教育部职业教育与成人教育司推荐教材

软件开发流程实训

主编 李政

中国财政经济出版社
China Financial & Economic Publishing House

教育部职业教育与成人教育司推荐教材

1	《中文Premiere Pro视音频编辑教程》	方　洱
2	《二维动画制作》	刘福生
3	《数据库应用与开发》	何代菊
4	《计算机多媒体实用技术》	栾元敏
5	《动画设计综合实训》	刘益红
6	《影视制作综合实训》	陈渤英
7	《平面设计综合实训》	乔朝阳、张　强
8	《计算机组装与维修》	刘新生
9	《操作系统与网络服务器的使用与管理》	陈德伍、钟一兵
10	《桌面编程之Visual Basic》	徐国强
11	《网页制作》	许多顶
12	《计算机操作与使用》	刘太安、包忠明
13	《三维动画制作3ds max6基础教程》	贺　鑫
14	《网络布线与小型局域网搭建》	朱宪花、郑金刚
15	《中小型网站的建设与管理》	李洪生
16	《计算机图形图像处理》	杜立东
17	《多媒体美术》	王正斌
18	《Internet应用技术》	田　青
19	《软件开发流程实训》	李　政
20	《基于过程的程序设计——C语言版》	许成云
21	《计算机网络技术及应用》	孙开绪
22	《多媒体演示软件制作综合实训》	孙万军

ISBN 7-5005-8411-3
9 787500 584117 >

中国财政经济出版社
教育分社
查询网站 www.csdn315.com.cn
查询电话 10106000　查询短信 95889

ISBN7-5005-8411-3/TP·0100　定价：11.00 元

教育部职业教育与成人教育司推荐教材

jiaoyubuzhiyejiaoyuyuchengrenjiaoyusituijianjiaocai

软件开发流程实训

主编　李　政
审稿　顾　浩　许多顶

中国财政经济出版社

图书在版编目（CIP）数据

软件开发流程实训/李政主编. —北京：中国财政经济出版社，2005.7
教育部职业教育与成人教育司推荐教材
ISBN 7-5005-8411-3

Ⅰ.软… Ⅱ.李… Ⅲ.软件开发-成人教育：高等教育-教材 Ⅳ.TP311.52

中国版本图书馆 CIP 数据核字（2005）第 076784 号

中国财政经济出版社 出版
URL：http：//www.cfeph.cn
E-mail：cfeph@cfeph.cn

社址：北京市海淀区阜成路甲 28 号 邮政编码：100036
发行电话：88190616 传真：88190655
北京财经印刷厂印刷 各地新华书店经销
787×1092 毫米 16 开 8.75 印张 203 000 字
2005 年 9 月第 1 版 2005 年 9 月北京第 1 次印刷
定价：11.00 元
ISBN 7-5005-8411-3/TP·0100
（图书出现印装问题，本社负责调换）

出 版 说 明

为了进一步贯彻落实《国务院关于大力推进职业教育改革与发展的决定》和全国职业教育工作会议的精神，适应中等职业教育发展的趋势，满足各类职业技术学校培养技能型紧缺人才的实际需要，我们组织编写了中等职业教育计算机应用与软件技术专业教学用书。从2005年秋季开学起，这些教材将陆续提供给各类职业技术学校使用。

该系列教材是根据教育部提出的“以综合素质培养为基础，以能力培养为主线”为指导思想，以教育部新近颁布的计算机应用与软件技术专业教学指导方案为依据，结合中等职业教育的教学培养目标而编写的，经教育部职业教育与成人教育司批准立项，并由专家审定，作为教育部职业教育与成人教育司推荐教材出版。新教材全面贯彻素质教育思想，从社会发展对高技术应用性人才的需求出发，在内容的构建上结合专业岗位（群）对职业能力的需要来确定教材的知识点、技能点和素质要求点，并注重新知识、新技术、新工艺、新方法的应用，注重对学生的创新精神和实践能力的培养。新教材在理论体系、组织结构和阐述方法等方面均作了一些新的尝试，以适应中等职业教育教学改革，满足各类中等职业技术学校的教学需要。在此，我们真诚的希望各类职业技术学校在教材的使用过程中，能够总结经验，及时提出修改意见和建议，使之不断完善和提高。

