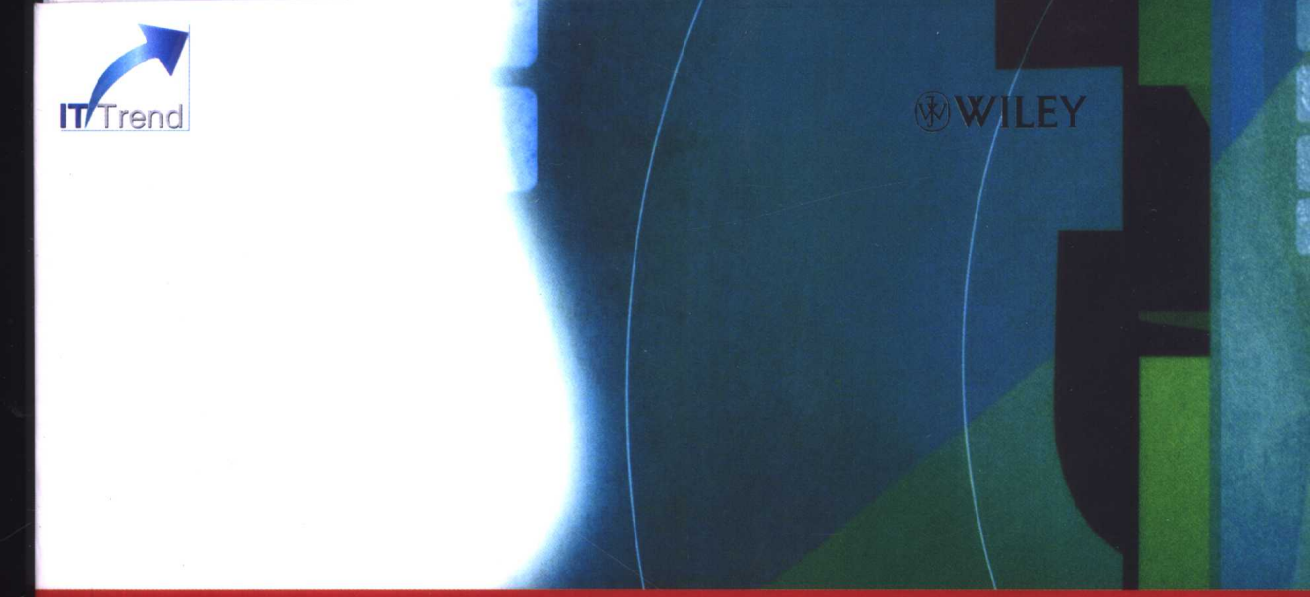


应用MDA

Model Driven Architecture

Applying MDA to Enterprise Computing

David S. Frankel 著
鲍志云 译

人民邮电出版社
POSTS & TELECOM PRESS

IT Trend 系列丛书

应用 MDA

David S. Frankel 著

鲍志云 译

人民邮电出版社

图书在版编目（CIP）数据

应用 MDA /（美）弗兰克尔（Frankel,D.S.）著；鲍志云译.
—北京：人民邮电出版社，2003.11
(IT Trend 系列)
ISBN 7-115-11785-3
Ⅰ. 应… Ⅱ. ①弗… ②鲍… Ⅲ. 软件设计 Ⅳ. TP311.5
中国版本图书馆 CIP 数据核字（2003）第 090711 号

IT Trend 系列丛书

应用 MDA

◆ 著　David S.Frankel
译　鲍志云
责任编辑　陈冀康

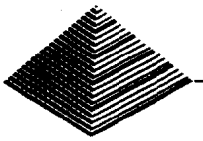
◆ 人民邮电出版社出版发行　北京市崇文区夕照寺街 14 号
邮编　100061　电子函件　315@ptpress.com.cn
网址　http://www. ptpress.com.cn
读者热线　010-67132705
北京汉魂图文设计有限公司制作
北京朝阳展望印刷厂印刷
新华书店总店北京发行所经销

◆ 开本：720×980　1/16
印张：22. 5
字数：491 千字　2003 年 11 月第 1 版
印数：1-4 000 册　2003 年 11 月北京第 1 次印刷

