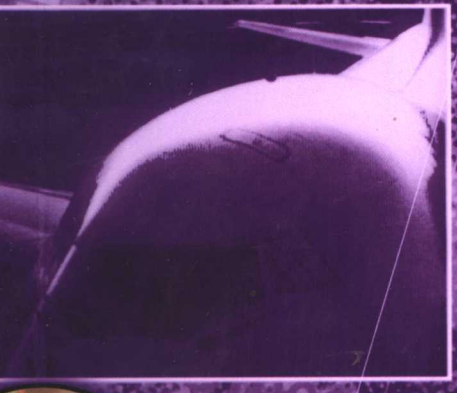
HZ BOOKS

MDA与可执行UML

Model Driven Architecture with Executable UML

(美) Chris Raistrick
Paul Francis
John Wright
Colin Carter
Ian Wilkie
著

赵建华 张天 等译



机械工业出版社
China Machine Press

MDA与可执行UML

Model Driven Architecture with Executable UML

Chris Raistrick
Paul Francis
（美） John Wright 著
Colin Carter
Ian Wilkie

赵建华 张天 等译
郑国梁 校

机械工业出版社
China Machine Press

本书以Rational统一软件开发过程（Rational Unified Process）为框架，描述了使用xUML的MDA开发方法在特定的软件开发过程中的应用。本书是作者多年的软件开发经验的总结，通过本书，读者可以学习到什么是模型驱动体系结构、如何使用可执行建模增强MDA、什么是xUML、如何建立xUML模型、如何通过对PIM的映射来完成代码生成过程，以及如何动态建模、如何使用xUML来表示这种映射等内容。相对于一些MDA理论研究的书籍，本书具有更好的实用性，对可执行建模的技术细节进行更为详尽的介绍。作者在本书中给出的很多方法、思想可以直接应用到软件开发实践（包括不使用MDA方法的开发过程）中去。作者以书中包含的多个实例揭示了MDA这个革命性软件开发方法的各种优点，并指明了利用这些优点的方法。

本书对于所有研究大中型软件项目开发方法未来发展趋势的人都是一个极好的参考，也适合作为高等院校计算机专业本科和研究生的参考书。

本书版权登记号：图字：01-2005-0679

图书在版编目（CIP）数据

MDA与可执行UML/（美）拉斯特瑞克（Raistrick, C.）等著；赵建华等译. -北京：机械工业出版社，2006.3

书名原文：Model Driven Architecture with Executable UML

ISBN 7-111-18371-1

Ⅰ.M…　Ⅱ.①拉…　②赵…　Ⅲ.①软件开发　②面向对象语言，UML-程序设计　Ⅳ.①TP311.52　②TP312

中国版本图书馆CIP数据核字（2006）第004682号

机械工业出版社（北京市西城区百万庄大街22号　邮政编码　100037）

责任编辑：范运年

北京牛山世兴印刷厂印刷·新华书店北京发行所发行

2006年4月第1版第1次印刷

787mm × 1020mm 1/16 · 21印张

定价：45.00元（附光盘）

凡购本书，如有倒页、脱页、缺页，由本社发行部调换

本社购书热线：（010）68326294

译 者 序

模型驱动的体系结构（Model Driven Architecture, MDA）是由OMG提出并且大力倡导的软件开发方法。在这种软件开发方法中，一系列不同抽象层次的系统模型成为软件开发过程的主要产品。这些模型被划分为两个较大的抽象层次：平台无关模型（Platform Independent Model, PIM）和平台相关模型（Platform Specific Model, PSM）。PIM描述了待开发系统的行为需求，但是不涉及如何在具体的平台上实现这个系统；PSM包含了所有在PIM中表示的功能，并添加了针对实现平台的设计因素。在MDA开发方法中，开发人员的主要工作是全面精确地描述系统的PIM，并根据特定的模型转换规则集合由PIM自动转换得到PSM。

