全国计算机继续教育研究会推荐用书

计算机新技术丛书

UML与系统分析设计

张龙祥　编著

人民邮电出版社
www.pptph.com.cn

全国计算机继续教育研究会推荐用书

计算机新技术丛书

UML与系统分析设计

张龙祥　编著

人民邮电出版社

图书在版编目(CIP)数据

UML 与系统分析设计/张龙祥编著.—北京：人民邮电出版社，2001.8
（计算机新技术丛书）
全国计算机继续教育研究会推荐用书
ISBN 7-115-09551-5

I. U... II.张... 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(2001)第 050080 号

内 容 提 要

本书介绍了 UML 语言以及 UML 在面向对象的软件系统分析和设计中的应用。

本书理论与实际相结合，既有 UML 语言的概念、结构、语义与表示法的介绍，又有具体的应用示例，着重实用性和可操作性，叙述深入浅出，便于学以致用。

本书是一本技术参考书，可作相关专业的大专院校教材和继续教育的教材。本书适合计算机项目管理人员、计算机软件开发人员与程序员、大专院校有关专业的师生使用。

全国计算机继续教育研究会推荐用书
计算机新技术丛书
UML 与系统分析设计

◆ 编　　著　张龙祥
责任编辑　邹文波
执行编辑　苗　颖
◆ 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@ pptph.com.cn
网址　http://www.pptph.com.cn
读者热线:010-67129212　010-67129211(传真)
北京汉魂图文设计有限公司制作
北京朝阳隆昌印刷厂印刷
新华书店总店北京发行所经销

◆ 开本:787 × 1092　1/16
印张:15.25
字数:362 千字　　2001 年 8 月第 1 版
印数:1 – 5 000 册　　2001 年 8 月北京第 1 次印刷

ISBN 7-115-09551-5/TP·2401

定价:25.00 元

《计算机新技术丛书》编委会

《计算机新技术丛书》
出版说明

计算机科学和技术的迅猛发展，强有力地推动了世界经济、社会的飞速发展。当今世界各国毫无例外地将大量的人力、财力、物力投向计算机科学、技术、工业、教育，以争夺这个对军事、政治、经济、社会发展具有决定意义的“制高点”。在这样的形势下，计算机科技的交流、推广成为急迫而又繁重的任务。

近几年来，我们与电子工业出版社、湖北科技出版社、西南交大出版社合作出版了《计算机继续教育》、《当代微机技术》、《今日微机技术》等丛书。从1994年起，与西南交通大学出版社合作编辑、出版了《全国计算机新科技与计算机继续教育论文集》，该论文集从1995年起每年出版上、下两集，到2000年已出版13集。这些丛书、论文集对于推广、交流计算机新技术和开展计算机继续教育起到了积极的作用，已经成为我国的计算机新技术交流、推广、教育的一个阵地。

但是，如何更快、更好地使从事计算机工作的专业工作者以及所有在自己的研究和工作中应用计算机的非计算机专业工作者，尽快地获取国外最新的计算机理论和技术，是一个值得进一步探讨的问题。2000年8月“全国计算机新科技与计算机继续教育学术会议”（新疆会议）讨论认为，我们能做到的一个好的办法是，组织人力、物力尽快地编写《计算机新技术丛书》，以尽快地引进先进国家的计算机新技术、新理论。关键在一个“新“字。不论层次高低，只要是新出现的、对我国有益的新技术、新理论我们都应尽快引进，以缩短我国科技队伍的计算机及其应用的整体水平与先进国家之间的差距，为我国的计算机事业出一把力。

由于这套丛书定位于“新”，必然涉及到计算机科技各个层次、各个领域，因此，该丛书将适用于各科研、工程机构，各院校研究生、大学生，以及急迫需要使用计算机新技术的工作人员。我会向全国各层次希望学习、使用计算机新技术的人员推荐这套丛书。我们欢迎已经掌握某项计算机新科技的学者加盟该丛书的编写工作，使该丛书更具有活力。

经过作者、出版社以及我们学会的努力，这套丛书终于开始出版了，我们预祝该丛书能达到预期的效果。

在本丛书的组织、编辑、出版过程中，得到了许多专家、教授、院校及出版社的支持，我们表示深切的感谢！

全国计算机继续教育研究会

2001年6月

前　言

面向对象的分析和设计方法已逐渐取代了传统的方法，成为我国当前计算机软件工程学中的主流方法。目前国内使用的面向对象的分析和设计方法主要有 Booch、Jacobson、Rumbaugh、Yourdon、Meyer、Martin 等人设计的方法，它们各有特色，也各有不足之处，而且术语不统一，缺乏共同的标准，常给软件开发人员带来困惑。