著作权合同登记　图字：01 - 2003 - 0667 号
ISBN 7-115-11785-3/TP • 3706

定价：45.00 元

本书如有印装质量问题，请与本社联系　电话：（010）67129223



内容提要

MDA 是 OMG 在 2002 年初确定的战略方向，将会成为对未来 IT 技术产生重大影响的开发方法学。

本书深入描述了 MDA 的概念、关键技术和所有特性，包括 MDA 基础知识、MDA 在企业计算中的地位和作用、UML 和 MDA 的关系、与 MDA 相关的 MOF 和 XMI、建模语言和建模方法、CWM 建模变换，以及涉及到代码生成的话题。

本书适合软件架构工程师和面向对象软件工程师阅读，也可作为 IT 从业人员和软件工程研究者了解 MDA 的读物。

出版者的话

在知识经济时代，IT 技术的每一次创新和发展总是带动着生产力巨大的进步和提高，也吸引着大量的优秀人才不断投身技术革命。作为 IT 行业从业者，不管你是经理人还是普通的工程技术人员，都需要敏锐的技术嗅觉、灵活的创新精神和旺盛的学习能力，以适应不断变化发展的技术趋势。

软件产业在 IT 产业中历来占有重要的地位，进入 21 世纪，这种地位继续得到巩固和凸现。纵观软件产业的发展历史，计算技术经历了“以机器为中心—以应用为中心—以企业为中心”的变化，同时，我们不难得出这样的结论：软件开发方法的进步有助于维持软件产品质量、全寿命期、生产成本的平衡。

以企业为中心的计算主要包括以下方面：

- 基于组件开发（component-based development）——将制造业中发展成熟的概念应用到软件开发中去。
- 设计模式（design patterns）——这也和制造业过程有共同之处。
- 中间件（middleware）——它在操作系统之上又迈出了提升计算平台抽象层次的一步。
- 说明性规约（declarative specification）——相对简单的说明取代了相对复杂的程序代码。
- 企业构架（enterprise architecture）——通过分离关注点来组织企业软件。
- 企业应用集成（enterprise application integration）——将孤立的遗产系统集成进企业系统的整体。
- 契约式设计（design by contract）——它有助于推动高质量的软件工程。

作为将这一系列新的趋势性技术整合到一起的开发方法，Model Driven Architecture（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指明了解决之道。也正因为如此，MDA 被面向对象技术专家预言为未来两年里最重要的方法学。

面对这一系列新的技术名词，你是否感到有些无所适从？或者你只是听说过，也许你对它们还略知一二，但相信大多数读者还没有系统地学习过相关的理论知

识。没关系，只要具备前面提到的IT从业人员的优秀素质，你肯定会对这些技术话题产生浓厚的兴趣，那么，你最需要的就是比较系统的技术书籍。

“IT Trend”系列图书是人民邮电出版社从Addison Wesley和John Wiley两大知名的国外专业出版公司引进、精心策划和出版的一套高技术定位、高制作品质的系列图书。图书内容定位覆盖了我们前边提到的那些IT趋势性技术，而且原书几乎都是全世界范围内第一本论述相关技术的专著。

这套图书的首批出版计划包括：

《IT体系结构与中间件——建设大型集成系统的策略》（*IT Architectures and Middleware: Strategies for Building Large, Integrated Systems*）

《应用MDA》（*Model Driven Architecture：Applying MDA to Enterprise Computing*）

《解析MDA》（*MDA Explained:The Practice and Promise of The Model Driven Architecture*）

《Design by Contract原则与实践》（*Design by Contract by Example*）

《基于组件开发》（*Component-based Development：Principles and Planning for Business Systems*）

我们希望这套新技术图书能够为广大读者打开一扇窗口、提供一个机会，以了解那些正在和将要对IT产业发展产生重大影响的技术趋势，帮助你准确展望和把握行业的技术前景。我们期待“IT Trend”系列图书“一步领先”的精彩，能够带给你“步步领先”的收获。

译者序

2002 年初，OMG 宣布：模型驱动构架（MDA）是它的战略方向。一年之后，业界一系列令人眼花缭乱的并购案尘埃落定（如 IBM 并购了 Rational，Borland 并购了 TogetherSoft、BoldSoft 和 Starbase），动作快的厂商已发布了具有部分 MDA 特性的产品，如 OptimalJ、BorlandC#Builder Enterprise 等。

MDA 依然是前沿技术，OMG 依然在努力继续改进和完善它。本书的作者也正在 OMG 努力工作，描绘 MDA 的美好明天。作为第一本关于 MDA 的著作，本书轻轻掀起了 MDA 的神秘面纱。从本书中，你将发现，MDA 可以用来创建良好的设计，应付多样化的实现技术，延长软件的生命期。MDA 基于广为使用的呈现、存储、交换软件设计和模型的业界标准（如 UML），并且赋予了它们新的内涵。

