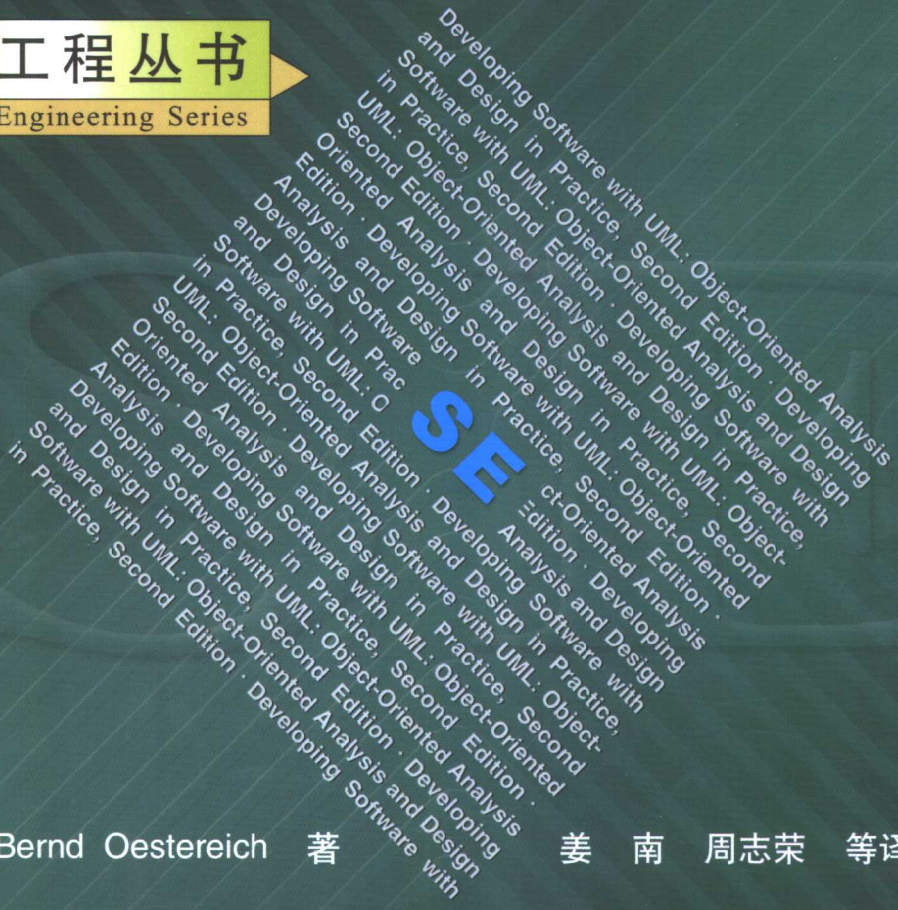
软件工程丛书
Software Engineering Series



[德] Bernd Oestereich 著　　姜南 周志荣 等译

软件开发方式

——UML面向对象分析与设计

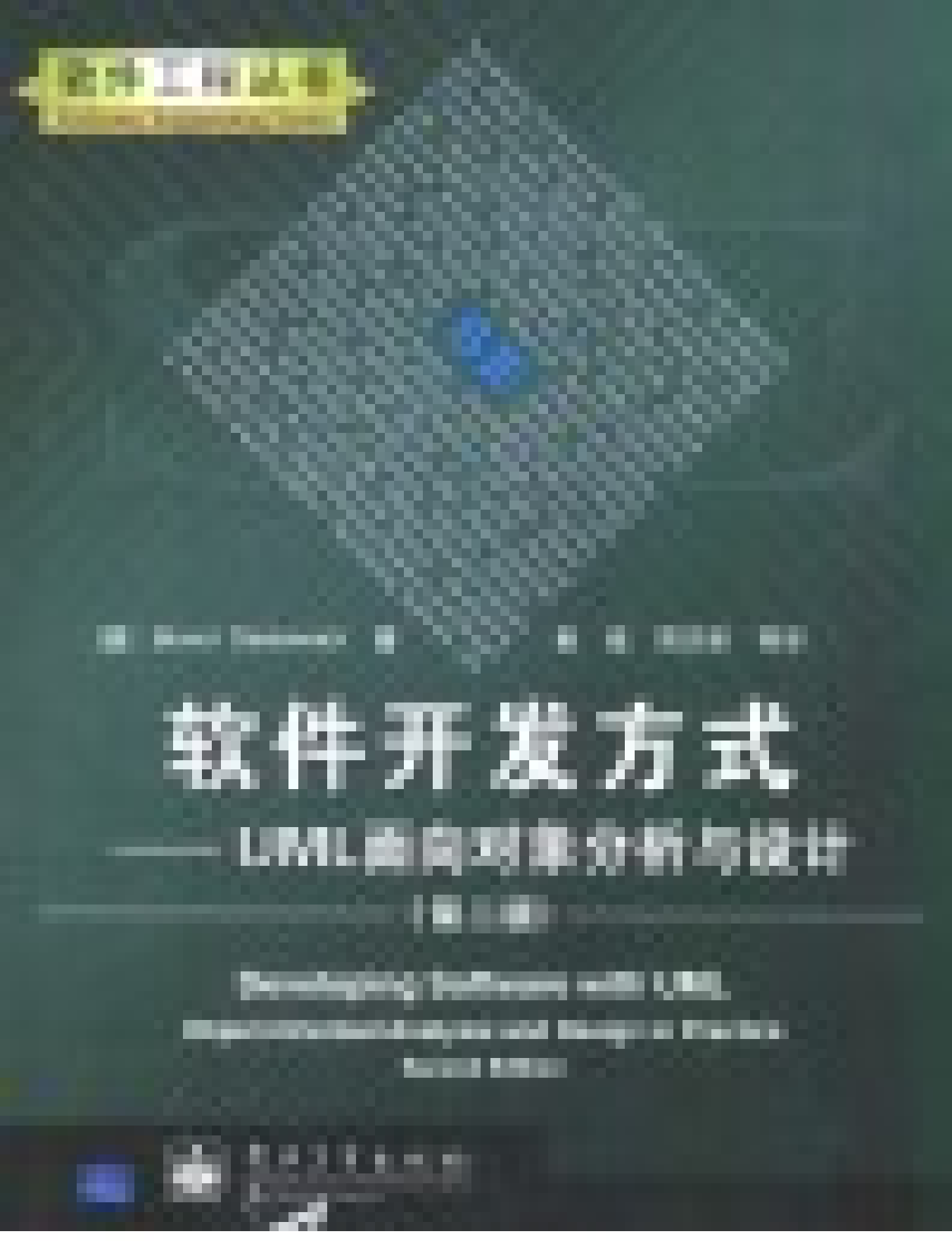
（第二版）

Developing Software with UML

Object-Oriented Analysis and Design in Practice

Second Edition

PEARSON Addison Wesley
PHEI
電子工業出版社
Publishing House of Electronics Industry
http://www.phei.com.cn



软件工程丛书

软件开发方式

——UML面向对象分析与设计

（第二版）

Developing Software with UML

Object-Oriented Analysis and Design in Practice

Second Edition

[德] Bernd Oestereich 著

姜 南 周志荣 等译

電子工業出版社

Publishing House of Electronics Industry

