# UML 软件建模

周秉锋　编著


北京大学出版社

北　京

## 内 容 提 要

本书结合软件开发实际，循序渐进地全面介绍了统一建模语言（UML：Unifide Modeling Language）的基本概念、实际应用和理论基础。全书共分为17章，分别介绍了：软件产品开发和软件建模的关系、统一建模语言简介、需求分析和用例视图、交互与交互图、结构建模与逻辑视图、类、关系、共用机制（common mechanism）、类图(class diagram)、接口、类型、角色、模型包、实例、对象图、状态机、状态机图和活动图等内容。章节和内容的安排按照软件开发过程的分析、设计、建造的顺序展开，并通过大量的建模实例详细解释了各种UML模型元素的语义、语法和建模原则。

本书既可以用做计算机应用技术专业的研究生课程的教材，也可以供软件产品的分析、设计和开发人员作为软件设计的具有实用价值的参考书。

**图书在版编目（CIP）数据**

UML软件建模/周秉锋编著. —北京：北京大学出版社，2001.11
ISBN 7-301-05299-5
I. U… II. 周… 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076923号

书　　名：UML软件建模
著作责任者：周秉锋
责任编辑：黄庆生
标准书号：ISBN 7-301-05299-5/TP·0620
出 版 者：北京大学出版社
地　　址：北京市海淀区中关村北京大学校内　100871
网　　址：http://cbs.pku.edu.cn
电子信箱：xxjs@pup.pku.edu.cn
印 刷 者：中国科学院印刷厂
发 行 者：北京大学出版社
经 销 者：新华书店
787毫米×1092毫米　16开本　11印张　235千字
2001年11月第1版　2001年11月第1次印刷
定　　价：18元

# 目　录

## 第一篇　引　言



## 第二篇　行为建模基础

**第三篇　结构建模基础**

## 第五篇　行 为 建 模

# 第一篇 引 言

# 第 1 章 软件产品开发和软件建模

## 1.1 为什么要建模

随着计算机科学技术的发展，计算机软件开发的目的从最初的科学研究和为科学研究提供辅助手段转为今天的为用户提供满足其需求的软件产品。这一开发目的的转换，使得今天的计算机软件系统成为名副其实的产品，而非“计算机程序”。

作为一种产品，软件系统具有其他工业行业所提供的产品的一切特征。首先，软件产品的开发者和软件产品的使用者是脱离的。这一点和“计算机程序”是不同的。对于“计算机程序”而言，程序的开发者往往就是使用者。其次，软件产品必须为用户提供他们期望此产品所提供的一切功能，而且这些功能必须是以一种用户满意的方式提供的。这包括软件的运行效率、使用的方便程度、对硬件环境的要求，等等。第三，软件产品具有质量的概念，当用户使用软件产品时，此产品发生故障的概率必须低至用户所能容忍的限度之下。第四，软件系统作为一种产品，应具有完备的用户手册和对应的技术文档，以备用户在使用遇到困难时查阅。

软件系统作为一种产品，其生产过程在生产对它的需求上，具有工业化生产的特点。例如：当前的软件产品是一个复杂的系统，要求由一个开发团队的多名开发人员分工合作，协同工作的才能开发成功；软件产品的生产是需要使用工具的（如：高级语言编译器）；软件产品的市场竞争压力，需要软件所采用的技术是可复用的，以达到软件版本的快速更新的目的；并需要软件生产的质量是稳定的，不应在不同版本的软件之间产生质量不稳定的现象，等等。但是，和其他传统的、成熟的工业生产行业（如：电子产品生产、机械制造业、建筑业）相比，软件生产实际上仍处于一个相当幼稚的阶段，上述对生产过程的需

求，仍未能完全满足。

例如，在电子产品制造业，当需要开发一种产品的时候。首先要确定此产品的性能指标，如：某台收音机有几个波段、有几个扬声器、每个波段的接受频率的范围是多少，扬声系统的频率响应范围是多少、应为使用者提供哪些控制手段（如：调谐系统，音质控制系统），等等。