2005年6月

前　言

目前，在我国职业教育中，有些课程的内容偏重理论知识的学习，而实际技能的训练相对不足，课程的内容滞后于专业技术的更新与发展，有关案例教学和项目教学的内容极少，导致学生在实际工作中分析问题和解决问题的能力较弱。

根据教育部办公厅、信息产业部办公厅有关文件精神，我国职业教育应该把培养学生的职业能力放在突出位置，加强实践性的教学环节，把学生培养成为企业生产服务一线迫切需要的高素质劳动者。为此，我们编写了这本适合中职学生学习的“软件开发”方面的教材。其主要内容包括软件开发的基本概念、软件开发的分析阶段、设计阶段、编写阶段、程序美学、逐步求精以及软件开发的测试阶段。

本教材力求做到体现“加强实践性教学环节”这一要求，增强教材的趣味性，给学生更多的动手机会，激发其学习的主动性。

本教材每个章节都是采用知识引入、案例分析的方法编写，让学生学会如何把学到的理论知识应用到实际例子中。本教材充分考虑到学习对象，尽量使用通俗易懂的语言来表达深奥的理论知识，采用图文并茂的手法，增强教材的趣味性和直观性。本教材中的大量实例采用 C 语言编写。

由于水平有限，时间紧迫，书中错误在所难免，敬请各位读者批评指正。

本教材由李政主编。第 1 章由李政编写；第 2 章、第 3 章由张赞编写；第 4 章由刘志强编写；第 5 章由姜秋明编写；第 6 章由雷卫编写；第 7 章由李晓方编写。

编　者

2005 年 4 月

目　录

第 1 章

软件开发的基本概念

内容提要

软件开发是软件生命期的重要时期。本章主要介绍软件开发中的基本概念，主要包括分析阶段、设计阶段、编写阶段和测试阶段的基本概念。

第 1 节 软件、软件生命期及软件危机

软件、软件生命期和软件危机是软件工程学中的重要概念。

在机械工程中，伴随着一台机器的设计制造，设计人员需要及时、认真地建立一整套图纸资料，例如结构图、零件图、使用说明书、维护手册等。由于以下一些原因，图纸资料是研制、使用和维护机器所必需的。

- 人的记忆力是有限的，所以各种技术细节必须以书面形式记录下来。
- 研制人员之间要借助于图纸进行交流。
- 后阶段的工作要在前阶段建立图纸资料的基础上继续进行。

所以在机械工程中，最终产品是机器以及一整套有关的图纸资料，图纸资料是否齐全直接影响着产品的质量。

同样，在软件工程中，为了保证产品的质量，我们需要为“软件”这个词给出一个定义。

20 世纪 80 年代初，软件还只是在高等院校和科研机构中被认识，那时提到软件，一般是指源程序。

目前对软件比较公认的解释为：软件是程序、支持程序运行的数据以及与程序有关的文档资料的完整集合。其中，文档是与程序开发、维护和使用有关的图文资料。这个定义强

调，在研制过程中，及时地按一定规格产生各种文档是研制工作的有机组成部分，必须充分地重视。

既然“软件”不再仅仅是指程序了，那么“研制软件”也就不仅是“编程序”了，它的含义也需要相应地扩充，这就需要说明“软件生命期”这个概念。

在机械工程中，一台机器的生命期（即从开始研制到机器被废弃不再使用为止）要经过分析要求、设计、制造、测试、运行（此时需要不断地维护）等几个阶段。同样，为了用工程化的方式有效地管理研制软件的全过程，软件系统的生命期也可以分为 5 个阶段：分析、设计、编写、测试、运行。

其中，前 4个阶段又总称为“开发期”，最后一个阶段称为“运行期”。

软件生命期中，每个阶段都有确定的任务，并产生一定规格的文档送给下一个阶段，下一个阶段在前阶段提供的文档的基础上继续工作。表 1－1 大致列出了每个阶段的基本任务、工作结果和参加人员。

表 1－1　　软 件 生 命 期

阶　段		基本任务	工作结果	参加者
开发期	分析	理解和表达用户的要求	系统说明书	用户、程序员
	设计	建立系统结构	模块说明书、数据说明	高级程序员
	编写	写程序	程序	程序员
	测试	发现错误和排除错误	可运行的系统	另一个独立的部门
运行期	运行	维护	改进的系统	

