



SPRINGER PROFESSIONAL COMPUTING

Guide to  
Applying the UML

# UML 高级应用

(德) Sinan Si Alhir 著  
韩宏志 译



清华大学出版社

# UML 高级应用

(德) Sinan Si Alhir 著

韩宏志 译

清华大学出版社

北 京

## 内 容 简 介

本书详细介绍了 UML 面向对象软件设计的基本概念、符号表示、术语、准则和原理。围绕 UML 的结构分析建模原理,并通过对路标的分析来阐述 UML 的应用方法。前三章简要介绍 UML、建模和面向对象的基本原理,第 4 章介绍路标,第 5 章到第 9 章阐述各种 UML 建模技术及其在路标中的应用,最后两章介绍 UML 扩展机制和对象约束语言。

本书可用作软件工程相关专业的教科书,也可供 UML 软件项目的设计、开发和管理人员参考使用。

Sinan Si Alhir: Guide to Applying the UML

EISBN: 0-387-95209-8

Copyright© 2002 by Springer Press Ltd.

Authorized translation from the English language edition published by Springer Press Ltd.

All rights reserved. For sale in the People's Republic of China only.

Chinese simplified language edition published by Tsinghua University Press.

本书中文简体字版由施普林格出版公司授权清华大学出版社出版。未经出版者书面许可,不得以任何方式复制或抄袭本书内容。

版权所有,翻印必究。

本书封面贴有清华大学出版社激光防伪标签,无标签者不得销售。

北京市版权局著作权合同登记号 图字:01-2003-4521

图书在版编目(CIP)数据

UML 高级应用/(德)阿尔赫(Alhir, S. S.)著;韩宏志译. —北京:清华大学出版社,2003.12

书名原文:Guide to Applying the UML

ISBN 7-302-07996-X

I. U… II. ①阿… ②韩… 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(2004)第 004263 号

出 版 者:清华大学出版社 地 址:北京清华大学学研大厦

<http://www.tup.com.cn> 邮 编:100084

社 总 机:010-62770175 客户服务:010-62776969

组稿编辑:曹 康

文稿编辑:王晓娜

封面设计:康 博

版式设计:康 博

印 刷 者:北京市昌平环球印刷厂

装 订 者:三河市新茂装订有限公司

发 行 者:新华书店总店北京发行所

开 本:185×260 印张:18.75 字数:389 千字

版 次:2004 年 2 月第 1 版 2004 年 2 月第 1 次印刷

书 号:ISBN 7-302-07996-X/TP·5795

印 数:1~4000

定 价:36.00 元

---

本书如存在文字不清、漏印以及缺页、倒页、脱页等印装质量问题,请与清华大学出版社出版部联系调换。联系电话:(010)62770175-3103 或(010)62795704

# 前 言

本书将全面讨论 UML 语言，详尽讲述路标，带您进入 UML 之门。

系统工程技术领域受到现实的约束，其中时间和空间维度为更改和复杂性搭建起共同活动的舞台。此处，必须审视一个问题：“要最大程度地拓展系统工程技术领域，应该用哪种语言呢？”正确的回答是“UML。”即统一建模语言(Unified Modeling Language)。通过 UML，人员与团队、过程、工具和技术可共同营造一个向同一目标迈进的环境，可以在减少工时、降低成本的同时，完善功能，提高质量，创造更多价值。

还有一个问题亟需解决：“如何才能有效应用 UML？”正确的回答是“路标。”路标专门从 UML 派生而来，独立于工具、过程和技术，为 UML 的有效应用提供了一种切实可行的方法。

## UML 概述

UML 是一种发展演变而来的通用建模语言，运用广泛，得到很多工具支持，并被用作行业标准；用于指定、显现、构建和记录系统密集过程的工件。

UML 最早主要源于 Rational Software 公司和三位盟友——最卓越的方法学家 Grady Booch、Jim Rumbaugh 和 Ivar Jacobson 的构思，并在他们的共同努力下逐渐发展起来。UML 是作为对象管理组(Object Management Group, OMG)和 Rational Software 公司的一个标准出现的，将信息系统和技术行业的最佳工程实践合并为建模技术集合。