然后，根据已确定了的性能指标，设计此产品的电原理图。电原理图是电子产品制造业通用的标准制图方法，它用集成电路符号、晶体管符号、电阻电容符号，以及各种标注符号等，描述了整个电子产品的工作原理和产品的电气连接关系。在电原理图上，可以读到构成此产品的各个元器件的标准型号，可以看到各元器件之间的电气连接关系，可以通过标注了解到各关键信号的电气指标，如：信号的频率，标准电平，标准电流，等等。

电原理图是电子产品生产制造的基础。在电原理图设计完成之后，就可以根据电原理图制作实验电路板，生产原型产品，调试，定型，最终投入生产。在进行电子产品的原型机的制造和调试的过程中，还需要绘制电路板的布线图以规范原型机的制造。

当电子产品定型并投入生产后，以上各种产品设计资料会被作为技术资料存档。当产品需要更新换代的时候，可以从这些技术资料找到可以直接利用的成熟技术，并根据新的产品需求和新兴的先进技术，设计生产新一代的产品。

上面描述的是电子产品制造业的一个典型的生产过程，这是一个成熟的、工业化的生产过程，所有其他成熟的工业生产过程都和此过程类似。而在软件生产行业，情形则有所不同。

在软件生产行业，当开发一个软件产品的时候，通常会先写一个系统分析报告，此系统分析报告通常由文字和一些示意性的框图构成。然后，再根据生产者的技术和开发环境，提出一个简单的设计报告，此报告也是由简单文本和非标准图形组成。系统分析报告和软件设计报告完成后，经过一个不是十分严格的评审，并得到通过后，就进入了编码、调试、测试、发行阶段。在大多数情况下，迫于开发和项目计划方面的压力，这时的工作已经和前两步工作脱钩了，开发人员在投入开发活动之后，较少会再查阅系统分析报告和软件设计报告，在调试和测试排错过程中对软件结构所进行的修改也较少会反映在软件设计报告中。

当软件产品开始发行，开发项目结束后，系统分析报告较少存档，即使归档了，也不大具备参考价值，因为此时软件设计报告已经和软件的实际实现脱节，无法反映软件产品的实际原理和结构。当此软件产品需要更新换代时，软件新版本的开发队伍所面对的只是老版本的软件产品本身加上相关的源代码。除此之外，没有任何其他的资源可以帮助他们分析和理解原产品的设计原理、结构和实现思路。

可以看出，软件生产行业的这种状况，相当于在电子产品制造进行收音机的生产时，不绘制电原理图就直接用集成电路、晶体管、电阻、电容制造收音机。这在电子制造业，这是不可想象的，会带来许多严重的问题。例如，产品发货后，如何对故障产品进行维修？

如何在原有产品基础上发展更先进的产品，并使新产品的开发周期和费用最短？等等。对于软件行业，同样会导致这样的问题。这些现象的存在，使得软件的技术复用难以进行，开发队伍无法得到明确有效的分工；软件的质量难以保证，从而严重降低了软件产品的开发效率；并最终束缚了软件生产行业的健康发展。

## 1.2　软件建模及建模的原则

从上面的对比可以看出，和其他成熟的工业生产行业相比，软件生产行业还处在一个比较幼稚的阶段。在其他成熟的工业行业的发展历史中，也存在过一个比较幼稚的“手工作坊”阶段。那时，在产品的设计和制造活动中，非标准设计描述手段大量存在，产品和技术的继承性靠工匠的高超的手艺来保证。因此限制了产品的开发和技术更新的效率。软件生产行业的现状，就和其他工业生产领域的“手工作坊”阶段类似。

分析比较这两个阶段的不同，可以看出，标准而规范的设计描述手段的引入，对生产的发展起着一个重要的作用。它作为一种标准的交流媒介，可以促进开发队伍的分工合作；作为一种通用的描述手段，可以提高软件产品的开发效率和质量。标准而规范的设计描述手段，从本质上讲，就是一种建模的手段。在产品的实际制造之前，先用模型来描述产品的特性和结构，从而使得参与产品设计和制造的人员都能了解目标产品的设计原理、内部结构、制造工艺和流程，并从中找出产品设计和生产过程中的困难和风险所在。

