PROGRAMMER TO PROGRAMMER™

Model-Driven Development with Executable UML

Executable UML
模型驱动开发

(美) Dragan Milicev 著
车立红 译

wrox™
WILEY
清华大学出版社

Executable UML

模型驱动开发

(美) Dragan Milicev 著

车立红 译

清华大学出版社

北 京

Dragan Milicev
Model-Driven Development with Executable UML
EISBN: 978-0-470-48163-9

北京市版权局著作权合同登记号　图字：01-2009-7473

图书在版编目(CIP)数据

Executable UML 模型驱动开发/(美) 米利塞维(Milicev, D.) 著；车立红 译. —北京：清华大学出版社，2011.10

书名原文：Model-Driven Development with Executable UML

ISBN 978-7-302-25631-1

Ⅰ. E…　Ⅱ. ①米…　②车…　Ⅲ. 面向对象语言，UML—程序设计—教材　Ⅳ. TP312

中国版本图书馆 CIP 数据核字(2011)第 096083 号

责任编辑：王　军　王滋润
装帧设计：牛艳敏
责任校对：胡雁翎
责任印制：杨　艳
出版发行：清华大学出版社　　**地　　址**：北京清华大学学研大厦 A 座
http://www.tup.com.cn　　**邮　　编**：100084
社　总　机：010-62770175　　**邮　购**：010-62786544
投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn
质　量　反　馈：010-62772015，zhiliang@tup.tsinghua.edu.cn
印 刷 者：清华大学印刷厂
装 订 者：北京市密云县京文制本装订厂
经　　销：全国新华书店
开　　本：185×260　**印　张**：38.25　**字　数**：1029 千字
版　　次：2011 年 10 月第 1 版　　**印　　次**：2011 年 10 月第1 次印刷
印　　数：1～4000
定　　价：85.00 元

产品编号：034807-01

作 者 简 介

Dragan Milicev 博士是贝尔格莱德大学电子工程学院计算机科学系的副教授。他是 Serbian Object Laboratories d.o.o. (SOL, www.sol.rs)公司的创始人兼 CTO，这是一家软件开发公司，致力于使用模型驱动的技术构建软件开发工具，并构建自定义应用程序和系统。在构建复杂的软件系统方面，Dragan Milicev 拥有 25 年的丰富经验，曾在 20 多个学术和国际行业项目中担任首席软件架构师、项目经理或顾问。值得一提的是他曾担任大部分 SOL 项目及其产品的首席软件架构师和项目经理，这些 SQL 产品包括：SOLoist，一个用于信息系统的快速应用程序模型驱动开发架构；SOL UML Visual Debugger，世界上首批 UML 可视化调试程序之一，针对 UML 建模工具 Poseidon 而设计；SOL Java Visual Debugger，用于 Eclipse 的插件，支持使用 UML 对象图表对测试对象结构建模。他曾在大部分知名的科学和专业的报刊、杂志上发表论文，为模型驱动的开发和 UML 的理论和实践作出了巨大贡献。Dragan Milicev 以前曾在塞尔维亚出版了 3 本有关 C++、面向对象编程和 UML 的书籍。您可以通过 dmilicev@etf.rs 联系他。

致　谢

非常感谢 Bran Selic，他是模型驱动软件的开拓者之一，是最值得尊敬的权威人士之一，为 UML 的发展作出了重要贡献。Bran Selic 仔细审校了本书的原稿，并提出了宝贵建议，从而改进了本书的结构并提高了技术准确性。

特别感谢我在 SOL 的同事，他们参与了 SOLoist 的开发及其在很多行业项目中的应用。他们付出了很多努力，使得本书讲述的很多理念得以实现，并使概念实用且高效地用于行业系统中。

感谢我的研究助理，并感谢贝尔格莱德大学电子工程学院计算机科学系的研究生和大学生们，他们参与了本书所表述的一些理念和概念的研究与开发。此外，他们还对本书的初稿提出了很多宝贵建议。