北京 · BEIJING

内容简介

本书解释了使用面向对象的方法进行软件开发的优点，同时从专业角度对这种技术进行了描述。本书采用了许多现实生活中的例子对面向对象分析与设计中的统一建模语言进行了系统的讲解。虽然只用较小的篇幅简单描述了UML（统一建模语言），但却包含了它的所有重要内容，其符号和语义是面向对象建模过程中所用的最新标准。为了更容易地切入主题，本书中不包含UML元模型的内容。本书所讨论的重点内容，即用例驱动的、以体系结构为中心的渐进式开发方法主要应用于嵌入式公司信息系统的开发中，但也适用于其他技术和应用领域。

本书概述了面向对象的结构，对一个软件项目进行建模的过程，以及组成统一建模语言（UML）的图和模型元素。本书是供具有很少的OO软件开发经验的开发人员学习面向对象的分析与设计的入门读本。

版权贸易合同登记号　图字：01-2003-2049

图书在版编目（CIP）数据

软件开发方式——UML面向对象分析与设计，第二版/（德）奥斯特海希（Oestereich, B.）著；姜南等译.
-北京：电子工业出版社，2004.11
（软件工程丛书）
书名原文：Developing Software with UML, Object-Oriented Analysis and Design in Practice, Second Edition
ISBN 7-121-00452-6

