第 1 章　概 论

1.1　软件工程简述

1.1.1　软件工程发展史

软件工程是随着计算机系统的发展而逐步形成的计算机科学领域中的一门新兴学科。软件工程的发展可分为 4 个时期。

1.1.1.1　20 世纪 40 年代中期到 60 年代中期

这个时期计算机硬件从电子管电子计算机发展到晶体管电子计算机，价格昂贵，运算速度低，存储量小。软件通常是规模较小的程序，软件的设计开发者和使用者往往是同一个人。软件设计通常只注意如何节省存储单元、提高运算速度，除了程序清单之外，没有其他任何文档资料。

1.1.1.2　20 世纪 60 年代中期到 70 年代中期

这个时期计算机硬件发展到集成电路计算机，运算速度和内存容量都相应提高了。出现了“软件作坊”,许多用户不再自己开发软件,而是去“软件作坊”购买软件。随着计算机应用的日益普及，软件需求量急剧增长。用户的需要和使用环境发生变化时，软件可修改性又很差，往往需要重新编制程序，其研制时间很长，不能及时满足用户要求，质量得不到保证。所谓“软件危机”由此开始。

如 IBM 公司的 OS/360 系统和美国空军后勤系统，在开发过程中都花费了几千人年的工程量，最后都以失败告终。其中 OS/360 系统由 4000 个模块组成,共约 100 万条指令，花费了 5000 人年的工程量，经费达数千万美元，结果却失败了。

1968 年北大西洋公约组织(NATO）的计算机科学家在联邦德国召开国际会议，正式提出了“软件工程”(Software Engineering）的术语。从此一门新兴的工程学科诞生了。当时“软件工程”还处于学术研究阶段，但已对软件开发产生了巨大影响。著名的例子：1971 年 IBM 公司运用软件工程技术成功地开发了“纽约时报情报库系统”和“空间实验室飞行模拟系统”，而且软件生产率比以前提高了一倍。

1.1.1.3　20 世纪 70 年代中期到 80 年代

这个时期硬件发展到大规模集成电路计算机，计算机硬件的功能和质量都不断提高。计算机应用不断地扩大，软件开发生产率提高的速度远远跟不上计算机应用迅速普及深入的趋势，软件产品供不应求，软件危机日益严重，为了维护软件还要耗费大量的成本。当时美国的统计表明，对计算机软件的投资占计算机软件、硬件总投资的 70%,到 1985 年软件成本大约

占总成本的 90%。为了对付不断增长的“软件危机”，软件工程学把软件作为一种产品批量生产。软件工程运用工程学的基本原理和方法来组织和管理软件生产，以保证软件产品的质量和提高软件生产率。

1.1.1.4 20 世纪 80 年代以后

计算机系统发展的第四代不再是单台的计算机和计算机系统，而是计算机软件和硬件的综合效果。由复杂操作系统控制的强大桌面机、广域网和局域网，与先进的应用软件相互配合，计算机体系结构从集中的主机环境转变为分布式的客户机／服务器环境。为此，软件开发的第四代技术可以举例如下：面向对象技术已在许多领域迅速取代了传统的软件开发方法；专家系统和人工智能软件从实验室进入了实际应用，解决了大量的实际问题；人工神经网络软件展示了模式识别与拟人信息处理的前景；并行计算、网络计算机、虚拟现实技术、多媒体技术和现代通信技术使人们可以采用和原来完全不同的方法进行工作。

1.1.2 软件危机

软件危机是指在计算机软件开发和维护时所遇到的一系列问题。软件危机主要包含下面两方面的问题：一是如何开发软件以满足对软件日益增长的需求；二是如何维护数量不断增长的已有软件。

1.1.2.1 软件危机主要表现形式

1. 软件开发成本高，研制进度不能预先估计，用户不满意

由于软件应用范畴越来越广泛，很多软件的应用领域往往是软件开发者不熟悉的.加之开发人员与用户之间信息交流不够，导致软件产品问题太多，研制的进度一再拖延，不能如期完成任务。因而，软件开发成本和进度都与原先的估计相差太大，引起用户不满。

2. 软件产品的质量差，可靠性得不到保证

软件质量无确切的评价标准，软件质量保证技术还没有应用到软件开发的全过程，导致软件产品质量问题频频发生。

3. 软件产品难以维护

早期的程序不注意可读性，不强调可维护性，程序中存在的错误很难改正。用户的需求往往会不断变化，为适应新的要求而维护软件相当困难。当时软件开发不注意保留文档资料，也造成开发和维护的很大困难。

4. 软件发展跟不上硬件的发展和用户的要求

硬件成本逐年下降,软件应用日趋广泛,软件产品‘供不应求”,与硬件成本相比,软件成本越来越昂贵。

1.1.2.2 产生软件危机的原因

产生软件危机的原因与软件本身的特点有关，也与软件人员开发时存在的问题有关。

软件是计算机系统中的逻辑部件.软件产品往往规模庞大，给软件的开发和维护带来客观的困难。软件一般要使用 8～10 年，在这漫长的时间里，很可能出现开发时没有考虑周全的问