软件具有如下特点。

➢ 软件是一种逻辑实体，具有抽象性。这个特点使它与其他工程对象有着明显的差异。人们可以把它记录在纸上、内存、磁盘、光盘上，但却无法看到软件本身的形态，必须通过观察、分析、思考、判断，才能了解它的功能和性能。

➢ 因为软件是一种逻辑实体，所以软件在使用过程中，没有磨损、老化的问题。软件在使用过程中不会因为磨损而老化，但为了适应硬件、系统环境以及需求的变化，可能要不断地修改，这些修改不可避免地会引入错误，导致软件失效的概率升高，从而使得软件的可靠性下降。当修改的成本难以接受时，软件就被抛弃。

➢ 软件一旦研制开发成功，其生产过程就变成复制过程。不像其他工程产品那样有明显的生产制造过程。由于软件复制非常容易，因此就出现了软件产品的版权保护和打击盗版的问题。

➢ 软件对硬件和环境有着不同的依赖性，这导致了软件升级和移植的问题。随着计算机技术的发展，计算机硬件和支撑环境不断升级，为了适应运行环境的变化，软件也需要不断维护，并且其维护成本通常比开发成本高出许多。

➢ 软件生产至今尚未摆脱手工方式。随着软件开发环境的改善，市场上出现了一些辅助开发工具，可以辅助生成代码框架、生成开发文档等。但是最终的核心代码仍然要程序手工编写和组织。并且，随着科学技术的进步，人们对计算机的依赖程度越来越高，因此对软件的需求数量和规模也更大，导致软件开发人员的工作压力也越来越大。

➢ 软件开发的手工行为导致了一个致命的错误，就是为应用“量身订做”软件。其他工程领域有产品标准，所有生产厂家按照标准生产产品，客户买来就可以使用。例如，不管哪个厂家生产的灯泡，只要瓦数、电压、电流、接口等几个指标符合要求，用户买来装上就可以使用。而长期以来，软件给人的感觉是很简单地修改几条指令。因此，客户总是强调软件要适应自己的业务要求。因此，大多数软件产品是为客户“定做”的，通用性差。

➢ 软件涉及人类社会的各个行业，常常涉及其他领域的专业知识，这对软件设计人员提出了很高的要求。软件开发实际上应该有比较明确的分工，例如，业务（咨询）专家负责规范、描述用户的业务流程；系统分析员负责将用户的需求转换为系统功能和性能的需求；系统设计人员负责规划系统框架、构建系统模型、设计系统蓝图；软件工程师负责实施设计方案；测试工程师负责软件测试，发现软件的缺陷；质量工程师负责制定软件质量标准、监督软件开发过程、评估软件质量等。实际上，软件开发过程中，不同的工作应该有不同人员专门负责实施，然而，实际工作中，我们却经常看到一个项目中几个软件人员从头做到尾，既做系统分析又做系统设计，最后还编写代码。

➢ 软件不仅是一种市场上推销的工业产品，它往往还是与文学艺术作品相似的精神作品，与体力劳动相比，精神活动过程的特点是“不可见性”，这大大增加了组织管理上的困难。

由于软件的这些特点，以及长期以来始终没有发明一种高效的开发方法，导致软件生产效率非常低，交付期一拖再拖，最终交付的软件产品在质量上很难保障。特别是软件对开发者的依赖性非常强，开发队伍中一旦走掉一个主力，很有可能使整个产品处于停止状态。这种现象就被定义为“软件危机”，它具体表现如下。

➢ 对软件开发的成本估计不正确，造成开发成本超出预算。

➢ 开发进度不能保证，交付时间一再拖延。

➢ “已完成”的软件不能满足用户的需求。

➢ 软件产品的质量没有保证，运算结果出错、操作死机等现象屡屡出现。

➢ 软件通常没有适当的文档资料，或文档与最终交付的软件产品不符合，软件的可维护程度非常低。

➢ 软件开发生产率的提高赶不上硬件的发展和人们需求的增长。

第2节

软件的分类

软件的分类有许多方法，不同人员由于不同目的，可能有不同的划分原则。通常的划分方法是按软件的功能和规模划分。此外，软件还可以按重要性划分为关键软件和非关键性软件。按处理方式可划分为实时软件、交互软件、批处理软件。按销售市场可划分为项目软件、产品软件。按使用频率可划分为高使用频率软件和低使用频率软件等。不同类型的软件开发要求不同，所遵循的开发标准也不同。

