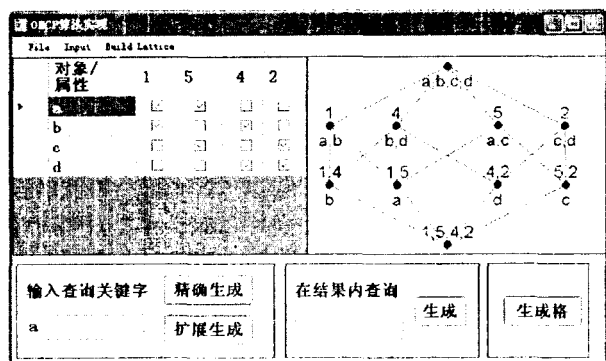


图 4  $K'$  及  $L(K')$

从图中可以很清晰地看到,  $L(K')$  同构于  $K$  的一个子序的概念格。如果用户利用精确分割后感觉生成的背景  $K'$  过小, 可选择利用扩展分割算法扩大形式背景的规模。由于本例的特殊性, 扩展后的形式背景就是原形式背景  $K$ 。在格的构造算法中, 首先生成  $L(K')$ , 然后再通过添加格节点生成  $L(K)$ , 由于从  $K'$  到  $K$  添加的对象的相容度很低, 所以新添加的节点对原有格结构的影响很小, 这样显然比直接生成  $L(K)$  的时间复杂度低。

## 5 总结与分析

利用 OMCP 算法分割形式背景, 改变了原有形式背景和概念格——对应的模式, 结合用户需求, 使格的

生成不仅仅依赖于形式背景的规模, 同时还依赖于背景中各对象间的相容度。该算法不但减小了格的规模, 而且使得生成的每个格节点都与用户需求相关, 从而有效地提高了概念格的有用性, 使得生成的格结构更易于应用于实际问题。例如将 OMCP 算法应用于专业搜索引擎, 可解决文档分类困难的问题。

为了验证 OMCP 算法的有效性, 在 Windows XP 下用 C# 语言实现了 OMCP 算法, 在 P4 1.7G 计算机上对随机产生的数据进行了测试。结果表明, 当背景中各对象的相容度有较大差别时, 算法表现出良好的效果; 当对象相容度的差别较小时, 算法效果不够理想。

实际应用中, 如果对象相容度差别较小, 可以将 OMCP 算法与形式背景的横向或纵向拆分算法<sup>[4]</sup>相结合, 从而进一步缩小背景的规模, 使其更有利于实现概念格的并行生成。

## 参考文献:

- [1] 李立峰, 王国俊. 一种求概念格属性约简的方法[J]. 计算机工程与应用, 2006, 42(20): 147-149.
- [2] 李云, 刘宗田, 吴强, 等. 概念格的分布处理研究[J]. 小型微型计算机系统, 2005, 26(3): 448-451.
- [3] 张磊, 沈夏炯, 贾培燕, 等. 基于同类概念的概念格横向合并算法[J]. 计算机应用, 2006, 26(8): 1900-1903.
- [4] Ganter B, Wille R. Formal Concept Analysis: Mathematical Foundations[M]. Berlin: Springer, 1999.
- [5] Ho T B. Discovering and Using Knowledge from Unsupervised Data[J]. Decision Support Systems, 1997, 21(1): 27-41.

(上接第 26 页)

兵”, 造成了模型和代码的不同步, 在代码不断被修改的同时, 模型不会被更新, 这样模型就失去了意义。弥补建模和开发之间的鸿沟的关键就在于将建模变为开发的一个必不可少的部分。基于 MDA 的软件开发模式彻底解决了上述问题, 并提出了一种创建系统的新途径。

## 4 结论

MDA 的出现, 为提高软件开发效率, 增强软件的可移植性、协同工作能力和可维护性, 以及文档编制的便利性提出了新的解决方案。MDA 被面向对象技术界预言为未来几年内最重要的方法学。

当然, 软件开发从来都是一个需要创造性思维的工作, 软件技术的进步也从来都是渐进的, 都是在已有

的技术上进行一个新层次的提升, 从来都没有什么“银弹”, MDA 也不例外。面对这种思维上的冲击, 既要睿智地去迎接新鲜事物, 又要冷静地在现有基础上进行提升, 这才是一个理智的开发团队本应该具有的素质。

## 参考文献:

- [1] OMG. Model Driven Architecture White Paper[EB/OL]. 2005. <http://www.omg.org/mda>.
- [2] Kleppe A, Warner J, Bast W. 解析 MDA[M]. 北京: 人民邮电出版社, 2004.
- [3] Frankel D S. 应用 MDA[M]. 北京: 人民邮电出版社, 2003.
- [4] Sawin 软件研发之窗[EB/OL]. 2006. <http://www.sawin.cn/doc/SoftMethod/MDA/blueski377.htm>.
- [5] UML 软件工程组织[EB/OL]. 2006. <http://www.uml.org.cn/UMLSearch/1217001.htm>.