UML 可用于不同类型的系统(软件和非软件)、域(业务与软件)和方法(或过程)。它支持并推广(但并不要求或强制)一种过程，这种过程的特点是用例驱动的、以体系结构为中心的、面向对象的、基于组件的、迭代的、递增的和抗风险的。但 UML 并未指示任何特定的系统开发方法，它非常灵活，可进行定制，以适应所有情况。

随着 UML 的出现，并鉴于其支持各种类型的过程或方法学的灵活性，各种协调使用 UML 的方法应运而生，其中包括：不使用方法学、使用重量级方法学或轻量级方法学。虽然这些方法都在一定程度上取得了成功，但是，要成功有效地应用 UML，并利用它所提供的一切功能性，需要在路标中捕获真正需要的信息类型。

## 学习目标

重量级方法学涉及大量规则和准则,例如三位盟友的统一过程(Unified Process, UP)架构。轻量级方法学也称“灵活”、“经验化”或“突现”的方法学,几乎不涉及规则和准则,例如 Kent Beck、Ward Cunningham 和 Ron Jeffries 合作开发的 eXtreme Programming(XP, 极限编程)准则。本书介绍如何整体性并结合性地组合 UML 的所有元素,并使用路标来填补 UML 和过程之间的空隙;路标为重量级方法学提供了一个成本识别方案,为轻量级方法学提供了一个成熟度识别方案。

本书未指定任何特定过程,其重点在于分析路标,说明在应用 UML 时应解决什么问题,其中包括判定点(计划临界点,必须在此临界点做出判定)及判定点之间的关系。方法学专用于解决各种过程元素,包括什么人对什么产品从事什么活动,何时、何地、为什么以及按什么方式完成这些活动。路标解决判定点及其相互关系,为方法学提供架构。为保证公司成功地应用 UML,可根据需要将路标伸缩为轻量级或重量级方法学。可将任何特定过程或方法学理解成遍历路标。

本书介绍语法规则、组成原理、风格指南和实用示例,介绍独立于工具、过程和技术并有助于有效应用 UML 的路标。它围绕 UML 结构分析组成 UML 模型的基本原理,围绕路标分析应用 UML 的基本方法。

本书论述精辟,观点新颖,介绍了 UML 的重要知识,既可用作教材,也可用作参考书。通过学习,您将了解如何在不过早应用特定过程的情况下,将 UML 应用于实际项目。

## 读者对象

本书适于软件工程及相关专业学生、执行项目的项目经理、从业者(工程师),以及任何希望有效应用 UML 的人员学习使用。

为了便于负责管理项目执行的经理和专家学习使用,本书全面介绍了 UML 的用法,使经理能够更加方便地制定项目执行的相关决策;可将路标用作公司实际过程的更高级别抽象,以进一步理解执行项目的各个选项;可将路标用作架构,这样,既可成功有效地应用 UML,又不致过早地强制使用过程。

为了便于负责执行项目各个相关活动的从业者(工程师)和专家学习使用,本书提供了应用 UML 的丰富示例,以便工程师在整个周期利用 UML 建模时,能够更加灵活地执行决策;认真分析本书介绍的语法规则、组成原理、风格指南和实用示例,进一步理解概念、说明和实现模型的创建方法。

为了便于经理和从业者学习,本书通过路标说明了 UML 元素之间的相关方式,还

说明了路标如何支持跟踪性和伸缩性以支持系统开发周期过程。

通过本书，可以学习(初级水平)和有效应用(中级到高级水平)UML。本书不要求您具备预备知识和技能，不过，如果了解面向对象概念，对学习更有益。

本书将带您感受路标，发掘其益处，开发其价值。不过，前进的道路上并非一帆风顺，会时常遇到各种问题，需深思熟虑，冷静分析，从多个方面进行认真思考，仔细剖析应该做什么、如何做以及为什么做，以便逐渐吸收利用路标的多种好处。在此过程中，应统观全局，不只是强调重要目标和达到目标的速度，而应在焦点、目的、历程、目标和达到目标的速度之间寻求一个平衡点。