### 1.2.1　模型是对现实世界的简化

模型，简单地讲，就是对现实世界的简化。在成熟的工业生产领域，建模的方法得到了广泛的应用。在电子产品制造业，人们以电原理图描述产品的原理和构成，以布线图规定电路的物理连接。在建筑业，不但用各种平面图表来为建筑工程建模，还会搭建立体化的小比例模型以帮助理解和表达。建模的方法甚至还可用于戏剧和电影行业，用来设计和表达剧情的发展。

在工业生产制造领域，包括软件生产行业，通过建模，可以使得被制造的产品得到更好的理解。它把被描述的软件系统中的人们关心的各个部分，用模型的方式表达出来，以帮助软件产品开发的相关人员对其所开发的系统的行为和结构进行有效的说明（specifying）和视化（visualizing），指导软件系统的建造（constructing），和为所建造的系统进行建档（documentation）。

说明、视化、建造和建档，是人们进行软件建模的四个基本目的。通过说明，可以在设计阶段为软件系统的各个组成部分规定其功能、结构和对外接口。通过视化，可以有

助于软件规格说明的表达和交流。完备定义的软件规格说明还可以进一步通过模型向源代码的映射支持软件系统的建造，从而提高软件系统的开发效率和质量。完备定义的软件模型本身就是反映软件系统的结构和实现原理的重要技术资料，当软件系统开发完成之后，它可以作为技术档案保存，以便后续产品或相关产品能有效地复用其中的成熟技术。

之所以要进行建模，是由于，对于一个复杂的软件系统，人们无法对其整体进行详细而全面的把握。人的大脑的能力是有限的，正是由于这个原因，一个复杂的软件系统必须由多个开发人员共同完成。对于开发队伍里的每个成员而言，他（她）只能对他（她）所负责的那部分子系统进行详细而全面的把握，而不可能对整个软件系统有全面而详细的了解。而对于一个软件产品而言，如果对整个系统的功能、原理和结构没有一个全面而详细的记载的话，将会对此软件产品的开发、维护、升级产生较为不利的结果。通过建模，可以把一个复杂的系统，按问题的不同方面，以一种约定好的、为大家共同接受的描述方式，分别进行描述。这样，人们在试图理解一个系统时，可根据他（她）所关心的某一方面的问题，查阅对应的系统模型，从而得到对此问题的理解。

### 1.2.2　建模的原则

为了准确而全面地描述软件系统，人们使用了建模的方法。在建模的过程中，必须遵循以下四条原则：

（1）准确的原则：模型必须准确地反映软件系统的真实情况。如前所述，在上面所述的软件开发方法中，实际上已经进行了一定程度的建模工作（系统分析报告和软件设计报告），但由于缺乏规范而有效的建模手段，最初的设计和最终的产品产生了分离，使得模型没能真实的反映系统的真实情况，从而使得模型失去了应有的价值。模型必须准确，意味着在软件开发的整个周期内，模型必须和产品始终保持一致。

（2）分层的原则：在建模的过程中，必须有不同的模型，以不同的抽象程度，反映系统的不同侧面。正如在设计电子产品时，必须绘制电原理图和布线图已分别反映产品的电路原理和物理联线一样。在软件构筑的不同阶段，不同的开发相关人员（stockholder），如：投资者、管理者、设计者、程序员、测试者，使用者，看待软件的侧重面有所不同。因此，软件系统的建模需要不同的模型以反映系统的不同侧面。如，可以用一类模型描绘系统的外部边界和行为。用另一类模型描绘系统的内部逻辑关系。

（3）分治的原则：不可能单独用一个模型来反映整个系统的任何侧面。软件系统是复杂的，对于软件模型的任意一个侧面，不可能用一个模型来反映所有内容，因此，需要把问题分解为不同的子模型，分别处理。这些模型相对独立，但又互相联系，综合起来构成了此侧面的一个完整的模型。

（4）标准的原则：模型必须在某种程度上是通用的。建模的一个基本目的就是交流，在一个开发队伍内部，不同的开发人员之间需要交流；同一软件的不同时期的版本的开发

队伍，需要参考以前版本的开发队伍的设计原理和实现方案。不同软件的开发队伍之间也需要交流以实现最大程度的软件复用，从而提高开发效率。如果各开发队伍和开发人员在建模的时候采用同样的方法和符号，这样的交流才会高效地进行。否则，交流的时候就会发生困难，甚至失败。