从 1995 年起，著名的软件工程学家 Grady Booch、Ivar Jacobson、Jin Rumbaugh 先后齐集于 Rational 公司，携手合作、共同努力，综合了他们各自原创的面向对象的分析与设计方法，加以扩充改进，并汲取其他同类方法的优点，提出了统一模型语言 UML（The Unified Modeling Language）。1997 年 UML 被美国工业标准化组织 OMG（Object Management Group）接受，并在当年发布了标准版本 UML 1.1。经过不断的使用、修改、补充、完善，UML 日趋成熟，得到了许多计算机厂商如 IBM、HP、SUN、Oracle、Microsoft 等的支持，已成为国际上领先的软件开发的有效方法和工具，得到了广泛的推广应用。

1996 年美国 Rational 公司曾将支持 UML 的 CASE 软件 Rational Rose 赠送给我国的中国人民大学等高校，以便介绍与推广 UML。自此 UML 逐渐在我国得到流传、研究和应用。但是目前在我国了解与掌握 UML 的人尚不多，只有少数大型计算机企业采用 UML 进行软件开发。本书的目标是针对我国软件开发的现状，介绍当前国际上较为先进的用于软件分析与设计的统一模型语言 UML，以及运用 UML 进行面向对象的软件开发的方法和工具。

本书的内容由 4 个部分组成：第一部分为第 1 章和第 2 章，叙述了面向对象的软件开发方法与过程；第二部分为第 3 章至第 10 章，叙述了 UML 的语言成分和表示法，及其应用；第三部分为第 11 章，介绍了支持 UML 的 CASE 软件开发工具 Rational Rose；第四部分为第 12 章，叙述了一个简易教学管理系统的分析与设计，作为运用 UML 的系统分析与设计的简单示例。

本书理论与实际相结合，既有 UML 语言的概念、结构、语义与表示法的介绍，又有具体的应用示例，着重实用性和可操作性，叙述深入浅出，便于学以致用。

本书是一本技术参考书，可作大专院校相关专业的教材。读者对象是计算机项目管理人员、计算机软件开发人员与程序员、大专院校有关专业的师生等。

中南大学铁道校区计算机科学系的刘伟荣老师帮助分析设计了简易教学管理系统并绘制了部分图形。在本书的编写过程中得到了全国计算机继续教育研究会的大力支持，特在此表示衷心的感谢。

作者

2001 年 5 月

目　录

第 1 章　基础知识

自 20 世纪 40 年代发明计算机以来，计算机的应用得到了迅猛的推广，使得计算机技术蓬勃发展。然而，长期以来计算机软件开发的低效率制约着计算机行业的发展。计算机业界努力探索和研究解决软件危机的途径，提出了软件工程的思想和方法，软件的开发方法也从传统的软件生命周期方法发展到了面向对象的方法，极大地提高了软件开发的效率和软件质量。

面向对象的分析和设计方法已逐渐取代了传统的方法，成为我国当前计算机软件工程学中的主流方法。目前国内使用的面向对象的分析与设计方法主要有 Booch、Jacobson(OOSE)、Rumbaugh(OMT)、Coad-Yourdon、Martin-Odell、Fusion、Shlaer-Mellor、Sally Shlaer 等人的方法，它们各有特色，也各有不足之处，而且术语不统一，缺乏共同的标准，常给软件开发人员带来困惑。

从 1995 年起，著名的软件工程学家 Grady Booch、Ivar Jacobson、Jin Rumbaugh 携手合作、共同努力，综合了他们各自原创的面向对象的分析和设计方法，加以扩充改进，并汲取其他同类方法的优点，提出了统一模型语言 UML（The Unified Modeling Language）。经过不断的使用、修改、补充、完善，UML 已趋于成熟，成为了一种软件开发的有效工具。1997 年 UML 被美国工业标准化组织 OMG（Object Management Group）接受，并发布了 UML 的标准版本，提供给计算机业界使用。

UML 是一种编制软件蓝图的标准化语言，它提供了一套描述软件系统模型的概念和图形表示法，以及语言的扩展机制和对象约束语言。软件开发人员可以使用 UML 语言对复杂的软件系统建立可视化的系统模型，编制说明和建立软件文档。