Ⅰ.软… Ⅱ.①奥… ②姜… Ⅲ.面向对象语言，UML-程序设计 Ⅳ.TP312

中国版本图书馆CIP数据核字（2004）第104807号

责任编辑：赵红燕　　特约编辑：詹文军
印　　刷：北京兴华印刷厂
出版发行：电子工业出版社
　　北京市海淀区万寿路173信箱　邮编：100036
经　　销：各地新华书店
开　　本：787 × 980　1/16　　印张：14.5　　字数：335千字
印　　次：2004年11月第1次印刷
定　　价：25.00元

凡购买电子工业出版社的图书，如有缺损问题，请向购买书店调换；若书店售缺，请与本社发行部联系。联系电话：（010）68279077。质量投诉请发邮件至zlts@phei.com.cn，盗版侵权举报请发邮件至dbqq@phei.com.cn。

前　言

你属于哪一类读者

- 你要做的事很多，有些细节对你并不重要，可以让他人去做。你并不打算亲自从事面向对象的分析或实现工作，但却对现代新技术怀有兴趣，并且会在新技术的实施时从事决策制定的工作。
- 你有多年从事软件开发的实践经验，知道软件是怎样开发的。在你看来，面向对象的程序设计方法已日趋成熟，从而准备拿出更多的时间关注这个领域。你希望获得某种面向实际的理论指导。
- 面向对象（OO）是你所熟悉的一种技术。一段时间以来，你一直关注着这一领域，可能还实际从事过面向对象程序的实现工作。因此，你的兴趣转向了分析与设计，以及面向对象方法论和表示方法的最新发展方面。
- 你关注软件开发技术，并在该领域积累了某些经验。对于面向对象方法论的基本概念已有所认识，进而需要针对该领域获得某种综合、系统的理论指导。

亲爱的读者

面对数百页的科技文献，你还肯花时间从头到尾去读吗？对于那些饱受厚书所困扰的读者，我试图向他们提供一本部头不大、面向实践的通俗读本。

本书采用模块化结构——从教学意义上说，各章节既自成篇章，又相互联系。因此，读者可选择通篇、交叉或跳跃等方式阅读。抓住本书要点的有效方法是通读“分析”与“设计”两章，同时查找和研究在“UML 基础”一章中的各个问题。

虽然本书只用较小的篇幅简单描述了UML（统一建模语言），但却包含了它的所有重要内容，其符号和语义是面向对象建模过程中所用的最新标准。尽管如此，本书首先是讲述面向对象分析和面向对象设计的入门级读本。对于UML基础知识的介绍，可见诸于面向对象软件开发的一般性问题和讨论的有关章节。为了更容易地切入主题，本书中不包含UML元模型的内容。实践中应用较少的特殊问题都用“高级UML”进行了标注，并且仅在需要时才加以论述。

本书中所讨论的重点内容，即用例驱动的、以体系结构为中心的渐进式开发方法主要应用于嵌入式公司信息系统的开发中，但也适用于其他技术和应用领域。

致　谢

在本书的写作过程中，我得到了我的许多朋友和同事的帮助，在此对他们表示深深的谢意。此外，还要感谢本书前一版的读者和参加我的研讨会的朋友们，谢谢他们的建议和批评意见。

Bernd Oestereich

对于下列许可使用他们的版权资料的人，谨在此表示最由衷的感激：

摘自Abziehbilder, heimgeholt. Essay 27, 得到了Literaturverlag Droschl, Graz-Wien 1995 (Kelly, R., Roubaud, J. and Schuldt, 1995)版权持有人的重印许可。摘自 Means of abstraction by Nicolai Josuttis, 得到了作者的许可。

在第 3 章和第 4 章中出现的方法论的探讨和使用得到了 oose.de GmbH 的许可。5.4.6 节是以 Christine Rupp、Sophist Group 的许可为基础发展起来的。

对本书中的某些实例，我们无法追踪到版权资料的拥有者，我们对能够使我们得到这一信息的任何帮助表示感谢。

目　录

第1章 引 言

本章解释了面向对象软件开发的特点、它的历史和它与传统方法的比较。

1.1 面向对象的软件开发

纲要

- 面向对象软件开发的优点是什么

使用推土机来种花与使用汤勺来从高楼中挖出东西一样，都用错了工具。那么正确的工具和方法是什么呢?