# 第 2 章　统一建模语言简介

## 2.1　UML 和软件体系结构

建模是软件生产工业化的重要手段。软件的建模必须满足准确、分层、分治、标准四个原则。统一建模语言 UML 就是满足这四个原则的建模语言。UML 是用于描绘软件蓝图的标准语言。它可用于对软件密集型系统进行视化、说明、建造和建档。

人们在生产一个软件产品的时候，首先要对此产品进行说明，这包括对软件的市场需求的说明，软件的功能的说明，软件运行原理的说明，软件内部结构和对外接口的说明，软件实现方式的说明，等等。通过建模，可以使这些说明准确化、完整化、标准化。UML 就是一个表达这些说明的标准工具。

在与软件产品生产活动中，存在着大量的交流和沟通活动。人们通过交流和沟通确定软件产品的原理、结构和其中的复杂关系。软件产品的这些内容往往超出了纯文本所能准确有效表达的能力。必须通过模型，把这些内容以一种图示化的方式表达出来。几乎所有工业建模方法都使用图示化的方法。例如：电子产品制造业的电原理图，建筑业的小比例实体模型。在软件生产行业，软件模型的图示化，或者说，视化，则是由 UML 提供的。UML 以一种明确定义的标准图符，描述软件产品生产活动中需要交流的各种内容。使得软件开发的所有相关人员，都能通过这种图示语言，了解和表达需要交流的内容。

建模是为软件产品的生产服务的。因此，如果模型所包含的信息足够完备，就可以以这些信息为基础，进行软件产品的建造。许多工业产品的建模方式都可以直接指导产品的生产。例如：在机械制造行业，生产车间可以按照机械图的规定生产出合格的成品。如果使用了 CAD/CAM 技术，存储在计算机内的机械产品的模型信息，可以直接控制数控加工设备，进行机械产品的加工。软件产品的建模，也应达到这样的目标。UML 就是能达到这一目标的建模语言。用 UML 表达的软件模型，可以直接和某种设计语言建立映射关系，通过 UML 建模工具，将 UML 模型转换为对应的程序设计语言源代码框架。

一个运行管理良好的软件产品生产企业，其生产活动产生的软件制成品（artifact）不应仅仅是软件的可执行代码等最终发行的产品，它还应包括在开发活动中产生的用于控制、评测（measuring）、交流的各种中间产物。如：需求分析报告、软件体系结构

（architecture）、设计报告、源代码、项目计划、测试计划、原型产品、发行版本等等。这些中间产物对于软件企业的健康发展，都是十分重要的，必须以档案的形式保存起来。好的软件模型应能支持这些软件制成品的建模和建档。在 UML 里，存在着标准的描述手段来表达这些内容，如：可以用用例视图来描述需求分析、用逻辑视图来描述软件的结构设计，等等。

作为一种语言，UML 定义了一系列的图符来描述软件密集型的系统。这些图符有严格的语义和清晰的语法。这些图符及其背后的语义和语法，组成了一个标准，使得软件开发的所有相关人员都能用它来对软件系统的各个侧面进行描述。

如前所述，分层是软件建模的重要原则之一。为了表达不同的软件开发相关人员在软件开发周期的不同时期看待软件产品的不同侧重面，需要对模型进行分层。在 UML 中，采用了根据软件产品的体系结构（architecture）进行分层的方法。

软件的体系结构是软件产品构筑过程中的一项重要的软件制成品。软件体系结构由一系列的决定组成，这些决定定义了如下内容：

（1）软件系统的组织；

（2）构成软件系统的结构元素的结构及它们之间的接口；

（3）结构元素的行为及元素之间的协同（collaboration）；

（4）结构元素的不断组合以构成日渐完备的子系统的过程；

（5）指导软件建造过程的软件构筑风格（architectural style）：静态和动态元素之间的接口、协同和构成（composition）。

软件体系结构不仅仅决定软件的结构和行为，而且还决定着软件的用途、功能、性能、应变性（resilience）、可重用性、经济和技术方面的限制和折衷以及美学考虑（aesthetic concern）。