感谢让我担任顾问的公司。很荣幸能有机会和同事们召开头脑风暴会议，并一起讨论本书表述的一些想法，很高兴能有机会将那些想法付诸于实践。

另外，塞尔维亚科学部还依据国家技术开发计划(批准号：TR-13001)对本书中方法的开发和实现提供了一定支持，在此一并表示感谢。

最后，感谢爱女 Mina、儿子 Miloš 和 Jovan 以及妻子 Snežana，在编写本书的过程中，她们给予了宽容、理解和支持，谨以此书献给她们。

前　言

在软件系统的规划、设计、开发、测试、部署、维护和使用中，其逻辑复杂性是导致出现问题和错误的主要因素之一。要知道构建复杂的软件系统需要细致的规划、出色的体系结构设计以及控制良好的开发过程。许多书籍、论文和软件工程课程也都论述了此问题的解决方法，但很多软件项目仍然遭遇了失败、延期或预算超支的命运。构建或维护复杂的系统(可能是软件系统，也可能不是)总会出现一些错误及未满足需求的情况，因为系统的构建者在一次处理太多的细节及相关组件时，出错是在所难免的。

然而，逻辑复杂性并不完全是软件系统所固有的。一方面，对于软件系统所处理的问题域，存在一个不可避免的复杂性组成部分。首先，根本复杂性(essential complexity)一词是指问题域所固有的那部分逻辑复杂性，该复杂性不是由问题的解决方案或实现技术引入的。因此，根本复杂性是自然存在的部分，不能被删除，存在于问题的每个解决方案中，因为针对该问题的简单解决方案根本不存在。接下来是偶发复杂性(accidental complexity)，它与根本复杂性相反，完全是由在解决方案中应用的实现技术、工具和方法引入的。与无论采用什么方法解决问题，都不能避免根本复杂性相比，偶发复杂性是由所使用的方法引入的。

作为一门学科，软件工程的主要任务之一就是找到能使偶发复杂性降到最低的方法。任何出色的软件体系结构、设计和实现，都会将偶发复杂性降到最低。

有时，偶发复杂性是由低效的规划或项目管理、项目优先级太低等失误引起的。然而，一些偶发复杂性总是在解决问题后出现。例如，在许多程序中因内存溢出错误而导致的偶发复杂性，就是由于有人想使用计算机解决问题引起的[Wiki]。

出现偶发复杂性的另一个重要原因是：为软件系统开发选择的技术或过程不匹配或不成熟。如果可用的技术(包括用于软件开发的语言)要求开发人员编写超出实际需要的指令或执行不必要的操作以完成某个决定，开发就会人为地复杂化。使用汇编语言实现非平凡算法以及使用文件系统界面构建数据库应用程序，是此类选择了不匹配技术的简单、极端范例。此类偶发复杂性的一个不太明显的例子，是当拖动一个对象并将其放到另一个对象上时，必须编写一些代码以说明要在两个对象之间建立关系。而这可以用一种更简单更直接的方式实现：示范。

因此，降低偶发复杂性的最基本的手段之一是提高用于软件系统开发技术的抽象层次，并以一种能够更好地匹配这些系统的问题域的方式来实现。提高抽象层次是软件工程作为一门学科演进的主要特征之一。Bran Selic曾经说过：“自从编译器问世后，软件工程还没有突破性的进展”。换句话说，当可以用更适合我们的语言(该语言可自动转换为机器语言)

而不需要用计算机硬件能理解的语言和其沟通时，我们便实现了软件工程领域最重要的突破。自此以后，需要做的一切工作，基本就是提高用于程序机的语言的抽象层次了。

提高抽象层次的重点是实现更好的表达力(expressiveness)。通过使用一种更有利于解决问题的语言，可以用更少的“文字”表达更丰富的“内容”，从而达到事半功倍的效果。此外，编写的文字不再是与机器交流的唯一方法。图片(如图表)、动作和声音(如口语)早已成为人和计算机程序交流的方式，它们同样也可以用于软件开发中。

本书讲述了某种软件系统的开发技术，并讲解了一种通过提高抽象层次和降低偶发复杂性来提高开发效率的方法。

模型驱动开发是一种提高抽象层次的方法，已成功应用了十多年。其基本前提是使用模型而非(只)使用代码来指定软件。模型通常为非线性的形式，这与本质上为线性的代码正好相反[1]。非线性意味着相比每个元素最多只能有两个相邻元素的简单序列而言，模型的元素可以采用更自由的方式关联。因此，模型通常使用可视化表示法(如图表)而非纯文本来呈现。