## 方法

过程的重量级和轻量级程度如何确定？这个问题可通过解决更基本的问题来解答：过程或方法学有什么价值？独立于过程的重量级或轻量级程度如何？为什么说某种过程或方法学比另一种更合适？有许多策略试图协调或辩论重量级方法和轻量级方法，但这失去了解决过程或方法学价值的根本问题的机会，并常引发核心过程的文化大战。这些派系均低估了语言的巨大影响，实际上，语言建立了思想和行为的边界，过程建立了该边界中的行为，工具建立了该边界中的自动操作。路标建立了一个连续区间，从中可导出和应用最佳方法(独立于重量级或轻量级)。该方法防止类似于重量级方法的工程化过程元素的过度使用，也防止类似于轻量级方法的工程化过程元素不足，它提供了一种切实可行的路标，以成功有效地利用 UML。

由于历史原因，我一直从事咨询工作。那时，如果一个人未能获得两个或更多哲学博士基本学位证书，及一个证明三十年及以上行业经历的实用证书，是不能称为咨询师的。

在为客户服务之前，我经常问：“咨询师的价值体现在哪些方面？”“是他们提供给项目的知识吗？”或许有人认为是，但实际上，咨询师的知识也是有限度的，这是一种极端认识。我又问：“是他们可以在项目中完成的工作吗？”又有人认为是，但实际上，咨询师并非是一个实用的“承包人”，这是另一种极端认识。问题的答案可能介于这两个极端之间，咨询师的价值可以归纳为：他们提出的问题(人们在这些问题的指引下寻求答案，并提出更多问题)，根据知识和经验提供的可选方案，为寻求可选方案而完成的工作，以及他们提供的辅导。在编写本书时，也遵循这样一种思维方式，并未列出讲述内容的所有答案，而是提供了一个架构：提出问题和回答问题，识别、计算和选择方案，分析特定方案的分支。

客户常问：“不同公司和团队使用的方法学不尽相同，在咨询时，您如何保持提供价值的一致性和可认知性？”我的回答是“路标。”路标并不是专门设计的、深奥的、非常抽象的东西，相反，它非常实用，便于应用，受时间限制，是有价资产，使从业

者可以用任何过程、工具和技术成功有效地应用 UML。在与各种从业者团体打交道时(它们有各自独特的文化,涉及过程、工具和技术),我一直使用 Socratic 方法将路标动态地传递给从业者。Socratic 方法是一个论证(转换或论述)方法,它不是简单地指出正确结论,而是由教师逐渐向学生提问,从而引导学生认知和得出结论。在从业者确认已理解结论之后,便将它们作为一个团体,授予协调使用路标的权力,以采用过程和方法学,并发展已采用的过程和方法学,通常,还将合并、优化和简化方法,以成功有效地达到目标和设想。此外,只要有对应的概念,也可使用除 UML 之外的其他任何建模语言。

有些客户对此持怀疑态度,特别是在阅读我的作品之后,便将我当成一个书呆子,常问:“您的方法实用吗?”对此,我的回答是“非常实用。”一开始,这些客户不大相信,不过,待到与我共同经历开发过程之后,他们终于心悦诚服。有人可能认为路标过于理想化,一点也不实用;但实际上,路标使我们注重实践,并将其真正应用于实际项目,如果缺少它,实践将成为附带事件。如总想着采用一些权宜之计或容易的方法,结果必然会差强人意。实用的并不代表容易的,理想的并不代表困难的。正确的理解是,容易的方法可能只能执行实用的或理想方案;而最困难的和最有意义的是在特定上下文(而不仅在真空中)中执行实际的方案,换言之,即实际应用理想方法,并注重实际方法和理想方法之间的力量平衡。许多人错将理想方法和实用方法割裂开来,但事实上,成功的做法是将实际上下文的理想方法实际应用于实际情形,这是我很久以前就接受的一个教训,在职业和个人生涯中,这个概念在我的头脑中生根发芽,逐渐成熟。许多人在面对困难和挑战时,选择牺牲实际方法之上的理想方法,而不是继续钻研并从实际方法向理想方法发展,由上述分析可知,这种做法也是错误的。此处的论述都是关于路标的实用信息,请认真体会理解!许多人与我共事后,往往有所感触,认为我不仅联系上下文含义,还考虑环境含义,但是在各种情况下(即使在同一上下文中),策略都是惟一的,所以我的着眼点既不面向过程,也不面向路标,而是从策略上面向结果。

值得注意的是, UML 不只是一个标准或另一建模语言。它是一种“范例”、“基本原理”、“变革”和“进化”,说明如何处理问题求解和系统问题。我们常说,英语是全世界的“通用语言”,同样,也可以肯定地说, UML 将成为信息系统和技术领域的“通用语言”。