UML 支持面向对象的技术和方法，能够准确方便地表达面向对象的概念，体现面向对象的分析与设计风格。

UML 独立于开发过程。UML 可以与 Rational 统一过程配合使用，也可以在其他面向对象的开发过程中使用。UML 独立于程序设计语言，用 UML 建立的软件系统模型可以采用 C++、Java、Smaltalk 以及其他任何一种面向对象的程序设计语言予以实现。

UML 一经推出便得到了许多著名计算机厂商如 IBM、Sun、HP、Oracle、Microsoft 等的欢迎和支持。现在在美国 UML 已得到广泛的使用，在国际上 UML 也正在广泛传播，预计不久的将来，UML 将成为一种软件开发的主流方法与工具。

本章介绍软件工程的一些基础知识和 UML 的概况。

1.1　软件开发方法概述

1.1.1　软件生命周期法

软件生命周期（Software Life Cycle）是指从软件的立项开发到软件的最终消亡的全过程。

软件生命周期经历下列阶段：制定计划、需求分析、软件设计、编码、测试、运行与维护。

在制定计划阶段（Planning），确定系统的目标，提出系统的功能、性能、接口、可靠性、可用性等方面的基本要求，进行系统开发的可行性分析，提出可行性分析报告，制定系统开发的实施计划。

在需求分析阶段（Requirement Analysis），对系统的需求进行详细的分析，并给出明确的定义，编制系统需求分析说明书和初步的用户手册，作为今后系统开发工作的依据。

在软件设计阶段（Software Design），根据系统的需求设计系统的体系结构和软件模块。软件设计一般可再分为两个阶段：概要设计和详细设计。在概要设计中，主要任务是设计软件系统的总体结构，即模块结构，定义每个模块的主要功能和模块之间的联系。在详细设计中，主要任务是进行模块设计，详细定义各模块的数据结构、算法、接口等，作为以后编码工作的依据。如果系统中使用了数据库，则在软件设计阶段还要进行数据库的逻辑设计和物理设计。在软件设计工作完成时应提交软件设计说明书。

在编码阶段（Coding/Programming），主要任务是选择程序设计语言和工具，编写计算机可以接受的软件代码程序，实现系统的各项功能。

在测试阶段（Testing），主要任务是测试软件，排除错误，确保开发得到的软件的功能和性能达到规定的要求，保证软件的质量。软件测试阶段包括两方面的工作：测试和排错（Debug）。测试是发现软件中的错误，排错是找出软件中的错误，并予以纠正。软件测试一般分为3个步骤：单元测试、组装测试和系统测试。在单元测试中，主要任务是对单个模块进行测试，纠正其在功能与结构方面存在的错误。在组装测试中，按一定的顺序把模块组装起来，检测其功能、接口等是否达到要求。在系统测试中，测试已完成的系统软件是否合格，是否满足原设计的各项功能、性能、可靠性等指标，是否能够提交使用。

一般地，软件的测试是与软件的编码结合在一起交叉进行的。

在运行与维护（Running/Maintenance）阶段，首先必须把已开发完成的软件系统，安装到实际的工作环境中试运行，对系统进行考验，发现遗留的问题并予以改进，然后系统才能正式投入运行使用。这称为系统的试航（Piloting）。在系统的运行使用过程中可能会陆续发现一些以前未曾发现的软件错误或缺陷，也可能会发现一些对系统的新的功能需求，也可能软件的运行环境需要变更，于是对系统要进行修改和完善。这个一边运行一边维护的阶段将一直进行到软件系统报废、软件生命终结为止。

传统的瀑布模型（Waterfall Model）是建立在软件生命周期上的一个软件开发的管理模型（参见图1.1），它规定了软件生命周期上各阶段的软件工程活动：制定计划、需求分析、软件设计、编码、测试、运行与维护。各阶段严格按顺序进行，前一阶段的任务没有完成，不能进入下一阶段工作；每一阶段的工作成果需经过评审，评审确认无误，才能作为下一阶段的输入和依据；若经过评审认为有问题而且问题源于前面阶段，则返回到前一阶段甚至更前的阶段重做。

在每一个阶段结束时要求交出规定的软件文档资料，作为评审的依据和下一阶段工作的依据，如计划阶段的可行性分析报告和软件开发计划、需求分析阶段的系统需求分析说明书、设计阶段的系统软件设计说明书、编码阶段的源程序清单、测试阶段的测试报告、运行与维护阶段的软件维护报告等。