MDA的好处之一是将系统的业务需求和实现技术分离开来。一方面，PIM的设计者可以专注于系统的业务逻辑，不需要考虑具体的实现技术。最终的系统通过模型转换规则自动生成。当实现平台发生变化时，只需要应用新的转换规则就可以从PIM重新生成新平台上的PSM，从而完成系统的移植；另一方面，有关系统实现的知识被表示为模型转换规则。这些规则和系统的业务逻辑无关，同样的转换规则可以被应用到不同的系统开发工作中。这些转换规则本身也成为了可复用的企业财富。MDA方法还可以带来多种其他好处，比如更好的可复用性、易理解性、更高的软件质量、较低的软件开发费用等。

UML是一个被广泛采用的，用以表示面向对象系统的各个方面的标准符号系统（www.omg.org/uml）。自UML被推出以来，它已经被工业界广泛接受为标准的建模语言。但是，UML的语义不是十分精确，难以满足MDA开发中模型自动转换的要求。可执行UML（xUML）对UML进行了扩充和改进。一方面，xUML用动作规约语言（ASL）对UML进行扩充，使开发人员可以精确地描述模型，并得到可执行的PIM；另一方面，xUML选取了UML表示法的一个子集，使得建模过程更加简洁明了。可执行建模可以使得模型更加精确，更加容易测试。

本书的作者以Rational统一软件开发过程（Rational Unified Process）为框架，描述了使用xUML的MDA开发方法在特定的软件开发过程中的应用。本书可以被看作是作者多年的软件开发经验的总结。相对于一些MDA理论研究的书籍，本书具有更好的实用性。作者在本书中给出的很多方法、思想可以被直接应用到软件开发实践（包括不使用MDA方法的开发过程）中去。作者以书中（包括随书光盘中）包含的多个实例揭示了MDA这个革命性软件开发方法的各种优点，并指明了利用这些优点的方法。

本书很适合那些试图了解软件开发方法新进展的读者阅读。虽然当前MDA开发方法还没有普及，但是它体现出的生命力，以及OMG的巨大影响力，必然会使得MDA被越来越多的人接受。虽然将来的MDA方法的具体细节可能和本书中的一些描述不同，但是MDA的基本思想是不会变的。预先了解这个方法，以及这个方法背后的思想，是很有好处的。

本书的引进出版首先由我的导师郑国梁教授提议。郑老师多年来一直从事软件开发方法的

研究，对这个方面有着深刻的理解。他选择这本书，说明这本书是值得大家仔细阅读的。在其后的翻译工作中，他还给我们提出了很多很好的指导和建议，对于提高本书的翻译质量有很好的帮助。在此我们表示感谢。参加本书翻译工作的还有：姜泉、杨璐、肖舒、雷斌、陈铭松、吴光、龚嘉宇。他们也为本书的翻译做出了很重要的贡献，在此一并表示感谢。

由于我们的知识水平和翻译水平的限制，翻译不当之处在所难免，请读者批评指正。

赵建华　张天

前　言

计算机软件业的历史就是将抽象层次分层的历史。从早年在曼彻斯特和费城紧随第一个存储程序计算机而出现的软件工具被发明开始，毫无疑问，同其他工程学科一样，控制系统复杂度的惟一方法就是建立多个抽象层，而每层都给出对其下一层次的简洁的描述。

第一种存储程序描述——我们称之为汇编语言程序——本质上是与底层实现结构一一对应的（比如，曼彻斯特的EDSAC的指令集结构）。人们很快就发现，降低编程复杂度对开发复杂的系统至关重要。由于这些系统的复杂性，人们必须对其复杂性进行抽象。甚至在EDSAC被制造并投入运行之前，为了简化编程，人们就开发了第一个后来被称为“宏汇编器”的工具和第一个连接/装载器。

从Wheeler和Wilkes的最早期工作开始，到高级编程语言（1959年的FORTRAN和Lisp，它们同时带来了软件移植性的概念）的发明，直到Algol语言中明确提出数据抽象的概念，软件行业找到了问题的“正交性”。从本质上讲，通过抽象的方法去掉软件开发中的某些复杂性，我们几十年来一直在尝试的就是将软件开发任务简化为描述“业务流程”，而将与描述流程不必然相关的代码编制的复杂度最小化。