一、按软件功能划分

按照软件的功能可以将软件划分为系统软件、支撑软件、应用软件。

1. 系统软件

通常是与计算机硬件密切相关的那些比较底层的支持软件。这些软件的规模通常比较大，并且与硬件的结构和性能密切相关。例如，操作系统、设备驱动程序、网络通信软件等。它们的作用是保障计算机各个部件能够正常运行，使相关的软件和数据协调、高效地工作。这部分软件在任何应用中都是必不可少的，也是首先要确定的软件。只有确定了系统软件的类型和版本后，才能考虑支撑软件和应用软件。

2. 支撑软件

支撑软件是支持软件开发和运行的工具性软件。其中包括数据库管理系统、软件开发环境、软件辅助设计工具、软件辅助测试工具、程序库等，这类软件非常多，分类也更加细致。

3. 应用软件

应用软件是为特定应用目的而开发、提供某些特定服务的软件。应用软件可谓是规模各异，种类繁多。不同的领域有不同的应用软件。既有大规模的应用软件，例如字处理软件、计算机辅助设计与制造软件、军事指挥系统、导弹防御系统，又有微型软件，例如，只有几条指令的微型控制软件。

二、按软件规模划分

根据软件开发所投入的人力和时间等资源，以及软件交付的文档和源程序的数量，软件可划分为微型软件、小型软件、中型软件、大型软件、超大型软件和巨型软件。具体划分标准见表 1－2。

表 1－2　　软　件　规模

软件规模	投入人数	开发时间	源程序行数	特　征
微型	1	1～4 周	500	不必有严格的设计和测试文档
小型	1～2	1～6 月	2000	通常没有与其他程序的接口
中型	3～5	1～2 年	0.5～5 万	需要有严格的文档和设计规范
大型	5～20	2～3 年	5～10 万	需要按照软件工程方法进行管理
超大型	100～1000	4～5 年	100 万左右	必须按照软件工程开发，有严格的质量管理措施
巨型	2000～5000	5～10 年	1000 万左右	同上

通常规模大、投入人员多、开发周期长的软件，其开发工作必须严格按照软件工程方法进行。而规模小的软件由于投入有限，不可能完全按照软件工程方法实施，需要开发人员根据具体情况调整软件工程的部分条款。

第3节

软件开发过程模型

软件开发过程模型反映的是从软件需求定义到软件交付使用后报废为止，在这整个生命期中的系统开发、运行、维护所实施的全部策略。到目前为止，已经提出了多种模型，主要有瀑布型、原型模型、螺旋模型、智能模型等。模型的选择是基于软件的特点和应用领域。下面介绍几种典型的模型。

一、瀑布模型

瀑布模型规定了软件生命周期的各项活动：问题定义、需求分析、软件设计、编码、测试、运行和维护。各项活动自顶向下、相互衔接如同瀑布一样。

瀑布模型广为流传，它的配合结构化方法和严格的软件开发管理手段，在软件工程化开发中起到了重要的作用。但是，通过长期的实践活动，人们发现，这种模型应付需求变化的能力非常弱。在项目刚刚开始时，系统分析人员和用户很难完全描述清楚新系统的需求。特别是用户日常的一些工作，在他们看来已经是习以为常的活动，常常被无意地忽略，而系统分析人员通常不是用户业务领域的专家，他们也不知道这些活动。直到开发人员按照需求文档实现了系统后，用户发现不符合业务要求，但为时已晚。因为这是对系统作修改，不但造成开发成本高、交付期延迟，最为关键的是会大幅度地降低软件的质量。有一个例子很能说明问题，一个客户要求汽车设计师为他订制一辆高级轿车，客户讲述了对轿车的总体要求，然后设计师开始设计，设计好后，拿着图纸给客户，客户说：“我对图纸一窍不通，只要你按照我的要求设计就可以了。”当设计好的轿车制造出来后，客户说：“我想在车的后面加一个高位刹车灯。”设计师则说：“这是不可能的，你知道把这辆造好的车拆开需要多少钱吗？拆开之后要修改电路、要重新布局和测试，这个工作量太大了。”开发软件与这种情况非常相似，许多用户在没有看到开发好的系统之前，对自己到底需要什么样的软件并没有一个完整的概念，当系统开发出来之后，他会提出许多合理的、非常好的意见。可是重新设计、编码和测试的工作量通常令开发商们难以承受。遇到这种情况时，开发人员和业务人员常常因此而造成不愉快，严重时还会给双方造成巨大的经济损失。

