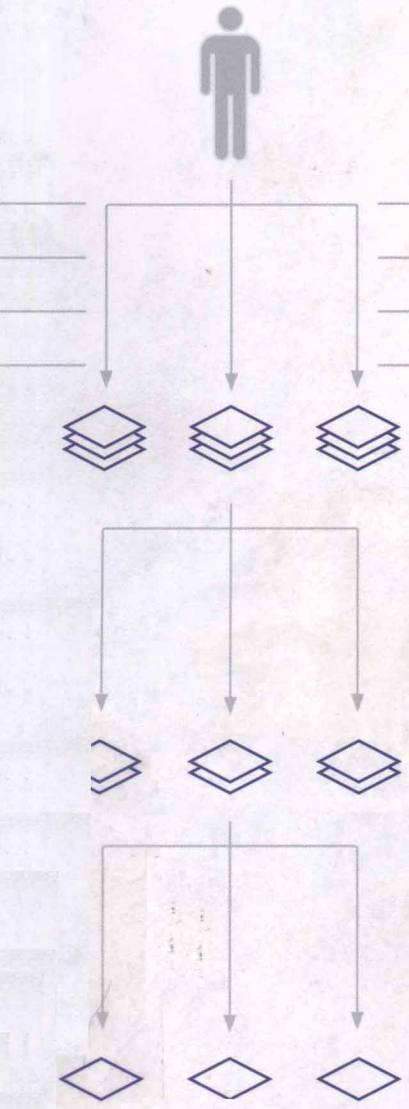
UML

UML2.0基础与RSA建模实例教程

曹衍龙 汪杰 编著

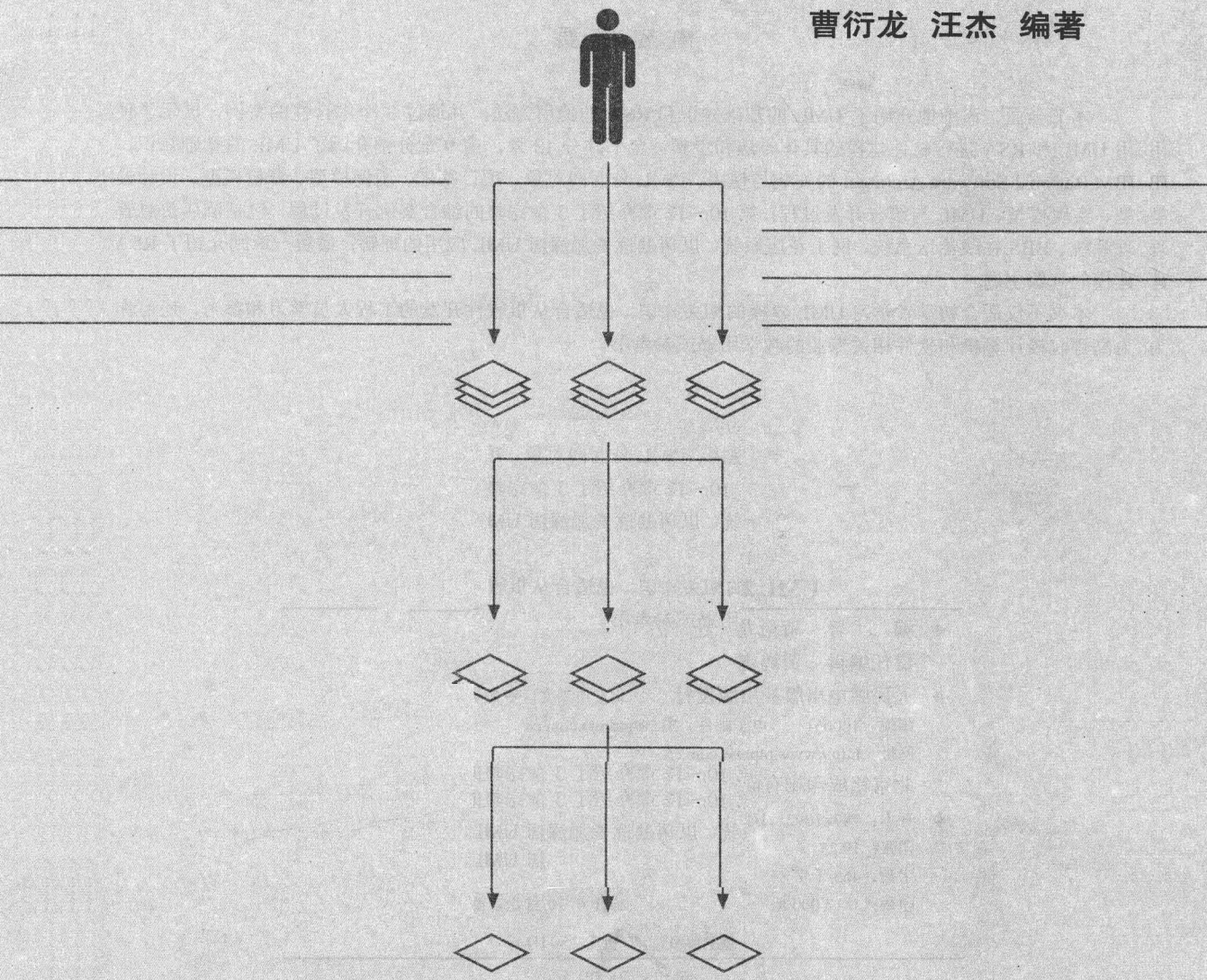


详解UML基础知识
细数各种模型的创建过程
剖析RSA建模综合实例
秉承软件工程的思维模式

人民邮电出版社
POSTS & TELECOM PRESS

UML2.0基础与RSA建模实例教程

曹衍龙 汪杰 编著



人民邮电出版社
北京

图书在版编目（CIP）数据

UML 2.0基础与RSA建模实例教程 / 曹衍龙，汪杰编著. -- 北京 : 人民邮电出版社，2011.10
ISBN 978-7-115-25919-6

Ⅰ. ①U… Ⅱ. ①曹… ②汪… Ⅲ. ①面向对象语言，UML－程序设计－教材②软件开发－教材 Ⅳ. ①TP312②TP311.52

中国版本图书馆CIP数据核字(2011)第144740号

内 容 提 要

本书全面、详细地介绍了 UML 的基础知识和 RSA 的使用方法，并通过 3 个综合性的案例，展示了使用 UML 和 RSA 进行软件建模的具体方法和步骤。全书共分 13 章，前 9 章分别介绍了 UML 的基础知识、IBM Rational Software Architect 的安装与使用、UML 与面向对象、项目基础、用例模型、分析模型、设计模型、实现模型、UML 与统一开发过程；第 10～12 章介绍了 3 个完整的综合案例开发过程，包括酒店信息管理系统、BBS 在线论坛系统、网上花店系统，以帮助读者加深对 UML 应用的理解；最后一章则介绍了 RSA 建模的高级主题。

本书不仅适合初学者学习 UML 建模的相关知识，也适合从事软件开发的工程人员学习和参考，还可作为高等院校计算机和软件相关专业的教学用书或参考书。