20世纪90年代，“中间件”系统得到广泛应用，它们隐藏了IT开发任务中各种困难的复杂度（事务完整性、持久性存储、名字空间和目录服务等等），同时保证了易移植性和互操作性。过去曾经有人引用过我的话，因为我把中间件定义为“没人愿意付费的软件”，因为开发者必须假定不管平台如何选择，中间件都是无所不在的，而用户通常不愿为这样的基础设施层付费。不过，这些关键（且难以设计）的实现服务包已经非常普遍，并成为软件抽象的下一步。

当然，对于一个给定的系统设计，所有抽象层次上的描述都是等价的；只是（我们认定）高层描述更容易理解且更充分展示了程序员所期望的业务功能。那么我们如何从一个抽象层转到另一个层次呢？通过转换机制；这些系统设计（我们称之为系统模型）间的转换通常被称之为“编译”和“解释”。

本书讨论了另一个描述系统的抽象层，这是一个强大的且经过检验的抽象层。本质上，这只是另一个编译阶段，但是它却最终让我们弥补了系统设计和系统实现之间的差异，一劳永逸地给了我们梦寐以求的软件蓝图的圣杯：一种可以使我们抛开系统的实现、编程语言的选择、指令集结构、网络拓扑结构等诸如此类的问题，直接去定义系统体系结构的能力。自从OMG的MDA（Model-Driven Architecture）方法出现以来，它在实际的IT和嵌入式系统的开发中被一次又一次地证明是有效的。

那么，为什么MDA没有风行全球？其实从某种意义上说它已经做到了：OMG的统一建模语言（UML）和相关模型转换以及文本式的描述语言MOF和XMI已经在很短的时间内很快成为软件建模领域的通用混合语言。从1997年UML被采纳为OMG标准起到20世纪末，UML根本上代

替了所有现存的面向对象建模语言（比如OMT、Booch和OOSE）。我们需要的是下一个步骤：促使人们认同这个观点，即建模必须是软件开发流程的一部分。建模不仅和需求分析、系统和软件设计有关，还支持软件实现、单元测试、系统测试、长期系统维护和集成等过程。

这就要求观念上的改变，而改变本身就需要很好的信息来源以及为基于模型的方法建立系统的训练。在你手中的这本书正好是第一步，它对模型驱动体系结构的目标和期望作了清楚的描述。更重要的是，它还给出了理解MDA并将其用于工作实践的翔实指导。在阅读本书时，不要期望看到其中隐藏了一个革命性进展；虽然生产效率的提高（而且支持可重用性、易移植性和互操作性）将使MDA成为你的得力开发工具，你还是应该仅仅把MDA看作是另外一层抽象，把MDA工具看成是编译器。

这只是软件开发工具发展过程中的另一次很自然的进步，但你会很喜欢这个进步！

Richard Mark Soley 博士
对象管理组（Object Management Group, Inc）
中国　湖北　武汉

致　谢

作者衷心感谢以下几位的贡献、影响和支持：

- Glenn Webby，感谢他对全文完整而非常有价值的评审；
- Allan Kennedy，感谢他的评述和建设性指导；
- Jeff Terrell，感谢他有益的评述和建议，特别是有关代码生成部分的建议；
- Andy Land，感谢他多年来在开发iUML的工作中贡献的独到见解；
- Bobby Doherty，感谢他整理实例研究；
- Chas Weaver，感谢他对本书的方法性以及幽默性的贡献；
- Steve Lewis，感谢他对项目管理和组织的见解，那些见解几乎可以轻易地写成另外一本书；
- Terry Ruthruff和Lockheed Martin 的Bary Hogan，感谢他们和我们共享如何将MDA应用到实际项目中的见解和经验，同时感谢他们创作了一个广泛使用的MDA表示法；
- Alenia Marconi的Mark Jeffard，感谢他在企业级Ada代码生成领域的先驱性工作；
- Marconi通讯公司的Don Stewart，感谢他给出的许多关于开发流程的意见；
- CAP Gemini的Charlie Darkins；
- Mike Clarke，感谢他关于动作语言和代码生成的工作。