例如，OMG 给我们带来了 UML，为工程师交换和表达思想提供了通用的、被广泛理解的可视化语言，从而为良好设计加油鼓劲。可视化建模的运用也因此而快速普及。但是，人们常常只把可视化建模看作是一种给软件画画的方法，接下来还必须花大力气才能把这样的画翻译成代码。于是，时下 XP 等各种“强调创建可执行代码，而不是‘无关紧要’的设计”的敏捷方法广为流行，其背后的理念是“详尽的模型只不过是一堆纸，与其浪费时间纸上谈兵，还不如把这些时间用在真刀真枪写代码上面”。当然，不是说设计不重要，敏捷方法认为设计的表述形式不重要，随手涂鸦或存于开发人员脑中足矣，不必写成连篇累牍的正规文档；重要的是设计的内容以及团队成员间关于设计的有效交流。

MDA 也认识到把宝贵的时间浪费在“纸上谈兵”上是一种罪恶。但它提供的解决之道却有所不同。XP 的核心工件是 3GL 代码，而 MDA 的核心工件是模型。MDA 非常强调建立良好的设计。这些设计不再停留在纸上，而是存储在标准仓库（repositories）中。投入这些模型的智力成本不是在纸面上无所事事地等着被重新塑造成代码，相反，它们可以被 MDA 工具读取，并通过各大厂商提

供的各种标准映射自动变成 C++代码、Java 代码、C#代码、测试框架、整合代码、Java 对象、.NET 对象、部署脚本乃至软件文档，而且可以映射到 JZEE、.NET 等各种平台。而且，软件开发各阶段工件（代码、文档等）以及各版本的同步问题不再存在，因为从本质上来说，MDA 方法极大地去除了重复劳动，你不需要在文档、模型、代码中把同一问题的答案用几种语言分别描述一遍，然后丢弃“纸上谈兵”版本，反复修改代码版本，也不需要维护同一套应用系统的 Windows、Linux 或者 J2EE、.NET 等多个版本；相反，你只需要辛苦一次，剩下的事情交给各种 MDA 映射去自动解决。投入到 MDA 模型的设计努力会被多次复用来生成各种组件。这些模型可以在应用的整个生命期中不断更新，以精确反映经过多次维护的软件现在是怎样工作的，而不是停留在设计阶段结束的一幅静止画面上。简而言之，MDA 是一个在今天多平台 IT 环境中创建良好设计的构架。

XP 等敏捷方法以其轻巧、灵活、务实赢得了我的心。但深入接触 MDA 后，我开始感觉到，严谨、精确也是一种美，而且模型的抽象层次确实比 3GL 代码高，正如 3GL 代码抽象层次比汇编代码高。这就意味着实实在在的生产力提升。MDA 不是一个针对目前软件开发中存在的问题的“quick and simple”解决方案，而是在现有技术的成熟普及的基础上迈出的影响深远的一步，它触及了软件开发的各个阶段。

鉴于现在敏捷方法的流行程度，有学者试图把 MDA 也归入敏捷方法。这种说法也有一定道理，因为 MDA 的极佳同步机制，使得在 MDA 中迭代开发乃至反复修改设计都不会给项目造成太大困扰，这点和“船小好掉头”的敏捷方法有类似之处。但我个人觉得，就 MDA 给建模注入的精确性、严谨性（这是必要的，否则无从自动生成精确的 3GL 代码）来看，MDA 的血统是“工程”，不是“工艺”。如果你觉得软件开发是一种艺术而不是一种工程，如果你觉得 informal 要比 formal 好，如果你觉得思维过于严谨会扼杀你的创造力，那么很抱歉，这本书不合你的口味。

此外，需要提醒读者的是，目前 MDA 还处在发展的初级阶段。这也就意味着，读完这本书后你并不能立刻让濒临 deadline 的项目起死回生，也不会让你的公司在 3 个月内转亏为盈，更不可能马上一星期干完 3 个月的活从而可以在剩下的时间内从容度假。本书能带给你的收益是，让你在下一次“生产力大解放”的浪潮中可以站稳脚跟，把握住机会（在任何跃升期、转型期，机会都会大量涌现），而不至于被浪头打得茫然不知所措。