本书不含源代码,重在分析建模语言和路标;它并没有说明如何将系统转换为特定实现,而是说明如何成功有效地应用 UML。

由于时间仓促,水平有限,书中难免有疏漏和错误之处,恳请得到社会各界和广大读者的批评和指正。本书介绍路标的精华部分,而不只是实质部分(实质是指必需和基本的内容,而精华是指精髓和典范),以成功有效地使用 UML;虽然 UML 在不断发展,但本书仍极具价值。在阅读本书的过程中,如果您有什么意见,或要对路标、UML、

面向对象及其他相关主题发表看法，欢迎来信告知，我的邮箱地址是 [salhir@earthlink.net](mailto:salhir@earthlink.net)；另外，您也可浏览我的 Web 站点主页，网址为：<http://home.earthlink.net/~salhir>。

## 内容介绍

本书共分 11 章。在后续章节中，个别地方会重述前面章节的内容，并对这些内容进行进一步阐释。这样，在实际执行项目的过程中，如果用到本书内容，您可以直接参考特定章节来获取所有相关信息，不必再查阅其他章节。

前 3 章重点介绍 UML、建模和面向对象。

- 第 1 章“UML 简介”首先介绍 UML 的目标、范围和发展史；然后讲述 UML 与过程的关系；最后讨论如何才能成功、高效地使用 UML。通过第 1 章的学习，您可以理解语言和要求。
- 第 2 章“建模”介绍建模语言、系统以及支持生成更好的系统模型的各种机制，并介绍建模与过程和方法学的相互关系。通过第 2 章的学习，您将理解路标是如何解决轻量级过程和重量级过程之间的争夺战的。
- 第 3 章“面向对象”介绍面向对象的基本原理、支持建立模型系统的基本概念，以及如何通过 UML 传递这些概念。通过第 3 章的学习，您可以了解 UML 表示法的详细指南概要。

第 4 章简要介绍路标，第 5~9 章详细介绍各种 UML 建模技术，以及这些技术在路标中的用法。第 4 章是第 5~9 章的出发点。第 5~9 章的各章都分为两个部分：第 1 部分通过实例，进一步阐述第 3 章介绍的各种 UML 建模技术，并介绍各种基本规则、原理和方式指导；第 2 部分讨论如何在第 4 章介绍的路标上下文中使用各种 UML 建模技术。基本规则解决关于 UML 建模元素“是什么”的问题；基本原理解决关于 UML 模型“是什么”的问题；基本风格指南解决关于 UML 建模元素“如何做”的问题。如要概括了解路标，请阅读第 4 章；如要进一步理解 UML 表示法和路标，请阅读第 5~9 章。这些章节并未列出所有答案，而是提供了一个架构：提出问题和回答问题，识别、计算和选择方案，分析特定方案的分支。对于 UML 初学者而言，如要进一步了解关于第 3 章介绍的各种 UML 建模技术，可以首先阅读第 5~9 章中关于建模技术的部分，然后再阅读第 4 章，最后再返回到第 5~9 章，着重理解在路标上下文中各种 UML 建模技术的用法。

- 第 4 章“路标”建立在第 2 章和第 3 章的基础之上，简要介绍路标。要派生路标需分析 UML 句子，建立路标空间，定义常规路标，细化明细路标，指定表示法路标。第 4 章是继续学习第 5~9 章内容的预备知识。



- 第 5 章“用例(用户)建模”着重分析用例图(见附录)并建立系统功能维度模型,即系统为用户提供了哪些功能性(在概念模型中捕获)。
  - 第 6 章“结构(静态)建模”着重分析类图和对象图(见附录)并建立系统结构或静态维度模型,即哪些元素及其关系构成系统(在说明模型中捕获)。
  - 第 7 章“行为(动态)建模”着重分析顺序图、协作图、状态图和活动图(见附录),并建立系统行为或动态维度模型,即协作构成系统的元素如何交互,从而为用户提供系统的功能性(在说明模型中捕获)。
  - 第 8 章“组件(实现)建模”着重分析组件图(见附录)并建立系统实现维度模型,即系统的实现方式(在实现模型中捕获)。
  - 第 9 章“部署(环境)建模”着重分析部署图(见附录)并建立系统环境维度模型,即实现或实现的系统在环境中的驻留方式(在实现模型中捕获)。
- 最后两章重点介绍 UML 扩展机制,以及 UML 的约束语言——对象约束语言(OCL)。
- 第 10 章“扩展机制”简要介绍 UML 的扩展机制。通过本章的学习,您将初步了解 UML 的扩展方法。
  - 第 11 章“对象约束语言”简要介绍 OCL。通过本章的学习,您将初步了解 OCL 的用法。