二、原型模型

事实证明，一旦用户开始实际使用了为他开发的系统，便会对系统的功能、界面、操作方式等提出许多建议，而且这些建议通常都非常合理，让人无法拒绝。为此，经过长期的实践总结，人们提出了原型模型。

原型模型的基本思想是：在与用户进行需求分析的同时，以比较小的代价建立一个能够反映用户主要需求的原型系统。用户在原型上提出改进意见，分析人员根据用户的意见，补充完善原型，然后再由用户评价，提出意见，如此往复，直到开发的原型系统满足了用户的

需求为止。通常原型系统是用户可以操作的系统，系统中已经包括了用户的需求，用户通过实际操作，比较容易发现漏掉的需求。

原型模型包括抛弃型和演化型两种。

三、螺旋模型

螺旋模型是一种特殊的原型方法，适用于规模较大的复杂系统。螺旋模型的基本思想是这样的：沿螺旋线自内向外，每一圈开发一个新的软件版本。首先，与用户交流确定软件的部分需求，根据需求制定计划和实施方案，分析方案的风险，然后开发实施、测试、集成，对用户进行培训，听取用户评价，并提出修改意见，到此完成了螺旋最内层的部分，相当于系统的一个完整子集。根据用户提出的建议，进入螺旋的第二层，又与用户交谈，制定新的计划和实施方案，然后再一次分析实施风险，如果风险太大，项目可以就此终止，否则进入实施部分，最后用户评价这一轮的实施结果，并提出修改建议。然后进入第三层螺旋……如此下去，最终用户获得完整的系统。在风险评估和实施之间，一般要建立原型，这样最外层螺旋对应的原型是可运行的系统。

螺旋模型的主要优势在于它是风险驱动的，每个方案在实施前都要经过风险分析。如果风险过大，则项目应该停止，或改变方案。但是，当开发人员水平比较低、不能准确地识别风险时，可能就会出现这样的现象：实际上软件项目已经危机四伏了，但开发组还认为一切良好。因此，在选择使用螺旋模型时，要确定开发组的成员是否具有判断风险的能力。

第4节 软件开发方法简述

前面已经列举了软件危机的现象以及产生软件危机的原因，为了克服软件危机，软件工程研究人员不断探索新的软件开发方法。下面介绍几种软件开发方法。

一、Parnas方法

D.L.Parnas在1972年提出了Parnas方法，这个方法主要有两个亮点：信息隐蔽技术和错误预防措施。

信息隐蔽技术的主要内容是：在概要设计时列出可能会发生变化的因素，并在模块划分时将这些因素放到个别模块内部。一旦由于这些因素变化需要修改软件时，只需修改相应的个别模块，其他模块不受影响。信息隐蔽技术的主要目的是提高软件的可维护性，减少模块之间的相互影响，提高软件的可靠性。

预防错误的主要内容是：在每个可能产生的错误之前，增加一些判断，防止软件出现不可预料的结果。例如，在分配和使用设备前，应该查看设备状态，检查设备是否可用，如果可用再分配。模块之间也要加强错误隔离措施，防止错误蔓延。

令人遗憾的是，Parnas方法的使用没有明确的工作流程，所以这一方法不能独立使用，

只能作为其他方法的补充。

二、Yourdon 方法

1978年，E.Yourdon和 L.Constantine 提出了 Yourdon 方法，即 SA/SD方法，也可称为结构化方法和面向功能的软件开发方法。

Yourdon方法是20世纪80年代使用最广泛的软件开发方法。它首先用结构化分析技术对软件进行需求分析，然后用结构化设计技术进行总体设计和详细设计，最后是结构化编程。这一方法的精髓是自顶向下、逐步求精，也就是将功能逐步分解，直到人们可以理解和控制为止。Yourdon方法的主要问题是以功能分解的软件系统不够稳定，用户的功能经常变换，导致系统的框架结构不稳定。