本书是多年经验和见解的综合。这些经验和见解，通过Dave Fletcher、Adrian King以及其他许多我们有幸与之一起工作多年的同事的趣闻轶事，年复一年地流传下来。

我们曾同许多知识渊博的人，诸如Leon Starr、Mike Lee、James Rumbaugh、Steve Mellor和 Sally Shlaer等，有过愉快的辩论、交谈和闲聊。Sally不幸于1998年11月12日逝世，她没有看到她和Steve于20世纪80年代末率先提出的思想以MDA和xUML的形式成为主流。Ovum Evaluates的Mike Budd说："Shlaer-Mellor的思想没有在她去世前得到完全的认同真是很让人难受，但可能这种认同是必然的。Shlaer-Mellor的严谨和快速的效率对程序员的聪明名声和现有文化提出了挑战。对他们的理论的更广泛的认同必将来到，但是和其他具有挑战性的思想一样，需要时间让更多的软件开发团体真心感谢Shlaer-Mellor的真正重要性。"本书便是对这个更广泛认同的宣言。

在随书光盘中包含了多个可执行UML案例研究，它们是由Paul Francis精心打造的。愤世嫉俗者可能会怀疑这本书是否只是另一个UML宣传用书，那么他们可以用光盘中的免费软件运行一下这些模型，以解除他们心中的怀疑。针对更多思想开明的读者，光盘提供了书中所述思想的多个具体的例子。

对致谢中被无意遗漏姓名的人，我们在此表示由衷的歉意。请告知我们，我们将在下一版中将您加入！

作者简介

Chris Raistrick　Kennedy Carter有限公司（www.kc.com）的项目工程管理服务主管。自1989年起，Chris主要致力于将面向对象方法应用于具有战略重要性的系统的开发中。他为电信、汽车、分布式控制、嵌入式系统、卫生保健等部门的客户提供咨询服务。Chris有长达5年的将UML成功应用于许多项目的实践经验。他在可执行建模、组件集成和代码生成技术领域发表过多篇具有开创性的论文，并曾在所有关于UML、OMG和嵌入式系统的主流会议上作过技术报告。

Paul Francis　Kennedy Carter有限公司（www.kc.com）的首席顾问。从事航空电子业的软硬件系统开发15年之后，他于1993年加入了Kennedy Carter公司。从那时起，他将可执行建模技术应用到水下武器、卫生保健、电信、航空、保险等多种领域的应用系统开发中。另外，他经常参与可执行建模技术的培训教学，曾在多个杂志或会议上发表过论文。他还负责Kennedy Carter公司正在开发的可配置代码生成器（一个针对可执行UML模型开发的基于元模型的转换器）项目。

John Wright　Aurora咨询有限公司（www.auroraconsulting.co.uk）的一位主管。从伦敦大学获得博士学位之后（研究实时并行算法），他在国防研究署（Defence Research Agency，DRA）工作了4年，并率先将MDA和xUML的先驱——当时新兴的面向对象分析引进该组织。自1995年离开DRA，他致力于帮助汽车、航空、政府、电子商务等领域的客户制定和改进业务流程，并从事xUML的指导和教学工作。John对软件体系结构和代码生成技术有着特别的兴趣。目前，他正积极着眼于如何使用UML、MDA开发基于J2EE技术的企业级体系结构。

Colin Carter　Aurora咨询有限公司（www.auroraconsulting.co.uk）的创办者，也是一位UML顾问、指导和方法论学者，他有20年应用软件工程方法的实践经验。他从1983年开始便使用面向对象方法进行工作，并将面向对象开发方法介绍给许多组织。自1998年起，Colin在多个项目中使用并深入利用了UML，其中包括从实时嵌入式系统到基于Web的应用的很多项目。他在可执行建模技术发展的早期便参与了该技术的开发，并具有将它应用于系统开发和业务建模的丰富经验。