## 致谢

本书的最终出版凝聚着许多人的付出和努力。特别是在我身边与我共度这段艰难时光的人,他们理解我,鼓励我,耐心地帮助我,为了本书的问世做出了很大的奉献。

感谢父亲 Saad 和母亲 Rabab,是他们把我培养成人,引导我走上成功之路。感谢弟弟 Ghazwan 和 Phillip,是他们的孜孜相助成就了我的事业。感谢妻子 Milad 在生活上对我的关怀。感谢女儿 Nora,她一直促使我学习“诚实”,并重检和审视已经学到的“诚实”,教我自惭,催我自新,使我学会了适应和发展。

感谢顾问 Carl V. Page 博士和 George C. Stockman 博士,是他们的谆谆教导,使我体会到:计算机科学的理想、理论和抽象领域与计算机职业的实际、实用和具体领域是密不可分的,它们相互依存,相互促进,为复杂的现实问题的解决方案奠定了基础。

感谢世界各地的从业者阅读我的第一本书籍《UML in Nutshell》(O'Reilly&Association, 1998 年),衷心地感谢他们对该书的支持和厚爱。

感谢负责本书出版的 Springer-Verlag 公司的员工;同时感谢评论家们为本书提出的宝贵意见。

我会永远记得你们,也希望你们记住我。

Sinan Si Alhir

# 目 录

|       |                      |    |
|-------|----------------------|----|
| 第 1 章 | UML 简介 .....         | 1  |
| 1.1   | UML 的定义 .....        | 1  |
| 1.2   | UML 与过程或方法学的关系 ..... | 2  |
| 1.3   | UML 的发展史 .....       | 4  |
| 1.4   | 有效使用 UML .....       | 7  |
| 第 2 章 | 建模 .....             | 9  |
| 2.1   | 语言 .....             | 9  |
| 2.1.1 | 字母 .....             | 9  |
| 2.1.2 | 单词 .....             | 9  |
| 2.1.3 | 句子 .....             | 10 |
| 2.1.4 | 段落 .....             | 11 |
| 2.1.5 | 节 .....              | 13 |
| 2.1.6 | 文档 .....             | 14 |
| 2.1.7 | 其他元素 .....           | 14 |
| 2.2   | 系统和上下文 .....         | 17 |
| 2.2.1 | 域或空间 .....           | 17 |
| 2.2.2 | 系统 .....             | 17 |
| 2.2.3 | 体系结构 .....           | 18 |
| 2.2.4 | 模型 .....             | 19 |
| 2.2.5 | 体系结构视图 .....         | 22 |
| 2.2.6 | 图表 .....             | 23 |
| 2.3   | 建模机制 .....           | 24 |
| 2.3.1 | 观点 .....             | 25 |
| 2.3.2 | 抽象级别 .....           | 26 |
| 2.3.3 | 二分法 .....            | 27 |
| 2.3.4 | 扩展机制 .....           | 28 |
| 2.4   | 过程和方法学 .....         | 28 |
| 2.4.1 | 开发循环和开发阶段 .....      | 29 |
| 2.4.2 | 迭代循环和迭代阶段 .....      | 30 |