UML 2.0 基础与 RSA 建模实例教程

◆ 编　　著　曹衍龙　汪　杰
责任编辑　贾鸿飞

◆ 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　　电子邮件　315@ptpress.com.cn
网址　http://www.ptpress.com.cn
北京铭成印刷有限公司印刷

◆ 开本：787×1092　1/16
印张：19.25
字数：465 千字　　2011 年 10 月第 1 版
印数：1－3 000 册　　2011 年 10 月北京第 1 次印刷

ISBN 978-7-115-25919-6

定价：38.00 元

读者服务热线：(010)67132692　印装质量热线：(010)67129223
反盗版热线：(010)67171154
广告经营许可证：京崇工商广字第 0021 号

前　言

20 世纪 50 年代，软件诞生。20 世纪 60 年代，软件工程的概念被提出。20 世纪 70 年代，面向对象的建模语言出现。软件工程领域在 1995—1997 年取得了前所未有的进展，其成果超过过去 15 年来的成就总和。其中最重要的、具有划时代意义的成果之一就是统一建模语言（Unified Modeling Language，UML）。1997 年，UML1.1 被 OMG（对象管理组织）采纳，成为了软件工业界事实上的标准。有了多年的经验后，OMG 提出了升级 UML 的建议方案，以修正使用中发现的问题，并扩充一部分应用领域中所需的额外功能，这个版本就是本书所介绍的 UML 2.1。

UML 是用来对软件系统进行描述、构造、可视化和文档编制的一种语言，它统一了 Booch、Rumbaugh 和 Jacobson 的表示方法，而且做了进一步的发展，并最终成为标准的建模语言。