最后，作为译者，我希望自己的努力为 MDA 技术的普及起到正面的推动作用，希望读者可以从这本书中受益。MDA 是新技术，本书是关于 MDA 的第一本著作。很多术语的译法没有现成的参考，本书中若有术语处理不当之处乃至对技术把握不准确之处，欢迎广大读者指正。我的 E-mail 是：zmelody@sohu.com。

鲍志云

于南京

2003 年 8 月

序

1951 年 7 月 14 日，在我出生后不久，美国人口普查局购买了第一台 UNIVAC 计算机。这是计算机产业历史上有据可查的第一笔买卖，而现在这个产业已经无比庞大了。那台 UNIVAC 是第一台大型机，尺寸大概相当于一个可以放一辆车的车库，并且需要 125kW 的供电量来加热和冷却它的 5 200 个真空管。

有了那台售价百万美元（如果按现在的美元币值计算，这个数字还要乘以 7）的 UNIVAC，人口普查局的程序员获得了“巨大的”1 000 字的内存容量。然而，和那时其余可供选购的计算机比起来，这已经非常强大了，因此在其后的 5 年中 UNIVAC 至少又卖出了 45 台。同时，UNIVAC 也孵化出了许多的竞争者。其中有一个竞争者叫 IBM，后来成了世界上最大的公司，它的支柱性业务就是大型机。

无论如何，最早的程序员通过努力完成了他们的工作。显然，从区区 1 000 个字的内存起步，他们的注意力只能集中在如何针对某一种特定机器来优化内存的使用和计算的速度。多年来，软件工程几乎完全被可用硬件的要求和局限所推动。这极大地影响了对所计算的问题的类型的选择，也影响了软件开发的方式。

令人惊讶的是，虽然摩尔那条很夸张的定律支配着硬件能力的指数增长，但这种以特定平台为中心对软件进行思考的方式在许多方面依然没有改变。1970 年，我把一台 IBM 360 MVS 的一个分区塞得满满的；在 1980 年，我绞尽脑汁使得我在 HP2100MX 小型机上的程序保持在 64KB 以内；在 1990 年，我依然在担心我的代码是否能够编译成足够小的 DLL，以能够在 PC 上高效地加载，我还在担忧我是否用掉了太多了的 Windows 句柄；在 1995 年，当我为 CORBA 互联网互操作性（Internet Interoperability，IIOP）标准工作时，我发现自己依然在想方设法提高每一个比特的利用率，以减少在网络上传输的消息的尺寸。

然而，从 1951 年到今天，重大的质变确实已经发生了。今天，典型的 IT 部

门的软件开发过程正越来越多地围绕着对要解决的业务问题的建模，而不是编写代码的细节。这一论断之所以成立，不是出于开发者的个人意愿，而是因为需要解决的业务问题越来越复杂，我们不得不采取需求驱动的方法。有些开发者（可能不情愿地）在摸索像 UML 这样的建模语言上所花的时间，可能和花在 Java 那样的传统编程语言和 XML 等描述语言上的时间一样多。

这一方法的萌芽可追溯到 1960 年。那时，第一种平台无关的面向商业的编程语言 COBOL 刚刚诞生。COBOL 的编程方式显式地将软件和硬件相分离，这使得程序员可以更多地思考业务逻辑，而极大减少了“纠缠于机器层次上处理每一个比特”所花的时间。COBOL 同时还给软件开发引入了相当数量的结构，这迫使用户至少要把程序流逻辑和数据描述、环境定义分开。

接下来的 20 年里，在大型机软件市场上 COBOL 不断增长的主导地位促成了整个软件产业事实上的构架和设计标准。这是软件开发的第一个“黄金时代”，那个年代里建立的许多系统依然是今天一些主要公司的数据处理的中坚力量。它们依然像哥特式大教堂那样醒目地矗立在后来建起的廉价公寓群中。不幸的是，当今很多软件就是“模子”和“粘合剂”搭建而成的，能凑合着用而已。

个人电脑的诞生在很多方面都对计算技术做出了重大贡献。但是，具有讽刺意义的是，也正是个人电脑的诞生结束了这个黄金时代。PC 不经意间开启了一个软件开发的混沌新时代。因为历史的偶然性，像 COBOL 这样的结构化、平台无关语言向 PC 移植的过程很慢。“乱世出英雄”，其他编程语言和开发工具纷纷崛起，抢滩 PC，填补了这片真空地带。为了迎合被早期的 PC 吸引而来的业余开发者这一新的群体，这些编程语言容忍甚至鼓励低层的、平台相关的编码风格。