本书描述的软件开发方法为模型驱动方法。

统一建模语言(Unified Modeling Language，UML)是一种用于软件建模的标准语言。此概念于20世纪90年代中期提出，1997年首次实现了标准化。它是一种用于对各种软件系统进行建模的通用语言。

本书描述的方法以UML为建模语言[2]，并遵从在参考文献中给出的定义和规范[UML2]。

然而，本书并没有涵盖所有种类的软件系统，而只讲述了一种称为信息系统的特殊软件系统。本书的第 I 部分对此给出了精确描述。简而言之，本书重点讲述了那些具有以下特性的应用程序：

- **复杂的概念性的基础理论**——应用程序依赖于来自其问题域的相当丰富的概念、特性及关系集。
- **大规模的动态实例化**——在开发期间，应用程序要处理其概念和关系实例涉及的巨大空间。这些实例以动态方式被创建、修改、检索、查询、提供和删除，它们传统上被称为数据对象。
- **运行时空间的持久性**——应用程序依赖于为其提供支持的数据库。
- **交互性**——应用程序要通过用户或机器界面与用户或其他系统进行大量交互，以实现其目的。

本书重点介绍了如何使用 UML 对信息系统进行模型驱动的开发。

然而，UML 不是一种完全形式语言。这意味着它的语义并未在其所有元素中都清晰地定义。因此，在使用 UML 时，不能像使用传统编程语言那样，可以由机器清晰地解释，如编译和执行软件系统的规范。因此，语言必须有形式清晰的语义，也就是它具有运行时效果的每个概念的解释都应该是独一无二的。

此外，UML是一种通用的建模语言，可以针对具体的问题域配置它。例如，标准的UML会留出一些语义变化点，这样就允许配置文件用几种方式解释某些语言中的概念。配置文

1. 注意，因为代码表示的是字符串，所以其为顺序形式。机器和人按顺序逐个字符地读取代码。有时，为提高其可读性，机器以二维视口的方式呈现代码，但本质上仍是顺序形式。
2. 编写本书时，最新的 UML 标准为 2.2 版。本书描述了此 UML 版本，且基于该参考文献[UML2]。

件还可以减少在特定问题域中使用的语言概念集，或以可控制的方式扩展概念的语义。通过这种方式，配置文件可以对标准语言进行自定义，使其成为完全形式(即可执行)语言。同时，内置在此配置文件中的模型也代表了该软件的实现，并且由于该模型是可执行的，因此并不仅仅是设计的非形式化骨架。

针对所描述的应用程序域，本书提供并描述了一个新的UML可执行配置文件。但它是UML具有形式化且可执行语义的几个现有配置文件之一，针对信息系统域量身定制[3]。

要知道，一方面，作为构建信息系统的基本技术，关系范式已得到验证并被广泛接受。另一方面，作为一种更抽象且表达力更强的概念，另一种软件开发范式面向对象已在编程领域成功使用了数十年。UML就是基于该对象范式的语言之一。

通过使用面向对象编程(object-oriented programming，OOP)语言在底层关系数据库(该数据库可能是通过执行对象-关系映射的数据持久层来访问的)上实现行为(或称业务逻辑)，基本上已完成了面向对象与信息系统开发的结合。该方法已部分地取代了符合关系范式的第 4 代编程语言的使用(在目前的技术发展阶段，这一广为使用的方法存在对象和关系范式之间的结合不完整或非形式化，导致开发中断的缺点)。

本书讨论了这些技术存在的问题、影响开发的方式及解决这些问题的办法。

主要讨论了以下内容：

- 信息系统的应用程序的快速开发技术。
- 使用对象范式和信息系统模型驱动开发的方法。
- 用于信息系统模型驱动开发的一个 UML 可执行配置文件。

本书的目标：

- 深入详解用于构建信息系统的模型驱动开发方法和 UML。
- 说明如何通过使用对象范式、模型驱动的开发方法及形式化且可执行的 UML 配置文件(非关系范式或与面向对象不完整的结合)，来更好地理解并更有效地开发信息系统。