Rational Software Architect（RSA）是由美国 IBM 公司的 Rational Software 部门（前身为独立的 Rational 公司，现为 IBM 所收购）开发的产品。RSA 是一个基于 UML 2.1 的可视化建模和架构设计工具，构建在 Eclipse 开源框架之上，具备可视化建模和模型驱动开发的能力。利用这个工具，可以建立用 UML 描述的软件系统模型，而且可以自动生成和维护 C++、Java、Visual Basic 和 Oracle 等语言和系统的代码，也适用于 Web Services。

为什么写本书

RSA 为基于 UML 进行业务建模，并完成底层代码生成的开发人员提供了可视化的建模环境，得到了广泛的应用。RSA 的 7.5 版本也是当下应用最多的版本之一，无论是实际开发人员，还是高校教学工作者，都对 7.5 版本有着很强的学习需求。

目前，国内介绍 UML 和 RSA 的资料较少，内容不够细致具体，特别是缺少综合介绍两者的中文资料。为此我们编写了本书，旨在帮助读者全面了解 UML 的相关知识，学习并实践使用 RSA 建模的具体方法和步骤。

本书的特点

- 内容全面。书中详细地介绍了 UML 的基础知识，如视图、图、模型元素和通用机制等，同时，结合具体的案例，给出了相关理论在 RSA 中的建模实践。值得一提的是，书中介绍了国内 UML 相关书籍中很少提及的正向工程和逆向工程在 RSA 中的具体实现方法。
- 案例丰富。全书共提供了 3 个完整的综合性 RSA 建模案例，即酒店信息管理系统、BBS 在线论坛系统和网上花店系统。另外，图书馆管理系统的建模案例贯穿于 UML 基础知识的相关章节，有助于读者边学习、边思考、边实践。
- 图文并茂。书中的各个章节都配有大量的设计流程图和建模图，有助于读者更加直观地理解 UML 的理论知识，并在实际的学习和工作中学以致用。

读者对象

本书适合从事面向对象软件开发的软件工程人员，如系统分析员、系统设计员、项目经理和实际的开发者等进行学习和参考，也适合高等院校计算机软件工程相关专业作为教学用书或参考书。

技术支持

为了方便高等院校相关专业的教学，本书配备了教学大纲和课件，如果需要，可以在人民邮电出版社的网站 www.ptpress.com.cn 获取，或者与本书的责任编辑联系，责任编辑的 E-mail 是 jiahongfei@ptpress.com.cn。

本书主要由曹衍龙和汪杰编写，负责图片绘制和资料整理的有李蒙、茅健、陈晓和、万星新和赵玉玺等。在编写过程中，我们站在读者的角度力求精益求精，但由于水平与时间所限，难免存在一些不足之处，如果读者使用本书时，发现差错或遇到问题，可以发 E-mail 到 jiahongfei@ptpress.com.cn 与我们联系。

编　者

2011 年 8 月

目　录

第 1 章　初识 UML 建模

本章对软件工程、软件模型和 UML 进行简要的介绍。通过对本章的阅读，读者可以对软件工程、软件模型和 UML 有一个清晰的概念，为进一步利用 UML 进行软件的设计和开发打下良好的基础。

1.1　软件工程概述

经过长期的发展，软件工程作为一门工程学已经形成了自己的体系结构。本节将主要介绍软件工程的历史和比较流行的软件开发模式。

1.1.1　软件工程的提出

1946 年，第一台通用电子计算机“埃尼阿克”（ENIAC）在美国诞生，宣告一个新的时代的到来。在接下来的短短数十年间，晶体管计算机、集成电路计算机和超大规模集成电路计算机相继诞生，计算机硬件设备经历了高速的发展时期。时至今日，令人惊讶不已的摩尔定律依然成立。然而，计算机软件的发展却经历了崎岖不平的道路。

在 20 世纪 60 年代计算机技术发展的初期，程序设计充满了“个人英雄主义”。一个或几个能力出众的程序员随心所欲地编程，产生出的代码不易被其他人理解和修改。随着软件规模的扩大和复杂度的日益增加，很多软件最后都得到了一个失败的结局。不少软件项目开发时间大大超出了规划的时间表，一些项目导致了财产的流失和开发费用的急剧增加，同时软件开发人员也发现软件开发的难度越来越大。人们把这种现象称为“软件危机”，即软件开发从质量、效率等方面均不能满足应用需求。