因此，在某种意义上，始于 20 世纪 80 年代，在 90 年代极度繁荣的 PC 时代，是计算技术的第二次伟大复兴。这个时代把软件推入各个专门领域，推到了用户面前，这是前所未有的。但是，随之而来的用多种语言编写软件的浪潮也确实让传统的 IT 软件开发群体有些喘不过气，而且这在很多方面实际上将系统化软件构架和设计的“时钟”往回拨了。另一次相似的“混沌浪潮”则伴随互联网的商业化而席卷业界。

当然，在 PC 时代和互联网时代，许多业界的智者依然建议开发者们“先设计后编码”，并且，近来还建议“先架构后设计”。事实上，在过去的 20 年中，在这方面有了许多积极的创新。但是，因为大部分开发人员是在以代码为中心的 PC 时代成长起来的，所以积习难改。对于这些开发者而言，把设计从编程中抽象出来，被本能地当作令人烦恼的经历，更不用说构架了，这样会“推迟了编码”。因为对此感到不适应，他们频繁（如果不是有意的）破坏设计过程的意图，从而“再一次地证明了”（至少对他们本人而言如此）他们如果尽早开始编码的话那么情况

会好得多。

结果就是，花了将近 20 年我们才再一次达到这个高度，可以再次在 IT 软件开发中建立某种精确性，而这种精确性是在过去的软件黄金时代中早已获得的。差不多在最近的 10 年里，有一种想法看起来似乎是一种回归，那就是独立于实现的设计和构架对于开发过程确实有内在的好处。有些时候，人们几乎听不到这一理性的声音，厂商们不间断的市场炒作和技术吃语把它淹没了，这就是我们所见到的软件产业。虽然如此，一种新的方法似乎正在开发人员和厂商中间获得多数人的支持，这种方法重建了第一个黄金时代中最佳的实践方法，同时把伴随 PC 和互联网革命而来的软件工程的创新融合其中。

本书所描述的模型驱动构架（Model Driven Architecture，MDA）就是往这个方向迈出的重要一步。是什么使得 MDA 同其他无数泛滥于软件社区的三字母缩略语相比显得如此与众不同？第一个理由，MDA 是由 OMG 推动的，OMG 是软件产业界最大的联盟。OMG 拥有令人羡慕的“光辉的过去”——它发布并维护了业界一些最成功的标准，比如 CORBA 和 UML。

另外，在 OMG 内部，MDA 从系统和软件供应商群体获得了异乎寻常的强有力支持。通常，这种程度的一致同意和支持需要好些年才能获得，但是，即便像 IBM、Sun 和微软这样不共戴天的对手们也都在支持 MDA 这一点上达成了一致[1]，并且积极地支持 MDA 所包含的主要标准——UML、XMI、MOF、CWM 以及 JMI 等。毫无疑问，它们会就细节问题争论不休，但是它们都坚定地支持这一方法。这意味着，MDA 成长和繁荣所必需的主流工具和平台的支持已经指日可待了。

最后，MDA 并没有声称要大规模取代以前的计算方法、语言或者工具。相反，它试图融合它们，使得每个人都可以按照他们自己的节奏，根据他们自己的需要，平稳地过渡到 MDA 的世界。MDA 同时也被特意设计得足够灵活，可以适应不可避免会快速浮现的软件新技术。毫无疑问，就像 PC 和互联网那样，新技术必定会很快产生，并且将否决我们以前对计算做的假定。

因此，我相信 MDA 实际上很有希望给软件构架实践带来新生，并促进软件开发的另一个黄金时代的开创。我们不会等太久，因为正如本书所说，需要计算的业务问题的复杂性不断增加，以至于正在挑战上一代软件开发方法的极限。

本书中肯地指出，为了应付这一复杂性，我们需要像 MDA 这样的设计和架构的标准来代替早先的方法，充当企业软件开发整个过程的焦点。这让人清晰地

---

[1] 译注：译者着手翻译此书时，业界几个主要的开发工具供应商刚刚有了大的并购动作，比如 IBM 并购了 Rational，Borland 并购了 TogetherSoft、BoldSoft 和 Starbase。