软件开发现在变得越来越复杂，但也变得越来越迷人了。软件系统是人类所创建的最复杂的系统之一。复杂的软件开发永远不会让人厌倦——它要求创造性、准确性、易于学习的能力，以及明智地分析和构造新事实的自觉性，有效地输入和输出通信（在开发团队内以及与客户和用户之间）。知道并且利用过程、方法、技术和工具等，可以明智地处理还没有开发好的问题、思想等。

高质量的软件开发变得越来越昂贵。从字母数字界面到事件驱动的图形用户界面（GUI）的变化，多层客户/服务器体系结构的引入，分布式数据库，Internet 等，使得软件开发的复杂性大大增加了。

在C++或类似的编程语言中实现这样的软件是一项劳动强度极大的工作。相反，可视及4GL 开发工具可以快速完成这项工作，但是也因此跳过了计划和形成概念的阶段，而使用了 FBTT（从想法到实现，From Brain to Terminal）方法。然而，不考虑实现所用的方法，在应用实现之前应该有一个计划阶段，这是本书的重点：现代软件的分析和设计。

1.1.1 技术复杂性

软件永远不会很完美地完成，有许多东西要修改或改进。既使不要求强制进行修改，进行修改活动也是令人赞赏的——直到这个软件从市场上消失为止。当然太多的修改和扩展很明显会引起程序越来越多地偏离原来的概念。当程序成功发布，正在不断地改进时，这种危险是特别关键的。

由于这些原因，寻找适当的方法，以求掌握复杂度或至少延缓衰退的过程并帮助维护软件的质量和可靠性，而不管结构的破坏和发展情况是非常明智的。

软件开发的历史是抽象性持续增加的历史，从位模式，通过宏指令、过程和抽象数据类型到对象、框架、设计模式和业务对象。面向对象的更强意义上的抽象不仅改进和进一步发展了传统的方法，也产生了一种新的思想方法。

虽然总是可以找到类似点和关系，但是简单的语句，如“消息仅是过程调用”或“面向对象只是一个有着新标签的老概念”会使我们误解面向对象的本质。

1.1.2　社会复杂性

当我们仔细考虑开发过程时，“软件开发主要是一项技术工作”这一广为流传的观点就有点失之偏颇了。今天，软件开发也是一个复杂的社会过程。其决定性的理由是：软件开发是一个与人有关的过程，其中心理、认识方面的知识和通信技术一起扮演着重要的角色。许多软件系统被嵌入到一个社会化环境中，即在一个涉及有人的组织中——这些是本书要研究的系统（与技术或其他系统相反）。

在应用领域的专家和在开发团队中的专家共同分享同一问题：越专业，就越难理解。然而，正确的相互理解是成功合作的基础。在完成某一项目的过程中，应创建应用世界的模型并开发软件——这些都存在于相关人员的头脑中。在这样的一个项目中生成的文档最多只是冰山的一角。因为有越来越多的东西存在于开发人员的头脑中，而不是显示在文档中，所以就整个内容征求多数人的同意就成为绝对必要的了。

因此，通信——交换模型、人们头脑中现有的知识和经验——就变成了一个核心问题。因为比起其他的技术系统，软件更多地和人的抽象、它的所有复杂性、不确定性和特殊性交缠在一起。比起其他典型的技术问题，它更类似于人类的组织结构。

1.2　面向对象的历史

纲要

- UML是怎样产生的
- 面向对象的历史背景和它的分析与设计方法

面向对象的概念可以追溯到30多年前，面向对象编程语言的开发也大约在同时开始。虽然不断地有关于面向对象编程的图书出版，但是关于面向对象分析设计方法的第一批书却是在20世纪90年代初才出现的。

这些包括由Booch、Coad、Yourdon、Rumbaugh et al、Wirfs-brock和Johnson、Shlaer和Mellor、Martin、Odell、Henderson-Sellers以及Firesmith撰写的书。尤其是Goldberg、Rubin和Jacobson更提出了新的重要的见解。许多方法专门限定于特定的应用领域。