为了解决“软件危机”这一问题，1968 年，在 NATO 会议上首次提出了“软件工程”这一概念，使软件开发开始了从“个体行为”向“工程”的转化历程。

40 多年来，尽管软件开发中仍然存在许多无法根治的问题，但软件产业的发展速度却超过了任何的传统工业，人们所担心的“软件危机”并没有出现。通过一代又一代研究人员和工程人员的探索和实践，软件工程的理论体系得到了不断的充实和完善，形成了一门严谨、完备和实用的学科。

1.1.2　软件开发模式

常用的软件开发模式有：瀑布模式、迭代模式、螺旋模式、快速原型模式、形式化描述模式等。其中瀑布模式和迭代模式比较常用。

1. 瀑布模式

瀑布模式将软件生命周期划分为需求分析、软件设计、程序编写、软件测试和运行维护等 5 个基本活动，并且规定了它们自上而下、相互衔接的固定次序，如同瀑布流水，逐级下落。从本质来讲，开发过程是通过一系列阶段顺序展开的，从系统需求分析开始直到产品发布和维护，每个阶段都会产生循环反馈。因此，如果有信息未被覆盖或者发现了问题，那么

就要返回上一个阶段并进行适当的修改。

瀑布模式的核心思想是按工序将问题化简，将功能的实现与设计分开，便于分工协作，即采用结构化的分析与设计方法将逻辑实现与物理实现分开。采用瀑布模式的软件过程如图 1-1 所示。

2. 迭代模式

迭代模式是 RUP（Rational Unified Process，统一软件过程）推荐的周期模型。在 RUP 中，迭代被定义为：产生可发布产品的全部开发活动。所以，一次迭代经历了完整的工作流程：至少包括了需求分析、系统分析与设计、系统实现和测试。

实质上，它类似小型的瀑布式项目。所有的阶段都可以细分为迭代。每一次的迭代都会产生一个可以发布的产品，这个产品是最终产品的一个子集。一次迭代经历的流程如图 1-2 所示。

图 1-1　瀑布模式阶段划分

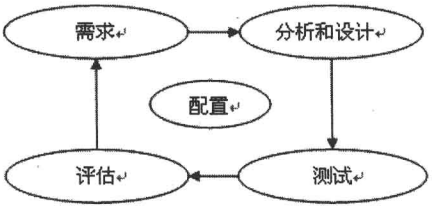


图 1-2　迭代所经历的流程

1.2　软件模型概述

软件工程的研究表明，一个软件项目要想获得成功，首先必须准确地理解用户的需求。在此基础上，才能对软件系统进行正确的分析和设计，并且证明这种设计是切实可行的。在这之后，才能对项目的费用和开发周期作出比较合理的估计。

对软件进行建模，就能够在很大程度上实现上述目的。本节将介绍什么是软件模型，以及软件建模的目的和重要性。

1.2.1　什么是软件模型

模型实际上是对现实世界的抽象和简化。它从某一个角度出发，抓住了事物最重要的方面而简化或忽略其他方面。

在工业界中，模型是零件的设计图，它精确定义了零件的大小尺寸和材质，具备了制造一个零件所需要的所有信息；在建筑业中，模型是用简易材料搭建起来的仿真品，它具备了与真实建筑一样的外观和结构，只不过经过了一定的比例缩放；在经济学中，模型是用来分析和预测经济走势的一些数学原理和公式，通过这些模型，经济学家就可以刻画和分析经济运行的规律和特点。

在软件业中，模型包含了构建软件系统的蓝图。从描述层次上来说，其中有对系统的总体设计，也有对系统某个局部的详细设计。从描述角度上来说，其中有对系统静态结构的刻画，也有对动态行为的说明。软件模型是从不同的角度对系统的近似表述。

1.2.2 软件建模的目的

“凡事预则立，不预则废”，可见在很早以前，我们的先辈就知道进行一项工作应当预先进行谋划，制订出工作计划，这样就能够提高办事的效率和成功率。

在现代社会，这样的例子比比皆是。

如果你想给打球的孩子们做个简易的遮阳棚，当你找齐了必需的材料和工具之后，稍微计划一下，就能在很短的时间内做出一个完全可以使用的遮阳棚。

但如果你想建造一个堪比“鸟巢”的大型体育场，那么立刻开始购买建筑材料和匆忙破土动工将是十分愚蠢的行为。因为，你正在使用政府的资金，他们将决定这个体育场的规模、形状和风格。通常，根据实际情况工程随时可能发生变化，你必须把这些变化也考虑在内，因为失败对任何一方来说都是无法承受的。于是，你可能会组建几个不同的工作小组来分别负责某一方面的工作。小组之间和政府官员之间通过各种各样的文件、图纸和模型进行沟通。不仅如此，你可能还需要若干表格来记录详细的工程实施计划，以此确保不会延误工期。