题，使运行出现问题，需要及时维护。

软件人员忽视软件需求分析的重要性，轻视软件维护，也是造成软件危机的原因。

1.1.2.3 解决软件危机的途径

目前，计算机的应用日益广泛，世界上发达国家的许多企业将全部投资的 10% 以上用于计算机,而其中 70% 以上用于管理方面。但到目前为止，计算机的体系结构在硬件上仍然是冯·诺依曼计算机。硬件的基本功能只能做简单的运算与逻辑判断，主要还是适用于数值计算。对于非数值计算问题，是用编制程序来解决的，因而使软件复杂、庞大，只能由专门的人员来编制软件。假设计算机能实现智能化，计算机能自动进行推理和运算，正确解决用户所提出的问题，那么软件危机就会有根本性的缓解。新一代计算机体系结构的研制可能还需要一段时间。

在目前的计算机硬件条件下，我们要解决以下问题：

(1) 要使用好的软件开发技术和方法；

(2) 要有良好的组织、严密的管理，各类人员要相互配合共同完成任务；

(3) 使用好的软件开发工具，提高软件生产率。

就像机械工具可提高人类的工作能力一样，软件工具可使软件开发工作做得既快又好。如果把各个软件生产阶段使用的工具集合成一个整体，支持软件开发全过程，就构成软件工程支撑环境。软件生产需要开发和使用好的软件支撑环境，为了解决软件危机，需要有技术措施（好的方法和工具），还要有组织管理措施。软件工程正是从技术和管理这两方面来研究如何更好地开发和维护计算机软件的。

1.1.3 软件、软件工程

软件是指计算机程序及其有关的数据和文档，也包括固化了的程序。

软件的发展大体经历了程序、软件、软件产品等 3 个阶段。早期的程序规模小，随着系统程序的增加，人们把程序区分为系统程序和应用程序，并把它们称为软件，在开发过程中很少考虑到它们的维护问题。当软件需求量大大增加后，人们把软件视为产品，强调软件的“可维护性”，确定了软件开发的各个阶段必需完成的各种规格书、说明书、用户手册等（称为“文档”）。B. Boehm 指出：“软件是程序以及开发、使用和维护所需要的所有文档(document)。”特别当软件成为商品时，文档是必不可少的。没有文档，仅有程序是不能称为软件产品的。

软件工程是软件开发、运行、维护和引退的系统方法。这是国标 GB/T 11457－1995 软件工程术语中定义的。

软件工程是指导计算机软件开发和维护的工程学科。软件工程采用工程的概念、原理、技术和方法来开发与维护软件。软件工程的目标是实现软件的优质高产。

1.1.4 软件工程的基本原理

著名软件工程专家 Boehm 综合有关专家和学者的意见并总结了 TRW 公司多年来开发软件的经验,于 1983 年在一篇论文中提出了软件工程的 7 条基本原理。

(1) 用分阶段的生命周期计划严格管理；

(2) 坚持进行阶段评审；
(3) 实行严格的产品控制；
(4) 采用现代程序设计技术；
(5) 结果应能清楚地审查；
(6) 开发小组的人员应该少而精；
(7) 承认不断改进软件工程实践的必要性。

遵循前 6 条基本原理，能够实现软件的工程化生产；按照第 7 条原理，不仅要积极主动地采纳新的软件技术，而且要注意不断总结经验。在本课程的学习中，读者将体会到这 7 条基本原理的含义和作用。

1.1.5 软件工程学

软件工程学的主要内容是软件开发技术和软件工程管理。其中，软件开发技术包含了软件开发方法学、软件工具和软件工程环境；软件工程管理学包含了软件工程经济学和软件管理学。

1.1.5.1 软件开发方法（Software Development Methods）

早期的程序设计属于个人活动性质，程序员无统一的方法可循，到了 20 世纪 60 年代后期，兴起了结构程序设计，人们采用结构化的方法来编写程序。

经典的结构程序设计只允许使用顺序结构、条件分支结构和循环结构这 3 种基本结构。这样不仅可改善程序的清晰度，而且能提高软件的可靠性和生产率。随后，人们认识到编写程序仅是软件开发过程中的一个环节。典型的软件开发工作中编写程序所需要的工程量只占软件开发全部工作量的 10%～20%。有效的开发应包括需求分析、软件设计、编写程序等几个阶段，于是形成了“结构化分析”、“结构化设计”、Jackson 方法、Warnier 方法等软件开发方法。到 20 世纪 80 年代又广泛应用了面向对象设计方法。各种软件开发方法的适用范围不尽相同，本书介绍一些比较成熟的目前广泛使用的软件开发方法。

1.1.5.2 软件工具(Software tools)

为了提高软件设计的质量和生产效率，已发展了许多“帮助开发和维护软件的软件，人们称之为软件工具（Software tools），也称软件自动工具(Automated tools)。

例如，我们要在微机上用某种高级语言开发一个应用软件，往往首先要用编辑程序把源程序输入计算机，然后用编译程序进行编译，如果发现错误，就要重新用编辑程序对源程序进行修改。编译通过后，用连接程序把所有的目标程序，同有关的库程序连接起来，构成一个可执行软件。这里，编辑程序、编译程序、连接程序及支持它们的计算机操作系统都属于软件工具。另外，有测试阶段的测试数据产生器、排错程序、跟踪程序、静态分析工具和覆盖监视工具等，设计阶段和分析阶段也有一些工具。众多的软件工具组成了“工具箱（Tool box)”或“集成工具（Integrated tool)”，供软件开发人员在软件生存期的各个阶段根据不同的需要选择使用合适的工具。目前，软件工具发展迅速，许多用于软件分析和设计的工具正在建立，其目标是实现软件生存期各个环节的自动化。