在20世纪90年代初，Grady Booch和James Rumbaugh的方法成为迄今为止最受欢迎的。Rumbaugh方法是更面向结构的，而Booch的方法覆盖了商业和技术领域，也包括时

间关键（time-critical）的应用。在1995年，Booch和Rumbaugh开始第一次把他们的方法用一个共同符号的形式组合在一起，以创建“统一的方法”（UM）。很快这就被更名为“统一建模语言”（Unified Modeling Language, UML），统一建模语言事实上是一个更贴切的名字，因为它本质上是标准化图形表示和建模元素语义的问题，而不是描述一个特殊的方法。建模语言本质上是一个表示符号的特定风格方式。

后来不久，Ivar Jacobson带着他的所谓的用例（use case）加入进来了。此后，这三个人互称“朋友”。因为Booch、Rumbaugh和Jacobson的方法是如此之受欢迎且占据了一个相当大的市场份额，他们对UML的集成形成了一个准标准。最后，在1997年，UML版本1.1被提交给对象管理组（Object Management Group, OMG）进行标准化工作，并被接受。版本1.2、1.3和1.4包含了一些修正。版本2.0目前正由OMG制定中。读者可以在http: //www.omg.org/uml上找到最新的信息（参见图1.1和图1.2）。

UML主要描述了统一符号和语义，也包含一个元模型的定义。开发方法的描述并不是它的一部分——它仅是在1999年初在Jacobson撰写的“The Unified Software Development Process”，即所谓的统一过程（Unified Process，UP）中提出过。

在本书中，“分析”和“设计”两章描述了一个过程，这个过程以一个特定的“统一过程”表达式（统一过程在实践中是很成功的）为基础。本书重点在商业的嵌入式信息系统，例如我们在服务或商业企业中遇到的那些。关于更加面向技术的系统，例如嵌入式和实时系统，可能需要一些调整。

UML包含几方面的内容，它也集成了其他作者的有趣的思想和概念。因此，除了Booch、Rumbaugh和Jacobson的思想外，你也会看到Harel的想法贯穿其中。

感谢“Amigo”，不同的符号已糅合在一起。另一方面，也存在独立的符号，不同的方法，例如Shlaer和Mellor的方法，以及“开放过程”（Open Process），它和OML组合在一起，OML作为它的建模语言。有30多个支持者属于OPEN组织，其中有Brian、Henderson-Sellers、Ian Graham和Donald Firesmith。

统一建模语言是一种模型化语言，本身要翻译为编程语言，即要生成代码。一方面，它的高级表示使之对CASE工具的要求很高，也使得逆工程变得非常困难。另一方面，它提供了广泛的建模功能。现有的工具远不能充分利用UML（参见http://www.oose.de/uml）。

1.2.1 太多的选择

面向对象软件开发方法论的基本概念是成熟的，并已在实践中广泛使用。然而，UML特别提供了丰富的细节，因此应小心使用。

我们认为在描述时的各式各样的可能性是相当令人困惑的，深刻理解UML在构造方面的变化需要花一定的时间和精力。因此，首先，人们应该把自己限制到基本元素上。虽然，潜在的语义方面的差别可能保留在建模中，但是，在实际中，在这个层面的工作已足够了。此外，也存在许多根本不适合语义的地方。