通过上述例子不难看出，人对复杂系统的理解力是有限的。而模型是对现实的简化，通过建模，可以将抽象的事物具体化，得到对工程的准确理解，并且大大方便人与人之间的沟通和交流。不仅如此，通过建模还能够缩小所研究问题的范围，一次只研究它的一个方面，实现对问题的各个击破。软件建模就像是建设体育场之前的规划设计，它使得软件开发的流程变得清晰、明确和可以控制。

总的来说，通过建模能够达到如下目的。

（1）直观化：以更直观的形式来表达系统或系统的某个方面。

（2）说明功能：模型可以详细说明系统的结构或行为。

（3）指导功能：模型可以指导我们构造系统。

（4）文档化：模型对我们做出的决策进行文档化。

1.2.3 软件建模的重要性

在软件开发的实践活动中，总会有程序员抛开软件建模而直接进行软件开发。也许有时候你是幸运的。如果面对的是一个“遮阳棚”级别的软件，在恰当的时间有足够的合适的人员，并且其他一切事情都很如意，那么你的团队有可能成功推出一个满足用户需求的软件产品。然而，一般的情况下，不可能所有人员都合适，时间并不总是充裕的，其他的事情也并不尽如人意。随着对软件开发要求的日益增长，软件系统的规模和复杂度也会迅速增加。当你面对的软件是“体育场”级别的时候，一切就变得困难得多了。

事实上，许多软件在开发之初看上去都像“遮阳棚”级别的，但如果事先不对系统做出充分的分析和设计，它迟早会膨胀成“体育场”级别的。此时再追悔莫及，就为时已晚了。在这方面，最著名的反面教材要数 IBM 的 OS/360 操作系统的开发了。这个操作系统在开发过程中不断加入了新的开发理念，不断有新的程序员加入和离开，导致的结果是发布日期一再推迟，直到 1967 年中期才蹒跚走入外面的世界，这比原计划推迟了近一年。此时，IBM

已经花了五亿美元在这个项目上，比原先的预算多了四倍。既便如此，整个系统依然漏洞百出。软件存在需要几年才能解决的几十种错误。往往旧的错误解决了，新的错误却出现了。这就像要修补一个漏水的水箱一样。

通过这个失败的例子，我们可以充分看到软件建模的重要性。首先，不建模就无法对开发目标和最终产品有明确的认识，项目的参与者只是盲目地专注于自己的工作，却没有兼顾全局。其次，开发人员之间没有交流的媒介，随着人员的变动，软件系统的设计理念和要达到的目标就渐渐变得模糊了。最后，没有模型就无法对时间和经费作出预测，毫无根据的估算与实际情况严重背离，极大打击了项目人员的自信心。

因此，在软件开发之前，利用软件工程的方法对系统进行建模是十分有必要的，它将大大提高开发效率，显著地减小软件项目失败的概率，奠定通向成功之路的基石。

1.2.4 软件建模的基本原理

各种工程学都有丰富的建模使用历史。这些经验形成了建模的四项基本原理。

第一，选择要创建什么模型对如何动手解决问题和如何形成解决方案有着意义深远的影响。换句话说，就是要好好地选择模型。正确的模型将清楚地阐明难以对付的开发问题，提供不能轻易从别处获得的洞察力；错误的模型将误导你，使你将精力花在不相关的问题上。

第二，每一种模型可以在不同级别上表示。最好的模型应该是这样的：它可以让你根据观察的角色以及观察的原因选择它的详细程度。分析人员或最终用户主要考虑“做什么”的问题；开发人员主要考虑“怎么做”的问题。这两类人员都要在不同的时间以不同的详细程度对系统进行可视化。

第三，最好的模型是与现实相联系的。在理想状况下，最好是有一个能够清晰地联系实际的模型，而当联系很薄弱时能够精确地知道这些模型怎样与现实相脱离。所有的模型都对现实进行了简化，但有一点要记住，不能简化任何重要的细节。

第四，单个模型是不充分的。对每个重要的系统最好用一组几乎独立的模型去处理。为了理解系统的体系结构，你需要几个互补和连锁的视图。每一种视图都可能有结构方面和行为方面的部分。这些视图一起从整体上描绘了软件蓝图。