回忆起 30 年前的情形：人们急切地需要 COBOL 和其他 3GL 来取代机器码和汇编语言作为业务应用的主要编程工具。

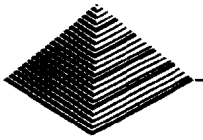
不过，肯定会有一些开发者试图忽视 MDA 这类技术的发展，并且像以前一样埋头工作，榨干他们已经掌握的以代码中心的开发技巧的最后一点潜力，直到最后一刻。但是，在不久的将来，毫无疑问，最成功、最有效率的开发者，将是那些愿意大步向前拥抱 MDA 所代表的以设计和构架为中心的方法的人。

因此，本书非常重要。作为第一本详尽论述 MDA 的著作，本书有机会为软件开发群体如何接受 MDA 设置准绳。幸运的是，作者 David Frankel 不仅是一个好的技术作家，还是 MDA 的公认权威；他还是在 OMG 推动 MDA 的策略和技术发展的骨干之一。

同样重要的是，我可以根据个人的了解告诉你，David 是一位造诣深厚的开发者和开发经理，他知道把真正的系统做完交货意味着什么。结论是，虽然这不是一本详细的说明书，但本书充满了如何将 MDA 运用于今天真正的开发者所面临的真实问题的例子。

简而言之，我找不出第二个更合适的人来把 MDA 的概念以及它背后的细节如此清晰地阐述给软件开发者和业界技术人员。因此，如果你是 MDA 的初学者，或者在理解 MDA 过程中遇到了麻烦，那么我的建议很简单：开始阅读本书吧。本书将为你掌握 MDA，并立即把它的概念直接而有效地应用于你自己的环境和问题打下基础。

Michael Guttman



前　言

计算机业界永远都在探索提高软件开发效率的方法，当然这一方法对于所开发出来的软件能够提高质量和延长生存期。面向对象、基于组件开发、模式、分布式计算基础构架等新方法都为这一探索贡献了自己的力量。

模型驱动构架（Model Driven Architecture，MDA）同样会做出重大的贡献。MDA 无意独领风骚而使得其他方法黯然失色，相反，它同其他方法相辅相成，可以起到“功率放大器”的作用，从而进一步改进我们开发软件的方式。

什么是模型驱动构架（MDA）

MDA 就是把建模语言当编程语言来用，而不只是当作设计语言来用。用建模语言编程可以提高生产率，改善质量，并使软件产品生存期更长。本书解释了为什么 MDA 会有这些优点，并介绍了支撑 MDA 的技术。

谁在使用 MDA

人们用 MDA 来生成实时及嵌入式系统已经有年头了，虽然 MDA 这一术语是晚些时候才被创造出来的。现在 IBM、Oracle、Unisys、IONA 以及其他开发商都在把基于 MDA 的技术融入到它们自己的企业级软件解决方案中去。

长期的过渡

在分布式对象计算的早期，CORBA 的创始人之一 Mike Guttman，曾被邀请在一个业界聚会上作演讲，将他对“技术的未来趋势”的远见同众人分享。他用

幻灯片作了演示，告诉大家，将来会有那么一天，分布式对象基础构架和在这个基础构架上运行的组件库会成为主流。

听众中有人问他认为在什么时候这一切会发生，得到的回答是，他觉得这至少要过 10 到 15 年。他紧接着补充说，哪怕在过渡的早期阶段，也能从中获得重大商业收益，但看来主办者们对此并不感冒，事实上他们还有些懊恼。他们的评论是，Mike 在暗示没有人应当为 10 年后才会带来收益的技术投资。

我把这本关于 MDA 的书展示给大家，也冒了类似的风险。毫无疑问，现在 MDA 的原则可以很好地投入实践了。我们已经有了一些基于 MDA 的特定技术，在我写这本书的时候还有一些技术正在浮现。使用模型将软件开发的某些方面自动化的工具至少形成了价值 5 亿美元的产业。不过，我依然要强调，MDA 是一个尚处于萌芽阶段的技术，要使之发挥全部潜能，我们至少还需要经过几年的努力。

MDA 和对象管理组织（OMG）

在 2002 年初，OMG 宣告模型驱动构架（MDA）是它的战略方向。OMG 尚处于定义这一进程的起步阶段，而我本人深深卷入了这一努力之中。我希望本书对这一起步阶段能起到一点指导作用，但我不保证我的想法同 OMG 完全一致。