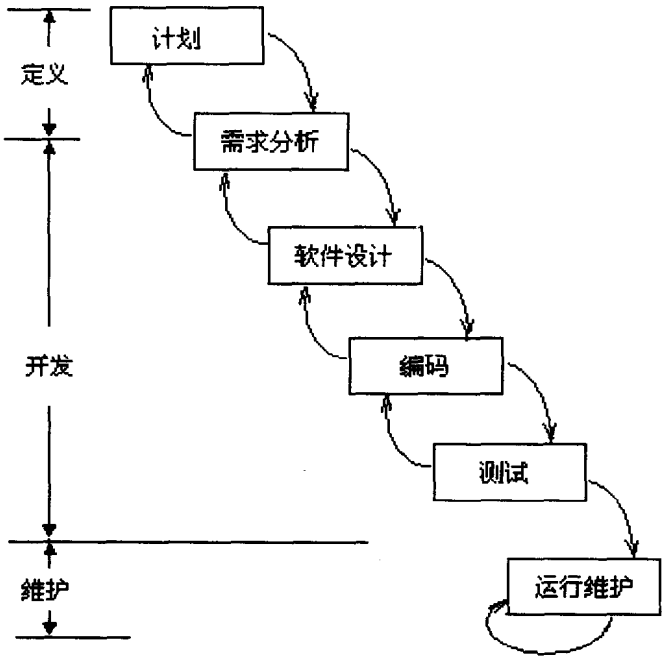

图 1.1　软件生命周期的瀑布模型

软件生命周期法遵循以下的软件设计准则。

1．认识抽象

软件开发的过程就是从现实世界的问题论域向软件系统的映射，其中抽象是认识复杂现象的强有力的思维方法。在建立软件的设计模型时常采用分层抽象的办法。在最高的抽象层次上，可以采用问题的环境语言概括地描述问题的解法；在较低层次上，则采用过程化的方法描述问题的解法。

抽象的内容包括过程抽象、数据抽象、控制抽象等。抽象与模块化相结合，软件生命周期的每一个步骤都是对软件问题解法的抽象层次的一次精化。当软件的源代码编写完成时，即完成了最低层次的抽象，完成了软件系统的开发。

2．模块化

软件模块（Module）是由数据说明、可执行语句等构成的一段程序，它可以实现某个功能。软件模块有唯一的名称，可以通过模块名标识和调用。例如，程序中的 procedure（过程）、function（函数）、subroutine（子程序）、macro（宏）、block（块）等都可以是模块。模块化就是把一个程序划分成许多模块，每个模块实现一个子功能，模块与模块之间有机地关联且相互调用。模块的集合就是程序的整体。

模块化把一个复杂的软件系统的设计问题转化为若干个较小的、容易处理的模块的设计问题，便于分别解决。而且，模块化设计要求模块的独立性强，模块之间的耦合强度弱，能够有效地缩小程序中出现的错误的影响范围，极大地增加了软件的可靠性和软件开发效率。

3．信息隐蔽和局部化

信息隐蔽是指每个模块的实现细节（过程和数据）对于其他模块是隐蔽的和不能访问的。

通过抽象，可以确定构成软件的过程（或信息）实体；通过信息隐蔽，可以定义和实施对模块的过程细节和局部数据结构的存取限制。

信息局部化是指把一些关系密切的软件元素物理地就近安置，例如模块中使用的局部数据结构。局部化有助于实现信息隐蔽。

信息隐蔽和局部化可以使得在测试和维护期间修改软件所发生的错误被局限在一个或少数模块内，不致波及其他部分。

4．结构化程序设计

结构化程序设计采用自顶向下逐步求精的方法，而且只使用以下3种实现单入口和单出口的基本控制结构：顺序、选择、循环。

在系统的总体设计阶段采用自顶向下逐步求精的方法，可以把一个复杂问题的解法分解成一个由模块组成的多层次结构的软件；在系统的详细设计阶段采用自顶向下逐步求精的方法，可以把一个模块的功能逐步分解成一系列的具体处理步骤。

5．软件规范

严格遵守软件规范是软件开发成功的重要保证。当今国际和国内都制订了一系列的软件规范，对软件开发的目标、开发方法、开发过程、软件文档、质量标准等都给出了明确的规定。软件规范是软件生产工业化的标志之一。

软件生命周期的瀑布模型是软件开发的完整的管理方法，它使早期的手工作坊式的软件开发转变为软件工程，在消除非结构化软件、降低软件的复杂度方面起了显著的作用，使软件的开发有一套严格的计划、步骤、规格、方法，保证软件产品达到预期的质量要求，按时交付。因此，软件生命周期的瀑布模型自20世纪70年代以来得到了广泛的传播，而且编入了许多软件工程规范中。