1.3 UML 概述

UML（Unified Modeling Language，统一建模语言）是一种通用的、面向对象的、可视化建模语言。它的主要作用是帮助用户对软件进行面向对象的描述和建模，它可以描述这个软件开发过程从需求分析直到实现和测试的全过程。

UML 本质上不是一门编程语言，它缺少大多数编程语言提供的语法和语义。但是可以使用代码生成器将 UML 模型转换为多种程序设计语言代码，或使用反向生成工具将程序代码转换成 UML。作为一门通用的建模语言，它适合对诸如计算机软件、固件和数字逻辑系统进行建模。

UML 通过建立各种类与类之间的关联、类与对象之间的动态行为等成分来组建整个模型。UML 提供了各种图来把这些模型元素可视化，使得人们可以清楚容易地理解模型。此外，人们可以从多个视图来观察模型，每个视图包含了若干相关的图，这样可以更全面地了解模型。

1.3.1 UML的产生和演变

公认的面向对象建模语言出现于20世纪70年代中期。从1989年到1994年，其数量从不到10种增加到了50多种。在众多的建模语言中，语言的创造者努力推崇自己的产品，并在实践中不断完善。但是，OO方法的用户并不了解各种建模语言的优缺点及相互之间的差异，因而很难根据应用特点选择合适的建模语言，于是爆发了一场“方法大战”。20世纪90年代中，一批新方法出现了，其中最引人注目的是Booch 1993、OOSE和OMT-2等。

Booch是面向对象方法最早的倡导者之一，他提出了面向对象软件工程的概念。1991年，他将以前面向Ada的工作扩展到整个面向对象设计领域。Booch 1993比较适合于系统的设计和构造。Rumbaugh等人提出了面向对象的建模技术（OMT）方法，采用了面向对象的概念，并引入各种独立于语言的表示符。这种方法用对象模型、动态模型、功能模型和用例模型共同完成对整个系统的建模，所定义的概念和符号可用于软件开发的分析、设计和实现的全过程，软件开发人员不必在开发过程的不同阶段进行概念和符号的转换。OMT-2特别适用于分析和描述以数据为中心的信息系统。

Jacobson于1994年提出了OOSE方法，其最大特点是面向用例（Use Case），并在用例的描述中引入了外部角色的概念。用例的概念是精确描述需求的重要武器，但用例贯穿于整个开发过程，包括对系统的测试和验证。OOSE比较适合支持商业工程和需求分析。此外，还有Coad/Yourdon方法，即著名的OOA/OOD，它是最早的面向对象的分析和设计方法之一。该方法简单、易学，适合于面向对象技术的初学者使用，但由于该方法在处理能力方面的局限，目前已很少使用。

概括起来，首先，面对众多的建模语言，用户由于没有能力区别不同语言之间的差别，因此很难找到一种比较适合其应用特点的语言；其次，众多的建模语言实际上各有千秋；最后，虽然不同的建模语言大多雷同，但仍存在某些细微的差别，极大地妨碍了用户之间的交流。因此在客观上，极有必要在精心比较不同的建模语言优缺点及总结面向对象技术应用实践的基础上，组织联合设计小组，根据应用需求，取其精华，去其糟粕，求同存异，统一建模语言。

1994年10月，Grady Booch和Jim Rumbaugh开始致力于这一工作。他们首先将Booch9 3和OMT-2统一起来，并于1995年10月发布了第一个公开版本，称之为统一方法UM 0.8（Un- itied Method）。1995年秋，OOSE的创始人Ivar Jacobson加入到这一工作。经过Booch、Rumbaugh和Jacobson三人的共同努力，他们分别于1996年6月和10月发布了两个新的版本，即UML 0.9和UML 0.91，并将UM重新命名为UML（Unified Modeling Language）。1996年，一些机构将UML作为其商业策略已日趋明显。UML的开发者得到了来自公众的正面反应，并倡议成立了UML成员协会，以完善、加强和促进UML的定义工作。当时的成员有DEC、HP、I-Logix、Itellicorp、IBM、ICON Computing、MCI Systemhouse、Microsoft、Oracle、Rational Software、TI以及Unisys。这一机构对UML 1.0（1997年1月）及UML 1.1（1997年11月17日）的定义和发布起了重要的促进作用。

UML是一种定义良好、易于表达、功能强大且普遍适用的建模语言。它融入了软件工程领域的新思想、新方法和新技术。它的作用域不限于支持面向对象的分析与设计，还支持从需求分析开始的软件开发的全过程。UML的产生、发展和完善的过程如图1-3所示。