1.1.5.3 软件工程环境(Software Engineering Environment ,简称 SEE)

软件开发方法和工具是软件开发的两大支柱，它们之间密切相关。软件开发方法提出了明确的工作步骤和标准的文档格式，这是设计软件工具的基础，而软件工具的实现又将促进软件开发方法的推广和发展。

软件工程环境正是方法和工具的结合。在 1985 年第八届国际软件工程会议上,关于“软件开发环境”的定义是“软件开发环境是相关的一组软件工具集合，它支持一定的软件开发方法或按照一定的软件开发模型组织而成”。

软件开发环境的设计目标是提高软件生产率和改善软件质量。本书将在以后章节中介绍一些常用的软件开发方法、软件工具及软件工程环境。

1.1.5.4 软件工程管理

一个企业如果只有先进的设备和技术，没有完善的管理，是不可能获得应有的经济效益的。软件生产也一样，如果管理不善，是不可能高质量、按时完成任务的。软件工程管理就是对软件工程生存期内的各阶段的活动进行管理。软件工程管理的目的是为了能按预定的时间和费用，成功地完成软件的开发和维护任务，圆满地完成预定的软件开发项目。

软件工程管理包括软件费用管理、人员组织、工程计划管理、软件配置管理等各方面内容。

1. 费用管理

一般来讲，开发一个软件是一种投资，人们总是期望将来获得较大的经济效益。要从经济角度分析，开发一个系统是否划算，从而让使用部门负责人正确地做出是否开发这项系统的决定。我们从软件开发成本、运行费用、经济效益等方面来估算整个系统的投资和回收的数量。

软件开发成本主要表现为人力消耗及相应的开发人员的工资报酬，软件运行费用取决于系统的操作费用和维护费用。其中操作费用包括操作人员的人数、工作时间、消耗的各类物资等项开支，系统的经济效益是指因使用系统而可以节省的费用和增加的收入。

由于运行费用和经济效益两者在软件的整个生存周期内都存在，总的效益和软件生存周期的长度有关，所以，应合理地估算软件的寿命。一般在进行成本／效益分析时一律假设生存周期为 5 年。

2. 人员组织

软件开发不是个体劳动，需要各类人员协同配合，共同完成工程任务，因而应该有良好的组织、周密的管理。

3. 工程计划管理

软件计划是在软件生存周期的早期确定的。在计划实施过程中，需要时，对工程进度应做适当的调整。在软件开发结束后应写出软件开发总结，以便在今后的软件开发工程中能作出更切实际的计划。

4. 软件配置管理

软件工程各阶段所产生的全部文档和软件本身构成软件配置。每当完成一个软件工程步骤，就涉及到软件工程配置，必须使软件配置始终保持其精确性。软件配置管理就是在系统整个生存周期内控制配置的状态和变动，验证配置项的完整性和正确性。

1.2 软 件 过 程

软件过程是为了获得高质量软件所需要完成的一系列任务的框架，它规定了完成各项任务的工作步骤。

ISO 9000 把软件过程定义为：“把输入转化为输出的一组彼此相关的资源和活动”.

过程定义了运用方法的顺序，应该交付的文档，开发软件的管理措施，各阶段任务完成的标志。软件过程必须科学、合理，才能获得高质量的软件产品。

本节介绍软件开发、维护全过程应完成的基本任务。

1.2.1 软件生存周期

软件产品从定义开始，经过开发、使用和维护，直到最后被淘汰的整个过程称为软件生存周期。

这如同一个人从出生开始，经过儿童、青年、中年、老年到死亡。在人的一生中，国家和社会对人的负担主要在儿童、青少年时期的培养及老年丧失劳力后的供养。而人从参加工作后就对国家与社会做贡献。贡献越大，人的价值也就越大。同样，软件生存周期中软件的开发要进行投资，消耗价值，当软件交付使用后开始产生价值，软件维护又要消耗价值。软件生存周期中，消耗价值越少，即软件开发与维护所花的费用越低，软件的使用寿命越长，产生的价值就越大，这就是掌握软件工程学的目的。

生存周期是软件工程的一个重要概念。一个软件产品的生存周期可划分为若干个互相区别而又有联系的阶段。把整个生存周期划分为若干个阶段，是实现软件生产工程化的重要步骤。赋予每个阶段相对独立的任务，逐步完成每个阶段的任务，能够简化每个阶段的工作，容易确立系统开发计划，还可明确系统各类开发人员的分工与职责范围，以便分工协作，保证质量。