Ian Wilkie　Kennedy Carter有限公司（www.kc.com）的技术主管，他有25年以上在多种目标环境中开发软件的经验。这些开发工作涉及了包括从高能物理数据分析的计算密集型系统到实时嵌入式控制系统等诸多领域。自加入Kennedy Carter后，Ian为许多项目提供过咨询，并全权负责开发用以支持MDA和xUML的iUML工具集。Ian还积极参与了“UML的精确动作语义”提案的撰写工作。该提案已经在近期被OMG采纳，成为xUML方法的重要基石之一。

目　录

第1章 引 论

1.1 为什么需要读这本书

这并不仅仅是另一本关于UML（Unified Modeling Language，统一建模语言）的书。它不是关于画代码图的，而是关于如何将你的智力转化为价值的书。它描述的是将任意主题的专业知识形式化表示的技术，该技术使得你的同事或者你的代码生成器都可以利用这些知识。它代表了一种人们期待已久的针对软件工程看法的重组。这种重组强调了那种通常只有在一个成熟的工程学科中才能找到的过程与标记符号的严格性。它将编程语言和中间件产品归入到事物模式中的各自合适的位置。编程语言和中间件产品通常被看作是一些用以支撑以模型方式存在的高级规约的技术手段。这些技术手段通常是以短暂的方式存在的。模型才是真正重要的东西，它们被建立后生存的时间要长于任何时下流行的中间件产品和编程语言的生存时间。

本书将向你展示如何建立具有长久生命力的规约。这些规约是用全球标准UML的一个简单子集来表示的。它们是按照对象管理组（Object Management Group，OMG）的模型驱动体系结构（Model Driven Architecture，MDA）的原理而组织并建立的。我们将会向你展示怎样才能够将这些模型转换到你所希望的底层实现技术上。本书并不是建立在理论和学术研究基础上的，但是书中所阐述的原理在概念上的一致性会吸引任何计算机科学家的注意力。本书中所论述的方法是建立在作者多年使用这些思想的实践经验的基础上的。目前对OMG的UML表示法和MDA过程的支持意味着UML和MDA将会取得全球瞩目的成绩。当MDA和UML的一个精确表示形式结合起来时，它所具备的真正潜力，虽然现在还没有，终将被全球的软件团体充分认可。

本书试图向读者提供一个作者认为即将到来的软件开发方法革命的实践性见解。那些在战略上或者关键性项目上已经使用了这种方法的组织，甚至从它们的第一个应用了此方法的项目上就意识到了此方法的重要价值，尽管采用这种全新的工程过程本身需要一次性支付学习和工具方面的开销。

1.2 从本书将会学到什么

本书中，你将会学到以下内容：

- 模型驱动体系结构（Model Driven Architecture，MDA）；
- 使用可执行建模增强MDA；
- 可执行建模的好处；
- 如何使UML可执行；
- 什么是xUML（Executable UML，可执行UML）；
- 用例的角色和应用；

- 如何划分系统;
- 如何使用xUML详述平台独立模型（Platform-independent Model，PIM）;
- 如何开发xUML的类图;
- 使用状态图的动态行为描述;
- 详述类操作;
- 附加于UML的OMG动作语义的好处;
- 动作规约语言（Action Specification Language，ASL）的实践应用;
- 一组xUML建模模式;
- 如何集成多个模型以描述整个系统;
- 如何描述转换规则以便可以从xUML模型生成任意目标语言的代码;
- 如何完全自动化地构建使用xUML详述的系统。

OMG的MDA有很多好处。可执行建模同样也有很多令人激动不已的优点。将两者结合起来，即在MDA框架下使用xUML模型，创立了一个新的软件开发过程。该过程会带给你如下有价值的优点:

- 关注点分离——可以将问题划分到不同的主题域，并在整个开发生命周期内保持这种分离;
- 表达知识资产——以标准化、可复用、可共享、易维护的方式表达主题域的有价值的专业知识;
- 可执行和可测试的模型——尽早地验证用户需求已被正确理解和尽早地演示所需的系统行为可以降低整个生命周期内的开销，降低风险，提高信心;
- 清晰且无歧义的模型——有助于在开发团队内部以及他们和用户、客户之间的交流;
- 尽早地集成——避免大爆炸式的集成引起的问题并支持迭代开发;
- 基于组件——组件的可用接口和所需接口的严格定义;
- 复用——在应用层、通用服务层以及平台和技术层上的知识资产的复用;
- 加速开发的生命周期——只要第一次分析的时候去理解需求，就可以在所有层次上进行复用;
- 可管理的轻量级开发过程——有严格的完成标准、最小的文档开销和清晰的工作分解结构;
- 可伸缩的、已被证明有效的工业级过程——已经被成功应用于各种工业部门的大大小小的项目中;
- 适应变化——功能需求的变化与技术平台的变化是相分离的，因此提供了无需重新建模就可以迁移到“下一代”平台的能力;
- 从模型的100%代码生成——这样就可以仅维护模型，而代码则成为衍生产品。模型也就永远都不会被废弃并且保证能够根据需求不断更新;
- 无冗余——最小的维护开销和最小的文档开销，支持“一个情况只在一个地方说明”的原则，也就是说一个事物只被描述一次;
- 完全的可配置代码生成——这样就能获取并表达有价值的平台和技术专业知识，并且可以轻松地适应变化;
- 生成高质量的软件——减少所引入的缺陷，具备在系统范围内进行系统化调整的能力;
- 减少生命周期开销——通过早期发现错误、保持关注点的分离，和降低变化所带来的影响

来减少开销；

- 可追溯性——通过可执行的PIM，需求到生成的代码之间具有可追溯性；
- 在各种应用领域的可用性——从实时嵌入式系统到大型分布式系统；
- 各种技术上的可用性——从分布式企业技术，嵌入式系统到“单片机系统（systems on a chip）”；
- 商用定制市场——当工具都采用OMG的互操作性标准时，就有可能提供了一个机会来通过集成“最好的工具”为一个组织或者工程项目建立一个经过优化的设计/工具链。

1.3 我们为什么写关于MDA和UML的书

你可能想知道这本书的作者是谁以及他们有什么资格来写关于UML、可执行建模和MDA的书。

我们是书中所提到的技术和过程的资深从业者，具有从20世纪80年代开始就从事面向对象软件开发的经验。我们有关定义软件开发过程的经验甚至比这还早，要追溯到第一个软件工程师费力地从原始海洋来到陆地的时候。在20世纪80年代，我们中的一些人已经从事于结构化和面向对象风格的软件开发（DeMarco（1978），Yourdon（1978），Page-Jones（1988），McMenemin和Palmer（1984），Ward和Mellor（1985），Hatley和Pirbhai（1988）见证了结构化分析技术的发展进程），并且也和Booch（1986; 1991），Buhr（1984），Meyer（1988）以及Nielsen和Shumate（1988）等一起参与了面向对象设计方法的创立过程。我们曾经在多个很早就采用这些方法的公司中工作过，我们看到了很多试图推动软件开发过程并取得了不同程度的成功的项目。在20世纪80年代末期，我们中的一些人在Kennedy Carter公司工作，提供有关使用结构化分析、结构化设计方法的咨询和培训，也为那些逐渐转移到Ada和C++，OOD（Object Oriented Design，面向对象设计）上的公司提供服务。我们开始应用OOD原理来改进结构化分析模型——从本质上讲，我们是在用面向对象的方法来进行结构化分析。