|              |                   |           |
|--------------|-------------------|-----------|
| 2.4.3        | 迭代阶段明细 .....      | 32        |
| 2.4.4        | 试探法 .....         | 33        |
| 2.5          | 过程和方法学的价值 .....   | 36        |
| 2.5.1        | 问题 .....          | 37        |
| 2.5.2        | 烹饪和系统开发 .....     | 37        |
| 2.5.3        | 路标 .....          | 39        |
| 2.5.4        | 答案 .....          | 41        |
| <b>第 3 章</b> | <b>面向对象 .....</b> | <b>43</b> |
| 3.1          | 面向对象原理 .....      | 43        |
| 3.1.1        | 抽象 .....          | 43        |
| 3.1.2        | 封装 .....          | 44        |
| 3.1.3        | 泛化 .....          | 45        |
| 3.1.4        | 多态 .....          | 46        |
| 3.2          | 结构(静态)概念 .....    | 46        |
| 3.2.1        | 类图和对象图 .....      | 46        |
| 3.2.2        | 用例图 .....         | 60        |
| 3.2.3        | 组件图 .....         | 63        |
| 3.2.4        | 部署图 .....         | 64        |
| 3.3          | 行为(动态)概念 .....    | 66        |
| 3.3.1        | 顺序图和协作图 .....     | 66        |
| 3.3.2        | 状态图 .....         | 75        |
| 3.3.3        | 活动图 .....         | 76        |
| 3.4          | 面向对象系统 .....      | 77        |
| 3.4.1        | 包 .....           | 78        |
| 3.4.2        | 模板 .....          | 80        |
| 3.4.3        | 模式和架构 .....       | 80        |
| 3.4.4        | 系统 .....          | 82        |
| <b>第 4 章</b> | <b>路标 .....</b>   | <b>87</b> |
| 4.1          | UML 句子 .....      | 87        |
| 4.1.1        | 协作和交互系统 .....     | 87        |
| 4.1.2        | 服务 .....          | 88        |
| 4.1.3        | 服务实现 .....        | 89        |
| 4.1.4        | UML 句子 .....      | 91        |
| 4.2          | 路标空间 .....        | 91        |

---

|       |            |     |
|-------|------------|-----|
| 4.2.1 | 观点和抽象级别    | 92  |
| 4.2.2 | 笛卡尔积       | 92  |
| 4.2.3 | 路标空间       | 94  |
| 4.3   | 常规路标       | 96  |
| 4.3.1 | 观点和抽象级别    | 96  |
| 4.3.2 | 过程规范       | 97  |
| 4.3.3 | 常规路标       | 98  |
| 4.4   | 明细路标和表示法路标 | 99  |
| 4.4.1 | 概念元素       | 99  |
| 4.4.2 | 机制         | 101 |
| 4.4.3 | 路标         | 103 |
| 4.4.4 | 路标示例       | 109 |
| 4.5   | 应用路标       | 129 |
| 4.5.1 | 重量级和轻量级方法  | 130 |
| 4.5.2 | 试探法        | 130 |
| 第 5 章 | 用例(用户)建模   | 132 |
| 5.1   | 用例图        | 132 |
| 5.1.1 | 参与者        | 132 |
| 5.1.2 | 用例         | 134 |
| 5.1.3 | 参与者关系      | 137 |
| 5.1.4 | 用例关系       | 139 |
| 5.2   | 应用用例图      | 146 |
| 5.2.1 | 结构         | 147 |
| 5.2.2 | 需求         | 148 |
| 5.2.3 | 合并         | 150 |
| 第 6 章 | 结构(静态)建模   | 151 |
| 6.1   | 类图         | 151 |
| 6.1.1 | 分类器        | 151 |
| 6.1.2 | 关系         | 165 |
| 6.2   | 对象图        | 175 |
| 6.2.1 | 分类器实例      | 175 |
| 6.2.2 | 关系实例       | 178 |
| 6.3   | 应用类和对象图    | 179 |
| 6.3.1 | 概念元素       | 180 |

---

|              |                 |            |
|--------------|-----------------|------------|
| 6.3.2        | 机制              | 180        |
| 6.3.3        | 结构              | 181        |
| 6.3.4        | 分析              | 183        |
| 6.3.5        | 设计              | 186        |
| 6.3.6        | 验证              | 192        |
| 6.3.7        | 合并              | 196        |
| <b>第 7 章</b> | <b>行为(动态)建模</b> | <b>197</b> |
| 7.1          | 顺序图             | 197        |
| 7.1.1        | 分类器角色           | 198        |
| 7.1.2        | 交互              | 199        |
| 7.1.3        | 生命线             | 203        |
| 7.1.4        | 激活              | 204        |
| 7.1.5        | 消息和激励           | 204        |
| 7.2          | 协作图             | 206        |
| 7.2.1        | 关联角色            | 207        |
| 7.2.2        | 协作              | 208        |
| 7.2.3        | 消息和激励           | 211        |
| 7.2.4        | 行为组织            | 214        |
| 7.3          | 状态图             | 218        |
| 7.3.1        | 状态              | 218        |
| 7.3.2        | 转换              | 220        |
| 7.3.3        | 子机              | 224        |
| 7.4          | 活动图             | 224        |
| 7.4.1        | 动作状态            | 225        |
| 7.4.2        | 泳道              | 226        |
| 7.4.3        | 流               | 226        |
| 7.5          | 应用顺序、协作、状态和活动图  | 228        |
| 7.5.1        | 顺序图             | 228        |
| 7.5.2        | 协作图             | 229        |
| 7.5.3        | 状态图             | 230        |
| 7.5.4        | 活动图             | 230        |
| <b>第 8 章</b> | <b>组件(实现)建模</b> | <b>232</b> |
| 8.1          | 组件图             | 232        |
| 8.1.1        | 工件              | 232        |