每一阶段中的工作均以前一阶段的结果为依据，并作为下一阶段的前提。每个阶段都要有技术审查和管理复审，从技术和管理两方面对这个阶段的开发成果进行检查，及时决定系统是继续进行，还是停工或返工。应防止到开发结束才发现先行工作存在的问题，造成不可挽回的损失和失败的浪费现象。每个阶段都进行的复审，主要检查是否有高质量的文档资料，前一个阶段结束了，后一个阶段才能开始。开发单位的技术人员可根据所开发软件的性质、用途及规模等因素决定在软件生存周期中增加或减少相应的阶段。

一个软件产品的生存周期可划分为若干个阶段，这是实现软件生产工程化的重要步骤。

划分软件生存周期的方法有多种，可按软件规模、种类、开发方式、开发环境等来划分生存周期。不管用哪种方法划分，划分的原则是相同的。

软件生存周期划分的原则：

(1) 各阶段工作任务彼此间尽可能相对独立，便于逐步完成每个阶段的任务，能够简化每个阶段的工作，容易确立系统开发计划；

(2) 同一阶段的工作任务性质尽可能相同，这样，有利于软件工程的开发和组织管理，明确系统各类开发人员的分工与职责范围，以便协同工作，保证质量。

软件生存周期一般由软件计划、软件开发和软件运行维护 3 个时期组成。软件计划时期

分为问题定义、可行性研究两个阶段。软件开发时期可分为需求分析、软件设计、测试等阶段。软件交付使用后在运行过程中需要不断地维护，使软件能持久地满足用户的需要。

下面简单介绍软件生存周期各阶段的主要任务。

1. 问题定义

该阶段是软件生存期中最短的阶段。这个阶段要确定系统的目标、规模和基本任务，要有书面报告。这就需要对系统用户和使用单位的负责人进行调查，问题定义报告要征得用户的同意。

2. 可行性研究

该阶段对问题定义阶段确定的系统目标进行全面的分析，探索问题是否有可能解决，具体地确定工程的规模、目标，并估计系统的成本和效益，得出系统是否需要开发的结论，及时中止不值得投资的项目，避免出现不必要的浪费。

可行性研究的任务是对今后的行动提出建议，其目的是确定问题是否值得解决，而不是立即去解决问题。要达到这个目的必须进行客观分析，而且应在尽量短的时间内确定问题是否可行。因而可行性研究是对系统进行简化了的系统分析和设计。首先需进一步分析问题定义，如果问题定义阶段确定的目标、规模正确，应进一步导出系统逻辑模型，经分析后提出几种可能的解法，对每种解法仔细研究其可行性。如果问题定义阶段确定的规模过大，目标难于达到或开发成本过大，收益甚微不值得投资，就应及时建议停止项目开发，避免人力、物力、时间的浪费。

一般来说，每种解决方法的可行性可从以下两方面进行研究。

(1) 经济可行性　根据系统规模目标，确定系统的硬件资源和软件资源的需求规格，估算软件开发成本，分析系统的经济效益是否能超过它的总成本，从经济角度分析系统是否值得投资。

(2) 技术可行性　分析系统是否有某种约束条件，如果有，要全部列出来。根据现有的技术，分析是否能实现系统目标，研究预定的操作方式用户是否能接收。

例如，某市招干考试成绩管理系统，考生分 3 个专业，不同专业考试科目不同。法律专业考政治、语文、法律 行政专业考政治、语文、行政学 财经专业考政治、语文、财经学。每位考生在报名时登记姓名、地址、年龄、报考专业。考生报名后，招干办公室(简称招干办)根据考生报考的专业及考生地址在市区或郊区编排准考证号码和安排考场。考生参加考试后，输入每个考生的各门课的成绩，并统计出每个考生 3 门课考试成绩的总分。按准考证号的顺序打印出考生考试成绩单，并分发给考生。各专业分别将考生按成绩总分从高到低的次序排序，供用人单位决定录取名单。

对本题进行可行性分析：

(1) 技术可行性　有几千名考生报名参加招干考试，若手工计算每位考生 3 门课成绩总分、填写考生成绩单需一式几份（一份给考生，另一份招干办公室留存）；再将考生按成绩总分排序后供录用单位参考。这些工作都是很繁重的，而且手工进行抄写成绩时，要抄写好几份，很容易出错。若将数据输入计算机，虽然需要花费些时间，但根据这些数据由计算机计算总分和按总分排序的速度很快，数据一次输入后可多次使用（输出考生成绩单给考生，考生成绩表供招干办留存；成绩排序后供录用参考；录用后输出录用通知书给考生，录用名单给招干办公室及录用单位）。可见，用计算机建立数据库、开发数据库管理应用系统进行招干考试成绩管理，在技术上是完全可行的。只要系统界面设计合理，系统用户的操作就会易学易用。

(2) 经济可行性　开发招干考试成绩管理系统时，如果能完全理解用户需求、设计开发出便于应用的软件，以后每年都可以使用该软件，解脱烦琐的成绩统计工作，省时、省力，从经济方面考虑也是完全可行的。

可行性研究阶段结束时应提供的文档有：问题可行性论证报告及项目开发计划任务书或问题应中止开发的论证报告。

软件问题定义、可行性研究评审通过后，软件项目才算真正立项，才能进入软件开发阶段。

3. 需求分析

主要确定软件系统应具备的具体功能。通常用数据流图、数据字典和简明算法描述表示系统的逻辑模型。用户熟悉系统的全部功能和完成任务的全过程，但常常不能完整地、准确地提出任务要求，而软件人员若不做深入的调查，就会对用户的具体要求了解不全面，理解不透彻，这样，设计出来的目标系统就不符合用户的需求。这个阶段应当准确、完整地描述用户的要求，因而必须经用户确认才能进入下一阶段。这样可防止系统的设计与用户的实际需求不相符的后果。