请注意，本书并非是：

- **一本关于整个通用 UML 的教程、参考文献或教科书**——本书确实讲述了有关 UML 的大部分知识，但并没有涵盖有关 UML 的全部内容。相反，本书重点讲述了在构建信息系统中很可能会用到的 UML 概念和相关知识。
- **一本关于对象范式或任何传统 OOP 语言的完整教程**——然而，本书确实讲述了面向对象的基本概念。
- **一本关于信息系统或所有相关技术的完整教程**——然而，本书第Ⅴ部分确实简要概述了有关信息系统及构建技术的主要内容，包括体系结构、关系范式、实体-关系、结构化分析及 SQL。
- **一本有关软件系统(尤其是信息系统)开发过程的全部内容的书籍**——然而，本书的第Ⅳ部分确实讲述了所提议开发方法的快速实用指南。

3. 因此，术语“可执行UML”并不是指UML任何特定的可执行版本，而是一个通用术语，表示任何标准UML形式且可执行的规范。本书讲述的就是这样一种可执行的规范。

- **一本描述用于构建信息系统的模式或其他技术及构件的书籍**——本书并没有通过复杂的综合示例及案例分析来讲述如何构建信息系统，而是用一些小的简单特定示例阐明了概念和原理。

0.1 本书读者对象

本书可供分析、指定、设计、建模、开发或测试信息系统的软件从业人员阅读，适用于想学习可执行 UML 配置文件及模型驱动的快速应用程序开发的读者。此外，系统分析员、系统和软件架构师、设计人员、开发人员和测试人员也可以从本书中获益。

本书也适用于想要探索新的软件开发策略、方法，尤其是研究模型驱动开发和编程的读者。本书还介绍了一些有趣的值得深入探讨的新概念和想法。

还可以将本书用作有关信息系统、模型驱动的软件工程及 UML 的高等教科书。

阅读本书时，读者最好了解对象范式或任何一种 OOP 语言的有关知识，但不了解也没关系，不会影响对本书的理解。本书将逐步介绍面向对象的基本概念和原理。

同样，如果读者具备有关关系范式或任何关系数据库管理系统(relational database management system，RDBMS)的知识，会有助于理解本书的内容，但不了解相关知识也不会影响本书的阅读。第Ⅴ部分专门为那些不熟悉这些内容的读者提供了概述性描述。另外，熟悉这些主题的读者也将从中领略到范式的转变。

最后，阅读本书完全不需要具备UML的有关知识。本书是针对UML初学者(配置文件)的教程，但也清晰地阐述了很多容易混淆的UML概念和语义，因此，经验丰富的UML老手也可以从本书中获益。

阅读本书的前提是要大致了解编程的有关知识。最好具备构建信息系统的知识和经验，但这些不是必备要求。

0.2 本书的组织结构

本书分为以下几部分：

- **概述(第Ⅰ部分，第 1 章～第 3 章)**——这部分首先简述了信息系统的有关知识，然后详述了传统的信息系统开发技术及其优缺点。这部分明确指出了广泛用于构建信息系统(尤其是关系建模或关系-实体建模)的传统范式存在的主要问题，以及面向对象(和 OOP 语言)和关系建模不完整的结合，并且分析了本书所述方法的原因。
- **OOIS UML 概述(第Ⅱ部分，第 4 章～第 6 章)**——这部分对本书提出的 UML 的可执行配置文件进行了简要概述，此配置文件称为 OOIS UML 配置文件。此外，还简要概述了在本书随后部分将要详细介绍的主要概念和想法。
- **概念(第Ⅲ部分，第 7 章～第 16 章)**——此为本书的核心部分，详细介绍了 OOIS UML 的概念和语义。
- **方法(第Ⅳ部分，第 17 章～第 19 章)**——这部分对使用 OOIS UML 配置文件构建信息系统提供了一个快速指南。

- **补充内容(第Ⅴ部分，第 20 章～第 24 章)**——这部分为当今广泛用于构建信息系统的传统技术提供了辅助资料，不学习这部分也可以理解本书的主要内容。这部分内容包括：信息系统的一般特征摘要、软件工程过程的一些基本知识、关系范式、实体-关系建模、结构化分析以及对象范式的一般原理。适合对这些主题及传统技术不太熟悉，但又有兴趣了解的读者阅读，而那些对此已拥有丰富知识的读者也可以借此回顾一下相关内容。

如果熟悉信息系统的概念及开发信息系统所使用的传统技术(包括关系数据库和实体-关系)，那么从头开始阅读本书即可。本书的前 3 章分析了读者可能会在工作中遇到的问题，并对问题的原因进行了解释说明，本书的核心部分将讲述这些问题的解决方案。