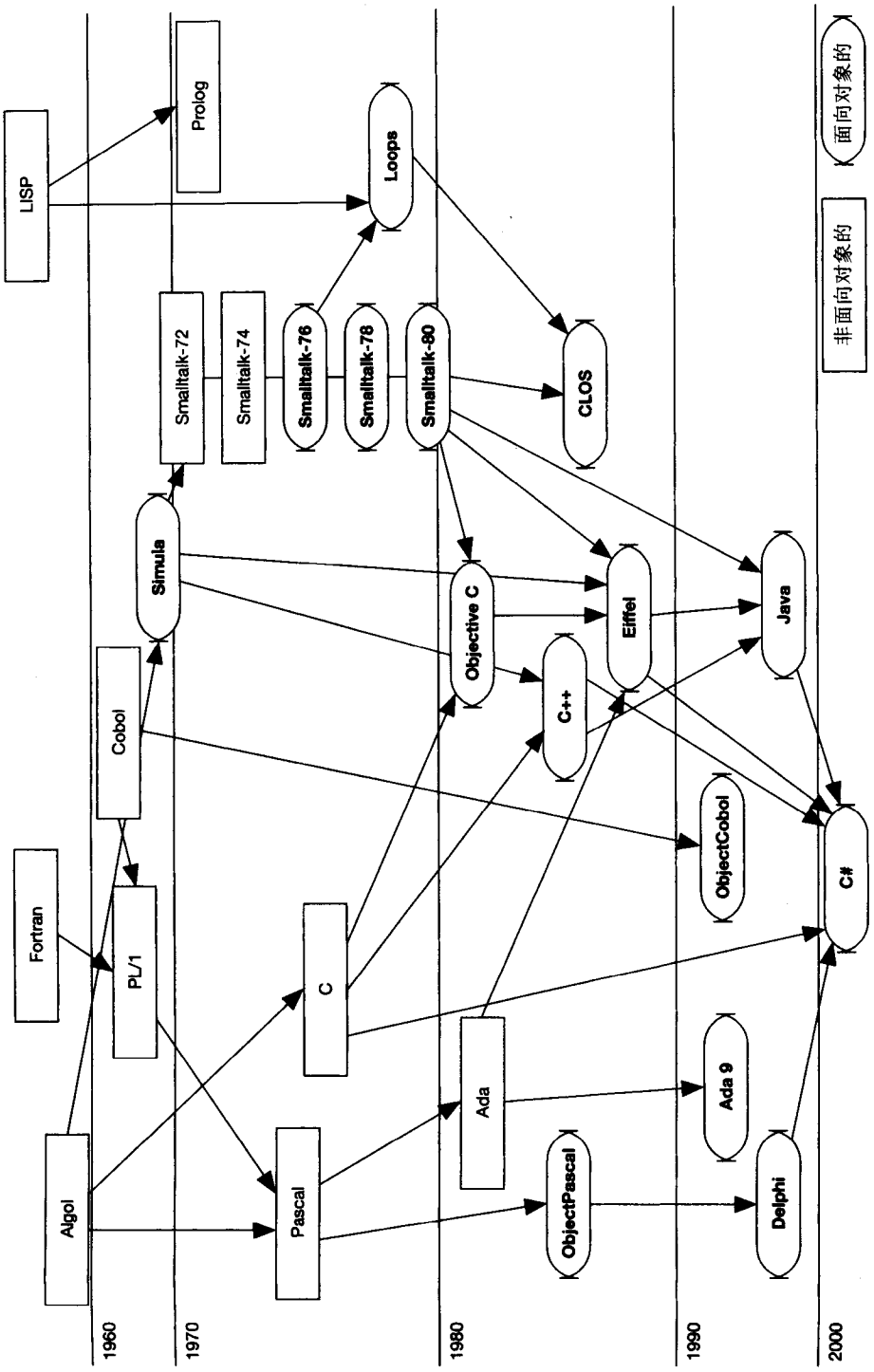


图 1.1 面向对象语言的历史发展情况

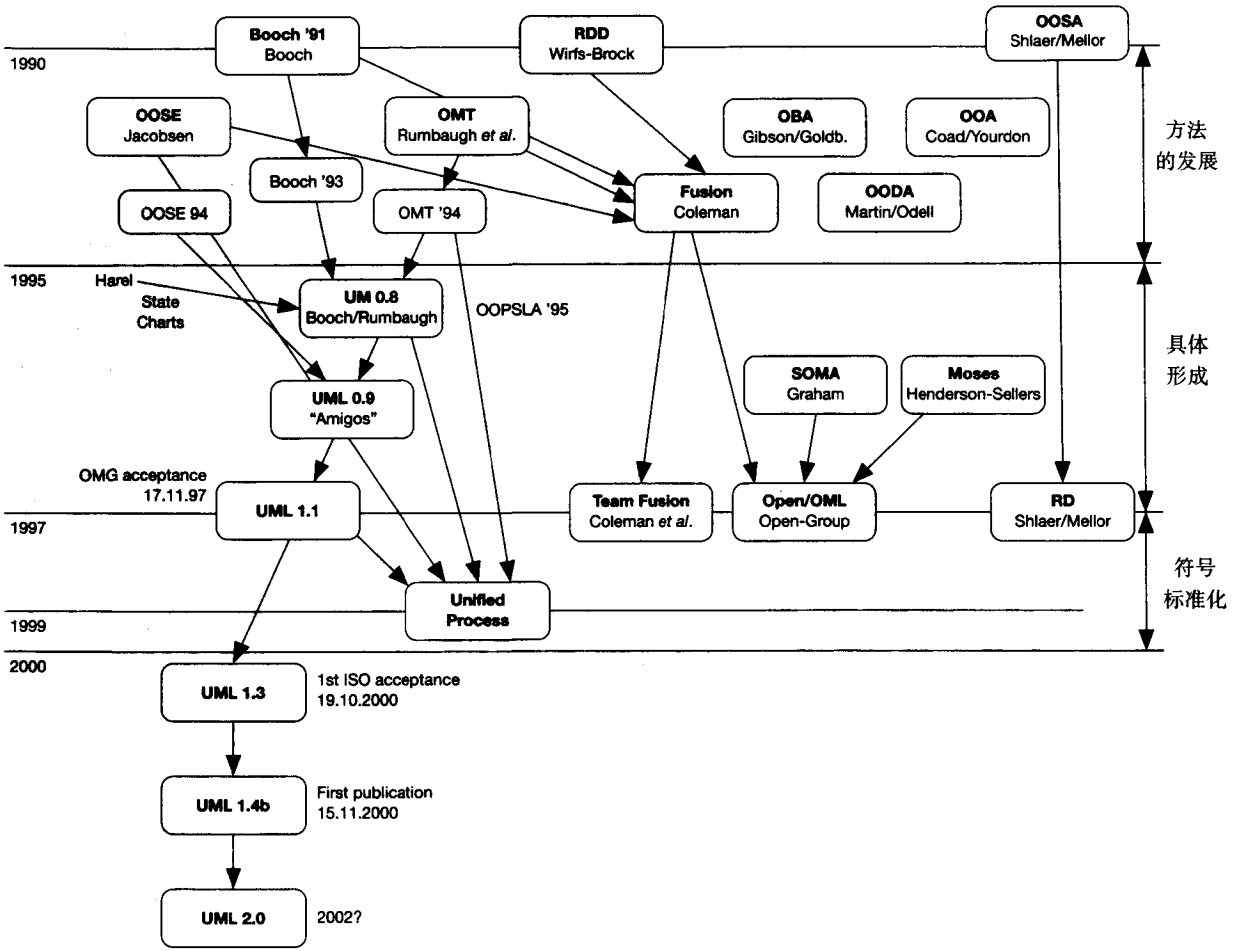


图 1.2　面向对象方法和 UML 的历史发展情况

还有，某些方法论方面的概念仅在特定的或高度细节化的环境中才有意义。这里，对所用元素的选择要受应用（信息系统、技术系统、实时应用等）和所需细节深度的指导。细节，例如可见性标记、复杂的原型等不仅导致更好的安全性和更高的质量，也会导致更高的开发努力和成本。与其他方法相比，UML的优点是：由于各种替代方法的可得到性，因此对符号或语义方面的限制就不那么严格了，当然，关于替代方法，只有相关和必需的那些才会被应用。

1.3　实践中的OOAD

纲要

- 不可盲目应用UML和基于它的方法。

如果你希望在你的公司建立一个面向对象的方法论，你可能会决定采用UML，使用特定的CASE工具并遵循一个特殊的开发过程，如统一过程（Unified Process）。

这些决定可能很好，但是它们不能解决所有的软件开发问题。相反，它们会创建某些新问题。UML是强大、精细的，可能包含大量你不需要的元素。此外，在UML、工具和方法/过程之间的交互不很明显，但是在你的项目中必须首先测试和确定。

因此，首先分析什么是典型的问题，其边界条件是什么，在处理它们时什么方法合适是明智的。如果开发人员只在方法论方面受过培训，但是由于部分方法论是不相关的，基本问题讨论的不够充分或不被所用的工具支持，不能把他们的知识应用到实践中，则会导致某种困境。试着设计出一个简单的方法论，省略所有多余的概念，然后系统地引入它。本书不是要介绍UML的所有细节，这些细节不是你所需要的。当介绍面向对象的方法论时，将使读者首先集中在核心领域，对此熟悉之后再扩展到符号、方法论和很好定义的过程，这对读者会很有帮助。