但是，无论如何，这个方法的应用是非常普遍的，至今仍然有许多机构在使用 Yourdon 方法，或根据机构的具体情况改进 Yourdon 方法。我们将在以后的学习中，重点学习这个方法。

三、面向数据结构方法

面向数据结构的软件开发方法有两种，一种是 Warnier是方法，J.D.Warnier于 1974年提出，另一种 Jackson是方法，M.A.Jackson于 1975年提出。其基本思想是：从目标系统的输入／输出数据结构入手，导出程序的框架结构，再补充其他细节，得到完整的程序结构图。这两种方法的差别有 3点：第一个差别是他们使用的图形工具不同，分别使用 Warnier 图和 Jackson 图；第二个差别是使用的伪码不同；第三个差别是最主要的差别，即在构造程序的框架时，Warnier方法仅考虑输入数据结构，而 Jackson方法不仅考虑输入数据结构，而且还考虑输出数据结构。 Jackson方法适合对中小型软件进行详细设计。因为 Jackson方法无法构架软件系统的整体框架结构，所以不适合进行概要设计。

面向数据结构的方法可以看成是结构化方法与面向对象方法之间的一种方法。它是以数据结构为研究的起点，针对不同的数据结构分配相应的处理，只是处理上没有将数据结构与处理算法封装在一起。

四、问题分析法

问题分析法（Problem Analysis Method，PAM）是20世纪80年代末由日立（HITACHI）公司提出的一种软件开发方法。它的基本思想是由输入、输出数据结构指导系统的分解，然后站在整个系统的角度再逐步综合。这一方法的具体步骤是：从输入、输出数据结构导出基本处理框；分析这些处理框之间的先后关系；按先后关系逐步综合处理框，直到画出整个系统的 PAD（Problem Analysis Diagram 图。 PAD 图是一种二维树形结构图。 PAM 方法是一种良好的详细设计方法，在日本较为流行，软件开发的成功率也很高。但由于在输入、输出数据结构与整个系统之间存在着鸿沟，所以，这一方法仍只适用于中小型软件的详细设计。从上述步骤中可以看出，这一方法本质上是综合的自底向上的方法，但在逐步综合之前已充分考虑系统的输入、输出数据结构。

五、面向对象的方法

面向对象的软件开发方法的研究开始于 1966 年，当时 Kisten Nygaard 和 Ole - Johan Dahl 开发了具有更高级抽象机制的 Simula 语言，Simula语言提供了比子程序更高一级的抽象和封装。大约在同一时期， Alan Kay 正在进行图形化开发模拟，由于软硬件的限制， Alan 的尝试没有成功。 20 世纪 70 年代初期， Alan Kay 加入了 Palo Alto 研究中心（PARC），他引用了 Simula 中关于类的概念，开发出 Smalltalk 语言。1972 年，PARC 发布了 Smalltalk 的第一个版本，同时，“面向对象”这一术语正式确定。 Smalltalk 被认为是第一种真正的面向对象的语言。在 Smalltalk 中，一切都是对象。 1981 年 8 月的 BYTE 杂志公布了 Smalltalk 开发组的研究结果，Smalltalk 关于开发环境的研究导致了后来的一系列进展：窗口（Window）、图标（Icon）、鼠标（Mouse）环境。之后出现了许多面向对象语言，自 20 世纪 80 年代到 90 年代初，面向对象研究从编程语言向着软件生命期的前阶段发展。也就是说，人们对面向对象方法的研究与运用，不再局限于编程语言，而是从系统分析和系统设计阶段就开始采用面向对象方法。这标志着面向对象方法已经发展成一种完整的方法论和系统化的技术体系。

20世纪 90 年代以来，面向对象的分析、设计、测试、度量和管理等研究都得到了发展。目前，面向对象技术的前沿课题包括面向对象设计模式、分布式对象系统和基于网络的对象应用等。