如果不熟悉这些传统技术，仍然可以从头开始阅读本书。然而，也可以跳过“补充内容”部分，仅学习还不了解的技术的一些基本知识。但是，此部分不是理解本书主要内容的前提。

最后，如果迫切想要了解本书的所有重点、所包含的新增内容及原始信息，那么，只需阅读第Ⅱ部分即可。读完第Ⅱ部分后，可以回到前面开始学习，也可以接着往后阅读，一切全凭您的喜好而定。

0.3　支持软件及合作站点

即便没有成熟的工具支持，也可以使用本书所描述的方法。本书作者曾参与过几个成功的行业项目，在这些项目中，只使用了定制的商用或热门开发工具来局部地支持方法中的一些活动(如 UML 建模工具、定制的代码生成器以及对象-关系映射架构)。即便没有成熟的工具支持，所提议的方法也能够提高工作效率，改进所生产软件的质量，原因是提高了抽象层次，建模语言和语义的表达力更强，软件系统的体系结构清晰，并且有一个控制良好的开发方法。

然而很显然，只有基于计算机的工具提供强大成熟的支持，才能获得所提议方法的全部优势。采用适当的工具支持，所提议的UML配置文件可以有许多不同的实现方法。本书作者就是这些方法的发明者，并且是SOLoist[4]工具的首席架构师，该工具自2000年开发问世后，已成功应用于各种行业项目中，并且支持本书中描述的许多概念。

可访问 www.ooisuml.org 查看有关本书讲述的配置文件、方法、尚未解决的问题、进一步改进的建议，以及它们在实际项目中的实现和应用的相关内容。

0.4　勘误表

尽管我们已经尽了各种努力来保证正文和代码中不出现错误，但是错误总是难免的，如果您在本书中找到了错误，例如拼写错误或代码错误，请告诉我们，我们将非常感激。通过勘误表，可以让其他读者避免受挫，同时也有助于我们提供更高质量的信息。

4. SOLoist 是 Serbian Object Laboratories d.o.o. (SOL)的商标。

请给 wkservice@vip.163.com 发送电子邮件，我们将会检查您的反馈信息。如果是正确的，我们将在本书的后续版本中采用。

要在网站上找到本书英文版的勘误表，可以登录 http://www.wrox.com，通过 Search 工具或书名列表查找本书，然后在本书的细目页面上，单击 Book Errata 链接。在这个页面上可以查看到 Wrox 编辑已提交的所有勘误项。完整的图书列表还包括每本书的勘误表，网址是 http://www.wrox.com/misc-pages/booklist.shtml。

05 P2P.WROX.COM

要与作者和同行讨论，请加入 p2p.wrox.com 上的 P2P 论坛。这个论坛是一个基于 Web 的系统，便于您发布与 Wrox 图书相关的消息和相关技术，以及与其他读者和技术用户交流心得。该论坛提供了订阅功能，当论坛上有新的消息时，它可以给您发送感兴趣的主题。Wrox 作者、编辑和其他业界专家和读者都会到这个论坛上来探讨问题。

在 http://p2p.wrox.com 上，有许多不同的论坛，它们不仅有助于阅读本书，还有助于开发自己的应用程序。要加入论坛，可以遵循下面的步骤：

(1) 进入 p2p.wrox.com，单击 Register 链接。

(2) 阅读使用协议，并单击 Agree 按钮。

(3) 填写加入该论坛所需要的信息和自己希望提供的其他信息，单击 Submit 按钮。

(4) 您会收到一封电子邮件，其中的信息描述了如何验证账户，完成加入过程。

注释：

不加入 P2P 也可以阅读论坛上的消息，但要发布自己的消息，就必须加入该论坛。

加入论坛后，就可以发布新消息和回复其他用户发布的消息。可以随时在 Web 上阅读消息。如果要让该网站给自己发送特定论坛中的消息，可以单击论坛列表中该论坛名旁边的 Subscribe to this Forum 图标。

关于使用 Wrox P2P 的更多信息，可阅读 P2P FAQ，了解论坛软件的工作情况以及 P2P 和 Wrox 图书的许多常见问题。要阅读 FAQ，可以在任意 P2P 页面上单击 FAQ 链接。

目　录

第Ⅰ部分　概　述

第Ⅱ部分　OOIS UML 概 述

第 I 部分

概　　述