3

在1989年同Steve Mellor的一次见面，让我们关注到他和Sally Shlaer形式化地描述了一个*面向对象分析*（Object Oriented Analysis，OOA）方法（Shlaer和Mellor 1988; 1992）。我们不但被这个方法所吸引，而且也认识到我们的经验告诉我们结构化分析的使用者可以如何转向OOA。我们认识到从功能的观点转移到对象的观点需要思维上的根本变化。我们见识了一些不过是“披着对象外衣的功能方法”，并对这类分析方法保持警惕。通过提供良好的培训课程以及后继的咨询和指导，我们已经成功地让很多组织“考虑对象”。

Shlaer-Mellor的OOA模型的一个特点是它有良定义的语义。OOA模型中的处理模型是由简单的规范处理过程组成的。原则上，这些模型是可执行的，尽管还没有工具来执行它们。在1993年，IBM的一个工作组定义了一种*处理规约语言*（Process Specification Language，PSL）并建立了一个基于Smalltalk的工具集，使Shlaer-Mellor模型成为可执行的。我们使用了这种方法和原型工具，并且认识到了有适当工具支持的可执行建模的潜力。

在1994年，Kennedy Carter公司的OOA建模工具集定义和支持了ASL的第一个版本。该OOA工具集被称为I-OOA（Intelligent Object Oriented Analysis，*智能面向对象分析*）。它们不但可以像现在的*计算机辅助软件工程*（Computer-aided Software Engineering，CASE）工具那样

获取、编辑和查看模型，而且可以执行和调试模型。此外，可执行模型具备支持代码生成所必需的信息。1994年我们交付了一个客户定制项目，其中90%以上的代码是从可执行模型中自动生成的。

在我们开发自己的可执行建模过程与工具的同时，我们也密切关注着面向对象方法的发展。Rumbaugh等（1991）的OMT（Object Modelling Technique，对象建模技术），Martin和Odell的面向对象分析与设计（1992），Jacobson的用例（1992），Gamma等的设计模式（1994）以及Fowler对UML的介绍（1997）都非常有影响力。

我们曾经工作于国防、航空、汽车、电信、政府、健康、保险以及过程控制等工业部门，因此我们看到了面向对象、可执行建模和MDA的广泛应用。我们曾和客户一起在包括小型嵌入式实时系统到具有几百万行生成代码的大型分布式系统在内的项目中应用过可执行建模和代码生成。我们是从多年的第一手经验中懂得了可执行建模和代码生成是有效的。我们同样知道了它在什么地方能够最有效地工作，而在什么地方需要小心。本书中我们致力于使用实践经验来支持技术细节。

在UML成为标准之前，我们就认识到了它所具有的好处和影响。因此，我们和Steve Mellor一起着手调整Shlaer-Mellor的过程和形式化方法，以便在其中使用UML（Wilkie & Mellor，1999）。我们和Steve Mellor、Jim Rumbaugh以及其他人一起，认识到要完成上面的工作，还需要向UML模型增加一些附加内容和约束以使得它们可以执行。这使得我们加入到OMG的动作语义协会中来，该协会从1998～2001年致力于使动作语义成为UML标准（OMG，2002）的一部分。

为可执行模型开发代码生成器引出了很多有意思的问题。每个目标语言潜在地需要一个代码生成器。同时，为了处理不同的平台之间的差异，还需要额外的生成器的变体。过去，如果想要进行代码生成，就要对每一种语言/平台的组合定制一个生成器。这样一来，平台迁移就成为开销很大的事情，因为每一次平台变动都要对代码生成器进行大返工。为了提供一个框架以使得这些代码生成器能够被轻松地产生和维护，我们形式化地建立了一个基于xUML本身的生成xUML模型代码的生成器的方法。这就意味着相同的形式化表示方法被用来详述代码生成器和代码生成器所应用的模型。这使得有关代码生成方面的问题能够使用我们处理其他领域问题时的同样的方法来解决。因此代码生成器本身也是完全文档化的（因为它是用xUML表示的），并且也像其他可执行模型一样是可测试的。最后，它还有个优点就是，用xUML表示的结构良好的代码生成器要远比那些定制策略中常用的包含了YACC、LEX和其他脚本的复杂集合要容易裁剪、调整和演化。