4. 软件设计

通常软件设计的第一步是进行概要设计，先考虑几种可能的设计方案，分析各种方案的成本／效益，与用户共同确定系统所采纳的方案，再进行系统结构设计，确定软件的模块结构。软件设计的第二步是详细设计，描述如何具体地实现系统。此后才是软件设计的第三步，即程序设计（也称编码）阶段。软件设计的第四步是测试阶段，先测试软件的每个模块，再将模块装配在一起进行测试（集成测试），最后在用户的参与下进行验收测试，用户验收后软件才可交付使用。

5. 软件维护

软件运行期间，通过各种必要的维护使系统适应环境变化，延长使用寿命和提高软件的效益。软件运行期间会由于潜在的问题而发生错误；用户在使用后会提出一些改进或扩充软件的要求；软件运行的硬件、软件环境有时也会发生变化等。这些情况使软件需要不断地进行维护才能继续使用而不致被废弃。每次维护的要求、方案、计划及如何修改程序、重新测试、验收等一系列步骤都应详细准确地记录下来，作为文档加以保存。

1.2.2　软件开发模型

根据软件生产工程化的需要，生存周期的划分也有所不同，从而形成了不同的软件生存周期模型（SW life cycle model），或称软件开发模型。

软件开发模型总体来说有传统的瀑布模型和后来兴起的快速原型模型。具体可分为瀑布模型、快速原型、喷泉模型、软件重用开发模型和螺旋模型，以下分别加以介绍。

1.2.2.1　瀑布模型(Waterfall Model)

瀑布模型遵循软件生存期的划分，明确规定每个阶段的任务，各个阶段的工作顺序展开恰如奔流不息拾级而下的瀑布。

瀑布模型把软件生存周期分为计划、开发、运行 3 个时期。这 3 个时期又可细分为若干个阶段：计划时期可分为问题定义、可行性研究两个阶段，开发时期分为需求分析、概要设计、详细设计、程序设计、软件测试等阶段，运行时期则边运行边维护。

瀑布模型如图 1-1 所示。

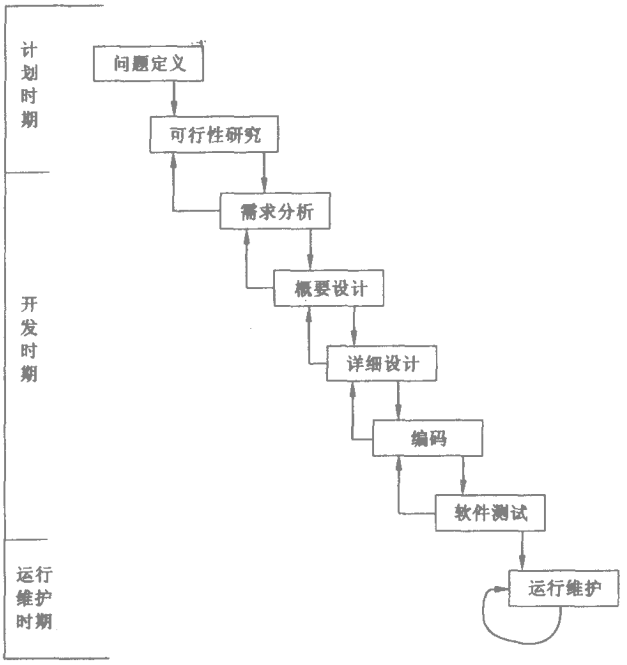


图 1-1 瀑布模型

瀑布模型软件开发有以下几个特点。

1. 软件生存周期的顺序性

顺序性是指：只有前一阶段工作完成以后，后一阶段的工作才能开始；前一阶段的输出文档，就是后一阶段的输入文档。只有前一阶段有正确的输出，后一阶段才可能有正确的结果。如果在生存周期的某一阶段出现了错误，往往要追溯到在它之前的一些阶段。

瀑布模型开发适合于在软件需求比较明确，开发技术比较成熟，工程管理比较严格的场合下使用。

2. 尽可能推迟软件的编码

程序设计也称为编码。实践表明，大、中型软件编码开始得越早，完成所需的时间反而越长。瀑布模型在编码之前安排了需求分析、概要设计、详细设计等阶段，从而把逻辑设计和编码清楚地划分开来，尽可能推迟程序编码阶段。

3. 保证质量

为了保证质量，瀑布模型软件开发在每个阶段都要完成规定的文档，每个阶段都要对已完成的文档进行复审，以便及早发现隐患，排除故障。

本书以瀑布模型为典型开发模型，介绍各阶段工作的具体方法、步骤、工具，对其他模型可以参照执行。

1.2.2.2 快速原型(Rapid Prototype Model)

正确的需求定义是系统成功的关键。但是许多用户在开始时往往不能准确地叙述他们的需要，软件开发人员需要反复多次地和用户交流信息，才能全面、准确地了解用户的要求。当用户实际使用了目标系统以后，也常常会改变原来的某些想法，对系统提出新的需求，以便使