作为几种建模语言标准的“大管家”，OMG 处于支持这一新产业发展的关键地位。但是其他标准团体，比如 JCP[1]、ebXML 以及 RosettaNet，都在以各种技术在不同的应用领域制定应用 MDA 原则的规范。随着产业界逐渐积累更多经验，标准团体还会发布其他的基于 MDA 的标准。本书为这些标准的发展提出了一些方向性的建议。本书还指出了已经标准化的建模语言的一些缺陷。为了完全实现 MDA 的设计方案，必须改正这些缺陷。

本书的目的

虽然 MDA 在实时和嵌入式系统的世界中取得了许多成功，但直到最近，很少有人全面地将其应用于开发和整合客户、收款账号、供应链、业务集成等事务管理的业务系统。

本书主要关心企业系统环境中的 MDA。我希望让企业系统的构建者们理解 MDA 目前的成就，以及 MDA 所提供的改进软件开发的潜能。我还希望帮助工具

---

[1] 译注：Java Community Process（Java 社区项目），是制定 Java 标准的机构，由业界厂商组成。

编写者们理解，他们能够做些什么来释放这一潜能。我想让这两类读者都能意识到，在试图将 MDA 推进到可以一致地支持企业系统的开发和整合的过程中，他们将会遇到些什么问题。

非本书的目的

为了让 MDA 成为成熟的技术，为 MDA 定义一个详尽的概念框架是必须完成的任务。本书并不试图定义这样一个框架，因为这取决于在 Aspect-Oriented Programming[1]和 Intentional Programming[2]这些领域的概念的发展。

另外，虽然本书阐释了一个全面基于 MDA 的企业构架会是什么样子，但本书不尝试定义这样一个构架。

本书主要关注 MDA 基本技术，而不包括对具体方法的详尽描述。但我无意暗示这类内容并不重要。我希望别人能够觉得我所做的基础工作有助于他们去完成更为艰巨的任务。

本书的结构

本书分为几个部分，每部分包含几章的内容。

第一部分：MDA 导论。陈述 MDA 的动机，带你鸟瞰 MDA 的方方面面。

第二部分：MDA 基础技术。手把手地教给你 MDA 的基础技术。这是本书的核心部分，包含了大部分材料。

第三部分：高级话题。概述了 MDA 一些更远大的目标。

结语。是对 MDA 之未来的“现实检测”。

读者如果想找一本“给经理人写的 MDA 书”，那么可以只读前言、第一部分、结语。更关心技术细节的读者还需要读第二和第三部分。

致谢

本书的大部分内容综合了他人的工作。我特别想要感谢 Mike Guttman，他是我 30 多年的导师和朋友，他帮助我定稿，提供了重要的建议和反馈，当然，还为

[1] 译注：有关 Aspect Oriented Programming 的内容可以参见《程序员》2002 年第 11 期的技术专题。

[2] 译注：在 Intentional Programming 中，程序源代码不以普通文本表示，而是位于层次数据库之中，数据库的每一个节点都是编程语言的一个构造的实例。编辑器含图形布局功能，编译器的结构也是可扩展的，用户可增加新的关键字。IP 最令人心动的概念是可以用问题领域的抽象概念来编写程序。人们也在试图把像 C++和 Java 这样的传统语言包容于 IP 之中。

本书写了序言。

我还要感谢 Jack Greefield 的贡献，他是这个领域的先驱者，我从他那里学到许多东西。

我还要特别感谢 Scott Ambler、Grady Booch、Steve Brodsky、Steve Cook、Philippe Kruchten、Scott Markel、David Mellor、Steve Mellor、Mike Rosen、Bran Selic、Oliver Sims、Jos Warmer，谢谢他们对我的书稿提出的宝贵意见。

我还要感谢其他一些人，他们在关于 MDA 的想法上帮助过我。这些人包括 Don Baisely、Conrad Bock、Cory Casanave、Desmond D'Souza、Simon Johnston、Sridhar Iyengar、Haim Kilov、Cris Kobryn、Wojtek Kozaczynski、Jason Matthews、Guus Ramackers、Jim Rumbaugh、Ed Seidewitz 和 Richard Soley。

关于作者

David S. Frankel 曾在 QuarterSoft 公司、IONA Technologies 和 Genesis Development 公司担任高级职位。在复杂的大规模分布式计算系统领域，他是一位声名卓著的专家。Frankel 曾多次在 OMG Architecture Board 担任工作。

David Frankel 的联系方式：

Email：df@davidfrankelconsulting.com 或 dfrankel@quartersoft.com

Tel：001 530 893-1100