---

|                       |            |
|-----------------------|------------|
| 8.1.2 组件              | 236        |
| 8.1.3 组件关系            | 236        |
| 8.2 应用组件图             | 237        |
| 8.2.1 结构              | 237        |
| 8.2.2 实现              | 238        |
| 8.2.3 合并              | 239        |
| <b>第 9 章 部署(环境)建模</b> | <b>240</b> |
| 9.1 部署图               | 240        |
| 9.1.1 节点              | 240        |
| 9.1.2 节点关系            | 242        |
| 9.2 应用部署图             | 243        |
| 9.2.1 结构              | 243        |
| 9.2.2 部署              | 244        |
| 9.2.3 合并              | 245        |
| <b>第 10 章 扩展机制</b>    | <b>246</b> |
| 10.1 UML 的体系结构        | 246        |
| 10.1.1 四层元建模体系结构      | 246        |
| 10.1.2 UML 元模型        | 249        |
| 10.2 定型               | 251        |
| 10.2.1 声明             | 251        |
| 10.2.2 应用             | 254        |
| 10.3 属性               | 257        |
| 10.3.1 约束             | 258        |
| 10.3.2 标记定义和标记值       | 258        |
| 10.4 配置文件             | 260        |
| <b>第 11 章 对象约束语言</b>  | <b>262</b> |
| 11.1 对象约束语言的定义        | 262        |
| 11.2 表达式              | 262        |
| 11.2.1 不变量            | 263        |
| 11.2.2 前置条件和后置条件      | 265        |
| 11.2.3 包语句            | 266        |
| 11.2.4 let 表达式和定义约束   | 266        |
| 11.3 属性               | 266        |
| 11.3.1 分类器和实例         | 268        |

---

|        |              |     |
|--------|--------------|-----|
| 11.3.2 | 关联和链接        | 269 |
| 11.3.3 | 分类器范围和实例范围属性 | 270 |
| 11.4   | 标准 OCL 类型    | 270 |
| 11.4.1 | 基本类型         | 271 |
| 11.4.2 | 集合类型         | 274 |
| 11.5   | 标准 OCL 包     | 280 |
| 附录     |              | 282 |

# 第1章 UML 简介

本章将概述统一建模语言(Unified Modeling Language, UML), 分析使用 UML 和过程的必要性。首先介绍 UML 的目标及范围; 然后讲述 UML 与过程和方法学的关系以及 UML 的发展史; 最后讨论如何才能成功、高效地使用 UML。

## 1.1 UML 的定义

在系统开发期间, 会遇到各种复杂的、不断变化的问题。若系统支持的业务过程和需求跨越使用不同技术的多个单位和区域, 情况将更为复杂。为应对之, 我们尝试采用包含相应技术的解决方案, 并力求在减少工时、降低成本的同时完善功能并提高质量。但事实上, 试图解决的复杂性愈高, 更改的内容也愈多; 反过来, 增加更改内容后, 又会滋生新的复杂性, 形成恶性循环; 问题和方案不断演变, 最终导致混乱。另外, 由于复杂性高, 又时常更改, 所以系统开发无法由个人完成, 必须通过团队合作, 由人员与团队、过程、工具和技术共同营造一个向同一目标迈进的氛围。为成功地组建团队, 需使用一种公共语言来沟通信息, 协调问题和方案, 并采用统一方法, 在管理复杂性和更改的同时实现开发目标; 否则, 在面对问题时将无从下手。