系统更加符合他们的需要。

理想的做法是先根据需求分析的结果开发一个原型系统，请用户试用一段时间，以便能准确地认识到他们的实际需要是什么，这相当于工程上先制作“样品”试用后，做适当改进，然后再批量生产一样，这就是快速原型法。虽然此法要额外花费一些成本，但是可以尽早获得更正确完整的需求，可以减少测试和调试的工作量，提高软件质量。因此快速原型法使用得当，能减少软件的总成本，缩短开发周期，是目前比较流行的实用开发模式。

根据建立原型的目的不同，实现原型的途径也有所不同，通常有 3 种类型。

1. 渐增型

先选择一个或几个关键功能，建立一个不完全的系统，此时只包含目标系统的一部分功能或对目标系统的功能从某些方面做简化，通过运行这个系统取得经验加深对软件需求的理解，逐步使系统扩充和完善。如此反复进行，直到软件人员和用户对所设计的软件系统满意为止。

渐增型开发的软件系统是逐渐增长和完善的，所以从整体结构上不如瀑布型方法开发的软件那样清晰。但是，由于渐增型开发过程自始至终都有用户参与，因而可以及时发现问题加以修改，可以更好地满足用户需求。

2. 用于验证软件需求的原型

系统分析员在确定软件需求之后，从中选出某些应验证的功能，用适当的工具快速构造出可运行的原型系统，由用户试用和评价。这类原型往往用后就丢弃，因此构造它们的生产环境不必与目标系统的生产环境一致，通常使用简洁而易于修改的超高级语言对原型进行编码。

3. 用于验证设计方案的原型

为了保证软件产品的质量，在概要设计和详细设计过程中，用原型来验证总体结构或某些关键算法。如果设计方案验证完成后就将原型丢弃，则构造原型的工具不必与目标系统的生产环境一致。如果想把原型作为最终产品的一部分，原型和目标系统可使用同样的程序设计语言。

软件快速原型开发方法的开发过程如图 1-2 所示。开发人员听取用户意见，进行需求分析，尽快构造出原型，原型的用途是获得用户的真正需求。原型由用户运行评估测试，根据用户的意见修改原型，再次请用户试用，逐步使其满足用户的需求。产品一旦交付给用户使用，维护便开始。根据需要，维护工作可能返回到需求分析、设计或编码等不同的阶段。

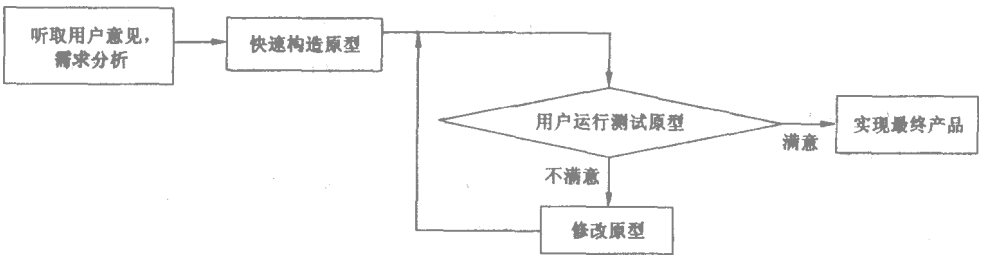


图 1-2　快速原型开发过程

1.2.2.3　喷泉模型 (Fountain Model)

按传统的瀑布模型开发、管理软件需要有两个前提：

(1) 用户能清楚地提供系统的需求；

(2) 开发者能完整地理解这些需求，软件生存周期各阶段能明确地划分，每个阶段结束时

要复审，复审通过以后下一阶段才能开始。

然而，在实际开发软件时，往往用户事先难以说清系统需求，开发者也由于主客观的原因，缺乏与用户交流的机会，其结果使系统开发完成后，修改、维护的开销及难度过大。

在面向对象软件开发的喷泉模型中，着重强调不同阶段之间的重叠，认为面向对象软件开发过程，不需要或不应该严格区分不同的开发阶段，如图 1-3 所示。

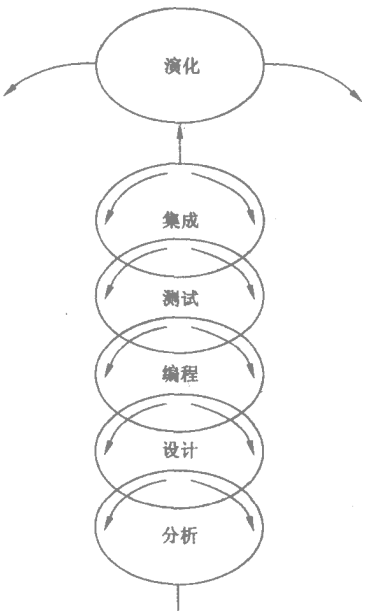


图 1-3 喷泉模型

基于喷泉模型，Hodge 等人提出将软件开发过程划分为概念模型分析、系统设计、对象设计与实现、测试和系统组装集成等 5 个阶段，体现出分析和设计之间的重叠。

1. 概念模型分析

这个阶段主要目标是建立系统模型。系统模型中的对象是现实世界中客观对象的抽象，结构清晰、易于理解、易于描述规范。在分析阶段面向问题域建立对象模型和过程模型。

2. 系统设计

给出模型对象和过程的规范描述。

3. 对象设计和对象实现（编程）

面向对象设计方法强调软件模块的再用和软件合成，因而在对象设计和实现时，并不要求所有的对象都从头开始设计，而是充分利用以前设计工作的成果。在软件开发时检索对象库，若是对象库中已有的软件模块可用，则可再用；否则，重新定义新的对象，进行设计和实现。