这个革命性的方法需要充分理解xUML的形式表示法以及一些强有力的抽象建模技巧。目前可用的商用产品iCCG（Configurable Code Generation）提供了这种方法。

从1992年起，所有的作者都在面向对象分析与设计方面，提供培训、咨询和指导，而最近更多的是在UML方面。通常，这些工作是在使用可执行建模和代码生成的项目中进行的。有了这些在做广泛的实际项目支持工作时得到的深刻理解，我们详述和开发了一个为xUML提供大力支持的CASE工具。详述这样一个产品所做出的努力导致了对xUML方法的深刻理解。这种理解自然而然地对我们的咨询工作有所帮助。

在2000年11月，OMG发布了一篇关于MDA的文章（Soley，2000）。我们立刻意识到了我们

的代码生成方法与MDA合作的重要性和好处，并随后主动参与了OMG的建立MDA标准（www.omg.org/mda）的工作。

1.4 什么是模型驱动体系结构

首先，按照www.omg.org/mda上的字面概述：

MDA开发首先是针对分布式应用或系统的功能和行为的。这些应用或系统并不受实现它们的技术的特性的干扰。MDA将实现细节与业务功能分解开。这样就无需在每次新技术（例如，XML/SOAP）到来的时候，对应用程序或系统的功能和行为重新建模。其他的体系结构方法通常和特定技术相关联。使用MDA后，系统的功能和行为需要也仅需要一次建模。从平台无关模型（Platform Independent Model，PIM）到MDA所支持的平台上的平台相关模型（Platform Specific Model，PSM）的映射将会由工具来实现，这样就使得支持新技术或者不同技术的任务变得容易起来。

在任何软件组织中，最有价值的财富就是积累起来的专业知识。这就要求这些专业知识能够被形式化表示，并且在组织范围之内可被人获取和使用。这样看来以下的想法就是合情合理的，即我们应该以一种可以被系统化地、潜在自动化地映射为具体实现的方式获取并表达这些最终要在软件系统中体现出来的专业知识。这个要求使得那些非形式化的方法，如叙述性文本，不再适用；它也建议我们用精确的UML模型的形式来体现我们的专业知识。

但是如果我们要建立数十年内仍有价值的模型，我们就必须停止那种长期存在的习惯：使用必定会被废弃的语言来描述模型。如果我们用像Java这样的传统编程语言来表示业务规则和系统行为的话，我们的模型的价值就会在新技术取代该语言时很快丧失。

MDA的关键性前提是如果我们希望保持在系统规约上的投资，那么系统就要以一种平台无关的方法详述，即“平台无关模型”。这立刻就引出了下面的问题：我们要和什么平台无关？毕竟，即使是汇编语言也能被视为与硬件平台无关的，Java更是常被宣传为平台无关语言。那么是否用Java来详述我们的系统行为就可以了？我们必须确定Java是否会成为今后几十年里受欢迎的编程方式。历史告诉我们不是这样的。历史表明，编程语言每过2年或3年就会发生变化，当然总会留下一些人们为之狂热追随的残留痕迹。

因此我们必须瞄准更高层次的平台无关性。在这个层次中，编程语言的变化不会影响到PIM。如果我们的模型要保持数十年不被淘汰，我们就必须保证它对于以下技术的独立性：

- 硬件；
- 操作系统；
- 编程语言。

但是，我们不可能建立真正意义上与平台独立的模型。聪明的读者已经猜到我们要建议使用UML来构造PIM。我们也确实要这样做。UML是使得MDA可行的一种关键的技术；每一个使用MDA构建的应用程序都是基于一个平台无关的UML模型。通过推广使用已被广泛采纳的OMG的建模标准，MDA使得人们可以建立具有易移植性和良好的互操作性的广泛类型的应用程序。这些系统类型包括从嵌入式的、到桌面系统的、再到大型计算机的，直至跨域Internet的各