面向对象的开发强调从实际问题域到软件程序和界面的直接映射，这更符合人类的自然思维方式。它的基本做法是用对象模拟实际问题域中的实体，以对象间的关系刻画实体间的联系。因为面向对象的软件系统结构是根据实际问题域的模型建立起来的，而不是基于对功能的分解，所以，当系统功能发生变化时，不会引起软件结构的整体变化，往往只需要进行一些局部的修改。反之，传统的软件开发方法以算法为核心，所建立的软件系统结构与系统功能密切相关，当功能发生变化时，整个软件结构就都动摇了。

面向对象的方法从问题模型开始，然后识别对象、不断细化过程。它从本质上讲是迭代和渐增的，整个开发过程就是一次次迭代反复的过程，随着迭代的范围扩大，系统不断完善。在这里，传统开发模式中的分析、设计和编码各个阶段之间的界限变得模糊起来，其原因是对象的概念贯穿了整个软件生存期。对象成为分析、设计和编码各阶段的共同表达媒介。开发的重心从以功能为中心向以数据为中心偏移。

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，他将以前针对 Ada 语言所做的工作扩展到整个面向对象设计领域。他发明了 Booch 方法，其中 Booch 93 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（Object - Oriented Modeling Technique，OMT）。这种方法用对象模型、动态模型、功能模型，共同完成对整个系统的建模，除了采用面向对象的概念外，还引入了独立于语言的符号集，所定义的概念和符号可用于软件分析、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（Object - Oriented Software Engineering，面向对象的软件工程）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要求的重要武器，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方面的局限，目前已很少使用。

1994 年 10 月，Grady Booch 和 Jim Rumbaugh 合作，他们首先将 Booch 93 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（Unitied Method）。1995 年秋，OOSE 的创始人 Jacobson 加盟到这一工作。经过 Booch，Rumbaugh 和 Jacobson 3 人的共同努力，于 1996 年 6 月和 10 月分别发布了两个新版本，并将 UM 重新命名为 UML（Unitied Method Language），即建模语言。它只提供了建模语言，没有包括建模方法。成为面向对象的软件工程工具。

面向对象方法的本质，是主张从客观世界固有的事物出发来构造系统，提倡用人类在现实生活中常用的思维方法来认识、理解和描述客观事物，强调最终建立的系统能够映射问题域，即系统中的对象以及对象之间的关系能够如实地反映问题域中固有事物及其关系。这恰恰是从分析和设计阶段入手才能根本解决的问题。

六、可视化开发

可视化开发是 20 世纪 90 年代软件界最大的两个热点之一。随着图形用户界面的兴起，它在软件系统中所占的比例也越来越大，有的甚至高达 60% ~ 70%。一时间图形用户界面的开发成了软件开发中的一个瓶颈，产生这个问题的原因是图形界面元素的生成很不方便，为此微软公司的 Windows 操作系统提供了应用程序编程接口（Application Programming Interface，API），它包含了 600 多函数，调用这些函数能够进行图形用户界面的开发。但是这些函数中包含了大量的参数，使得基于 Windows API 的开发变得相当困难。为此 Borland C++ 推出了 Object Windows 编程，它将 API 的各部件用对象类进行封装，提供了大量预定义的类，并为这些类定义了许多成员函数。开发人员利用子类对父类的继承性，以及实例对类方法的引用，可以省去或减少对这些类和成员函数的定义。但是这些预定义的类数量很多，要掌握它们，仍然是一个负担，因此人们利用 Windows API 和 Borland C++ 的 Object Windows 生成了一批可视化开发工具。

可视化开发方法就是在可视化开发工具提供的图形界面上，直接使用界面元素，诸如菜单、按钮、对话框、编辑框、单选框、复选框、列表框和滚动条等。开发人员引用这些元素，并且在这些元素的某些事件上添加需要处理的代码，即可自动生成应用软件。这类应用软件的工作方式是事件驱动，对每一事件，由系统产生相应的消息，并传递给相应的响应函数去处理。

实际上，可视化开发目前还不能单独作为一种开发方法，确切地说，它是一种辅助工具。目前大多数开发人员主要在编程环节上使用可视化工具，随着系统分析和系统设计的可视化工具的逐渐推出，人们会真正享受到可视化开发所带来的软件开发新理念。