4. 测试

测试所有的对象及对象相互之间的关系是否符合要求。

5. 系统组装集成

面向对象软件特点之一是软件重用和组装技术。对象是数据和操作的封装载体，组装在一起才构成完整的系统。软件是对象模块的复合，而软件设计是将对象模块经过进程控制而构造生成所需的系统。

采用进程控制语言 PCL 可以选择和操作对象模块，并将它们组装成软件系统过程。PCL 的控制结构包括顺序、选取和重复 3 种基本形式，还允许基本活动的并发执行。因而软件系统的构造组装过程可以用 PCL 语言表示为一系列可以顺序或并发执行的活动。PCL 语言用程序的形式控制软件的构造组装过程，尽可能支持对象模块的再用，大大提高编程效率。

1.2.2.4 软件重用开发模型（Software Reuse Model）

这种开发模型旨在开发具有各种一般性功能的软件模块，将它们组成软件重用库，这些模块设计时考虑其适应各种界面的接口规格，可供软件开发时利用。软件重用的主要优点是减少软件生产中的重复开发，避免软件开发人员的大量重复劳动，提高开发效率，缩短开发周期，降低开发成本。软件重用库的模块不仅要便于选择使用，而且还应具有允许扩充、积累其成分的性能。

通常软件重用分两种：

(1) 重用程序以各种源程序形式存库；

(2) 重用程序是经过编译的目标程序。

软件重用模式开发过程如下。

1. 设计重用库模块

重用库中的模块要经历模块定义、功能规格描述、模块设计、编码、模块功能测试、模块登记、模块目录编制，此后可放入重用库中。

2. 软件系统设计

在重用库建立后，软件系统的设计步骤为需求分析，功能定义、设计，在重用库中选择模块,编码,测试,验收,运行维护。

在选择重用库中模块时，有时会不十分满意或找不到可重用的模块，这就需要修改原有模块或建立新的模块，以扩充可重用库。

1.2.2.5 螺旋模型(Spiral Model,简称 SM)

瀑布模型要求在软件开发的初期就完全确定软件的需求，这在很多情况下往往是做不到的。螺旋模型试图克服瀑布模型的这一不足。

螺旋模型是在 1986 年由 B. W. Boehm 提出的。SM 把软件开发过程安排为逐步细化的螺旋周期序列，每经历一个周期，系统就细化和完善一次。SM 每一螺旋周期由下列 6 个步骤组成,如图 1-4 所示。

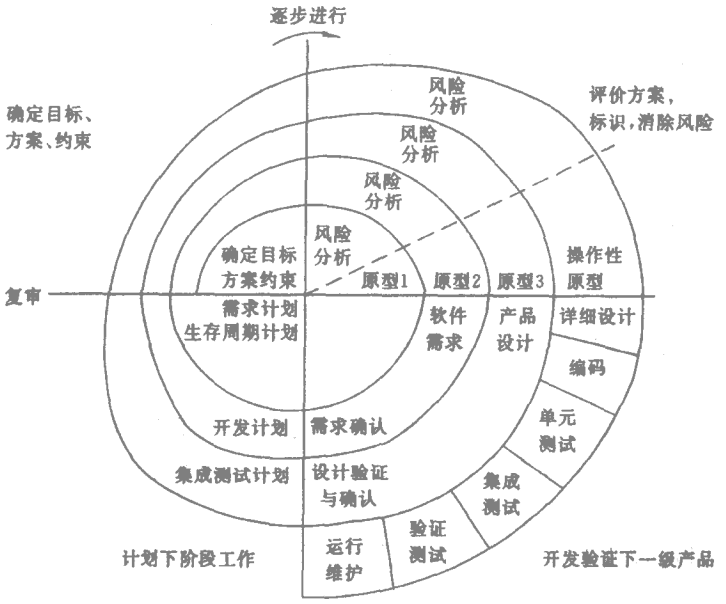


图 1-4 螺旋模型开发过程

1. 确定任务目标

根据初始需求分析项目计划，确定任务目标、可选方案和约束。

2. 选择对象

对各种软硬件设备、开发方法、技术、开发工具、人员、开发管理等对象进行选择,并决定进行研制、购买还是利用现有的软件。

3. 分析约束条件

软件开发的时间、经费等限制条件。

4. 风险分析

评估目标、对象、约束条件三者之间的联系，列出可能出现的问题及问题的严重程度等，把最重要的问题作为尚未解决有风险的关键问题。

5. 制定消除风险的方法

应有详尽的说明和周密的计划，并估计可能产生的后果，依此来开发软件，为制定下一周期的计划打下基础。

6. 制定下一周期的工作计划

在第一个螺旋周期，确定目标、选择对象、分析约束，通过风险分析制定消除风险的方法，初步开发原型 1，制定系统生存期计划。

在第二个螺旋周期，进一步明确系统的目标、开发方案及约束条件，通过风险分析制定消除风险的方法,在原型 1 的基础上开发原型 2。进一步明确软件需求，进行需求确认，修改开发计划。