应如何处理呢? 答案是使用 UML。通过它, 一切问题均可迎刃而解。

UML(统一建模语言)用于指定、显现、构建和记录系统密集(system-intensive)的过程的工件(artifact)。即为“语言”, 当然是一种交流手段, 使我们可以沟通主题、正在讨论的系统及周边环境或上下文。“建模”是指 UML 通过对主题及其上下文的模型模式化而突出主题; 模型捕获与主题相关的一系列观点或抽象(即吸纳理解主题所需的本质信息, 摒除所有可能对影响理解的无关或偶然信息); 通过建模, 可进一步管理系统开发的复杂性。“统一”是指 UML 源于对象管理组(Object Management Group)和 Rational Software 公司, 结合信息系统和技术行业的最佳工程实践, 实现行业一体化, 使我们可以协调使用那些经实践检验、证明切实可行的技术。

“指定系统”是指可通过 UML 来交流系统要做什么, 以及系统如何满足这些要求。

“显现系统”是指可通过 UML 进行可视化交流, 此所谓“百闻不如一见”, 使我们透过想象查看系统, 检测模式。“构建系统”是指可用 UML 来指导物理系统的构建, 类似于设计蓝图。“记录系统”是指可利用 UML 来记录整个系统开发周期过程的信息, 免于事后为追加记录而苦思冥想。UML 的使用方便了指定、显现、理解和记录问题,



方便了获取、交流和协调使用解决问题的方法，也方便了指定、显现、构建和记录解决方案。

UML 是一种演变而来的通用建模语言，运用广泛，并能得到工具支持，被用作行业标准。UML 最早主要源于 Rational Software 公司和三位盟友——最卓越的方法学家 Grady Booch、Jim Rumbaugh 和 Ivar Jacobson 的构思，并凝聚了面向对象界核心建模概念的共识。UML 可用于整个系统开发周期过程，可用于不同类型的系统(软件或非软件)、域(业务与软件)及方法或过程，可由各个供应商和工具支持。UML 既非专用的语言，也非闭合的语言，而是一种开放的、可充分扩展的行业认知语言，得到了构成 UML 合作者联盟和 OMG 的多个组织的支持。

设计 UML 时，力求使其成为一种简单明了的可视建模语言，较自然语言更简单精确，具有可扩展性，独立于实现和过程。若过于复杂，将令人敬而远之，最终无法被广泛接纳和使用。若不精确，将背离支持通信的初衷。若不能扩展，是闭合的，则只能解决已知需求，无法解决尚未发现的需求。若依赖于实现或过程，其使用范围将严重受限，难以达到通用和广泛运用的目标。

UML 促成了前述三位方法学家的概念在建模语言级别上的融合：Grady Booch 的 93 方法，Jim Rumbaugh 的对象建模技术(Object Modeling Technique, OMT)以及 Ivar Jacobson 的面向对象软件工程(Object-Oriented Software Engineering, OOSE)，并凝聚了面向对象界核心建模概念的共识。需澄清一点，UML 既不是一种可视化编程语言、工具或存储库规范，也不是一种方法或过程。

## 1.2 UML 与过程或方法学的关系

UML 采用并推广(但并不要求或强制)了一种过程，这种过程的特点是用例驱动的(use-case-driven)、以体系结构为中心的、面向对象的、基于组件的、迭代的、递增的和抗风险(risk-confronting)的。但 UML 并未指示任何特定的系统开发方法，它非常灵活，可进行定制，以适应所有情况。

迭代和递增过程采用渐进方法开发系统。各个步骤(或迭代)都由若干个更小的步骤组成，负责完成全部系统的部分工作(可能采用并行方式)；而各个递增则随时间的推移开发系统的其他部分。迭代并无特别之处：由构成方法的活动或步骤的偏序(semi-order)集合组成，有始有终，按规划设计，有评估标准，需要资源支持，旨在生成一些示范以提供能指导未来迭代的反馈。反馈使迭代经验化。通过使用迭代和递增方法，可以在验证需求并在系统演变的各个步骤中协调使用反馈时，我们可以，随着系统的扩展更好地管理复杂性，更好地结合需求和技术革新，发现和揭示未知需求。在迭代方法中，需要进行一些重复工作，以逐渐发展和改进系统，但这不能作为迭代的主要任务，