UML 将软件的体系结构分解为五个不同的侧面（如图 2.1 所示），称为视图（view）。它们分别是：用例视图（Use case view）、设计视图（design view）、进程视图（process view）、实现视图（implementation view）、和分布视图（deployment view）。设计视图和进程视图又可被统一称为逻辑视图（logical view）。每一个视图分别关注软件开发的某一侧面，它们由一种或多种模型图（diagram）构成。模型图描述了构成相应视图的基本模型元素（element）及它们之间的相互关系。

（1）用例视图（use case view）：用例视图用来支持软件系统的需求分析，它定义系统的边界，关注的是系统的外部功能的描述。它从系统的使用者的角度，描述系统外部的动态行为和静态的功能组合。系统的动态功能由 UML 里的交互图（interaction diagram）、状态图（state-chart diagram）、和活动图（activity diagram）构成。

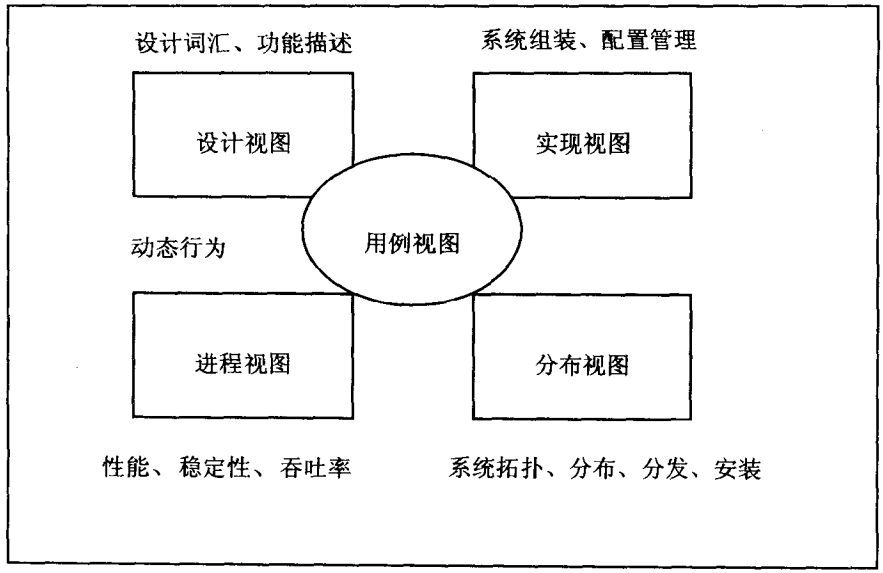


**图 2.1　为软件的体系结构建模**

（2）逻辑视图（Logical View）：逻辑视图定义系统的实现逻辑。它描述了为实现用例视图描述的功能，在对软件系统进行设计时，所产生的设计概念，设计概念又称为软件系统的设计词汇（vocabulary）。逻辑视图定义了设计词汇的逻辑结构和存在于它们之间的语义联系。设计词汇包括系统的类、协同、接口及其关系。对逻辑视图的描述在原则上与软件系统的实现平台无关。它相当于电子产品生产中的电原理图。逻辑视图包含的模型图有：类图（class diagrams）、对象图（object diagrams）、交互图（interaction diagrams）、状态图（state-chart diagrams）、活动图（activity diagram diagrams）。

（3）实现视图（implementation view）：当系统的逻辑结构在逻辑视图里被定义之后，需要定义逻辑结构的物理实现。这包括：用什么文件保存相应的设计元素的源代码，各物理文件之间的关系，存放路径，等等。实现视图就是定义这些内容的地方。它相当于电子产品的印刷电路板的布线图。实现视图描述的是组成一个软件系统的各个物理部件，这些部件以各种方式（如：不同的源代码经过编译，构成一个可执行系统；或者不同的软件组件配置成为一个可执行系统；以及不同的网页文件，以特定的目录结构，组成一个网站，等等）组合起来，构成了一个可实际运行的系统。实现视图包含的模型图有：部件图（Component diagram）、交互图、状态图、活动图。

（4）分布视图（Deployment View）：当一个软件产品被安装以后，它将运行在计算机硬件系统上，如果软件产品是面向网络的应用系统，则有可能同时运行在多个计算机上。分布视图用来描述软件产品在计算机硬件系统和网络上的安装、分发（delivery）和分布