在实践中软件生命周期的瀑布模型暴露出了很多缺点。最主要的问题是它缺乏灵活性，很难面对和处理软件开发中存在的各种风险。瀑布模型按照现代工业的工程化生产模式，规定了严格的软件开发阶段和过程，忽略了软件开发本身的特殊性，过于理想化。例如，一个很突出的问题是关于系统的需求分析。瀑布模型假设对系统的功能需求可以预先严格精确定义，要求系统的需求分析在软件设计前完成，并给出明确的、所有的需求定义，这在实际上往往是做不到的。因为在系统开发之初，无论是领域专家、终端用户还是软件系统分析人员，无论他们如何努力地调查研究与分析，都不可能对未来系统的一切需求都定义得周到详尽、完整无缺。往往是在以后的设计阶段甚至编码阶段，才发现原来对系统的需求定义必须进行某些修改和补充，这样就会导致开发工作的返工，延误软件交付时间，增加资源和费用的投入，降低软件的质量。

1.1.2 原型法

按照传统的瀑布模型先要进行繁琐冗长的需求分析，将系统的需求完全确定下来后再进行软件设计和编码测试，这正是造成系统开发困难的重要原因。一方面系统分析员不可能在系统分析的初期就将所有的用户需求考虑周全，另一方面用户的需求是会变化的，用户对未来系统的认识有一个逐步深化的过程，所以对系统的需求分析不可能在系统开发之前完全确定下来。为了解决这个问题，出现了一种新的软件工程开发方法——原型法（Prototyping）。

原型法开发模型的基本思想如下。

- 首先取得基本的需求，迅速构造一个系统原型。
- 运行原型系统，对原型进行评价，提出修改意见。
- 根据新需求，再实现新一轮的系统原型。
- 重复上述步骤，直到实现满意的最终系统。

原型法与传统的瀑布法不同的地方是在开发的过程中引入了用户评价。系统开发人员首先根据系统的最基本的需求，快速构造出一个原型系统，然后在计算机上运行这个原型系统，系统开发人员与用户一道对这个原型进行评价。在原型法中用户成为了程序开发的参与者，从而使用户提出的需求更加明确和具体，使系统的开发能够沿着正确的方向进行。原型法的工作流程如图 1.2 所示。

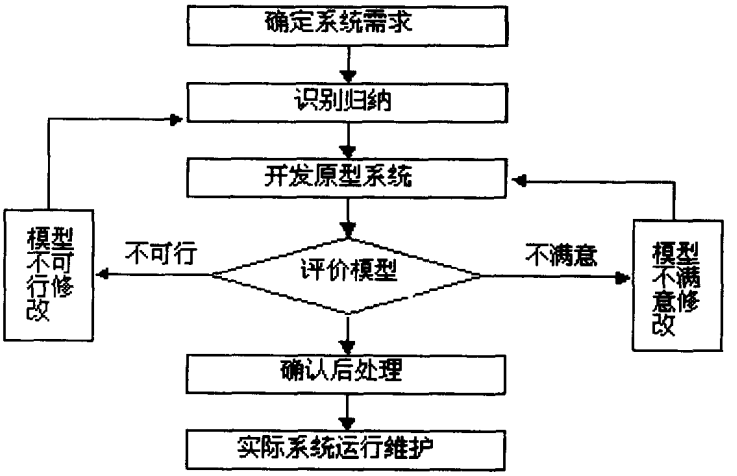

图 1.2　原型法工作流程图

原型是系统的一种简化表示，虽然原型不是完整的最终系统，但它具有最终系统的重要特征。这些特征由原型的用户、所需要的信息和应用领域所决定。

原型的主要特征如下。

1．系统功能

原型只需要实现系统的基本功能。在开发中应抓住问题的重点，简化原型系统，使之能快速地实现。在建立原型时可以运用“二八”法则，即一般情况下在系统的整个生存期中，80%的时间运用的只是系统 20%的功能。以这 20%的功能建立原型系统，将可迅速测试和深入分析未来系统的频繁使用的部分。

2．系统复杂性

原型的内在的系统复杂性可以低于实际系统，以便于测试系统的主要性能指标。例如，一个分布式系统的原型，可以是只在单机上运行，先完成单机上的系统性能测试，暂时回避多机通信问题，减少原型系统的复杂性。

3．用户接口

原型应包括最终系统的用户接口，使用户可以在原型上测试未来系统的人机界面、屏幕格式和操作命令等，以得到用户的反馈意见和肯定。

4．简化的数据结构