在第三个螺旋周期，再进一步确认系统目标、开发方案及约束条件，进行风险分析，制定进一步消除风险的方法，在原型 2 的基础上开发原型 3。此时可进行产品设计，再对设计进行验证和确认，制定集成测试计划。

在第四个螺旋周期，软件开发方案、系统目标和约束条件得到确定，在风险分析的基础上，开发具有实用价值的可操作性原型；此时可对产品进行详细设计，进入编码、单元测试、集成测试阶段；最后进入验收测试，验收合格后交付用户使用，进入运行、维护阶段。

1.3 软件开发方法

软件开发的目标是在规定的投资和时间限制内，开发出符合用户需求的高质量软件。研究软件开发方法的目的是使开发过程规范化，使开发有计划、按步骤地进行。软件开发方法的基本内容是：把解决的问题划分成若干个工作步骤；有具体的文档格式，即把每个工作步骤都记录下来，保证人员之间的相互交流；要确定出软件评价标准。已经推出的软件开发方法和技术有很多种，要根据软件的实际情况选择合适的方法。本书主要介绍以下 3 类软件开发方法——面向数据流设计方法、面向数据结构设计方法和面向对象设计方法。

1.3.1 面向数据流设计方法

1.3.1.1 建立系统逻辑模型

数据流是软件开发人员分析问题的出发点和基础。数据流从系统的输入端进入系统后要经过一系列的变换或处理，最后由输出端流出。为了描述这样的过程，可以用数据流图这一有力的图形工具。这实际上就是建立软件系统的逻辑模型，面向数据流设计就是在需求分析的基础上把数据流图转换为对软件结构的描述。在结构化设计方法中软件结构用结构图来描述，后续章节将一一介绍。

通常所说的结构化设计即 SD(Structured Design)是属于面向数据流的设计方法，是由

E. Yourdon 和 L. L. Constantine 等人于 1974 年提出的；是在模块化、自顶向下逐步求精、结构化程序设计等技术的基础上发展起来的。它与结构化分析（SA)一起,构成一个完整的结构化分析和结构化设计技术，又称 Yourdon 方法。这是目前使用最广泛的方法之一，也是本书要着重介绍的方法。

面向数据流的设计方法把数据流映射成软件结构，根据数据流的类型不同,映射的方法也不同。数据流可分为变换型和事务型两种。使用面向数据流设计方法时，首先要对数据流进行分析，判断数据流属于变换型数据流还是属于事务型数据流；然后对两种数据流进行不同的处 理 。

1. 变换型数据流

数据沿输入通路进入系统，同时由外部形式变换为内部形式，通过变换中心，经加工处理后再沿输出通路变换为外部形式离开系统，这种数据流称为变换型数据流。

2. 事务型数据流

数据沿输入通路到达某一个处理，该处理根据输入数据的类型在若干个处理序列中选择某一个来执行，这类数据流称为事务型数据流，而这样的处理称为事务中心，如图 1-5 所示。

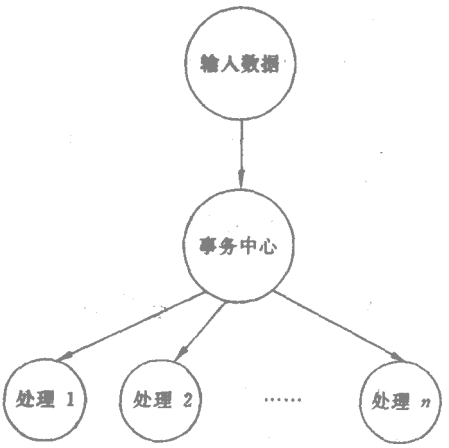


图 1-5　事务型数据流示意图

使用面向数据流方法进行设计时，首先判断数据流的类型。若是事务型数据流，则分析其事务中心和数据接收通路，再映射成事务处理结构，分析每个事务以确定它的类型，选取一条活动通路；若是变换型数据流，则应区分输入和输出分支，映射成变换结构，经加工处理变换为输出数据离开系统。

例 1-1　招干考试成绩统计系统。

该系统的主要工作过程是：

先输入每个考生的基本情况(姓名、地址、年龄、报考专业),再输入考生的各门课的成绩,并统计出每个考生 3 门课考试成绩的总分。打印出考生考试成绩单，并分发给考生。各专业分别将考生依据成绩总分从高到低的次序排序，以便决定录取名单。考试后要进行考试成绩统计。

数据进入系统后,进行处理(计算总分、排序、录用),并得到结果,因而属于变换型数据流。

例 1-2　工资管理系统。

职工工资数据库里存放了所有职工的工资数据：职工号、姓名、性别、职务、职称、基本工资、工龄工资、岗位津贴、车贴、伙食补贴、房贴、病事假扣款、房租、水电费等。该系统含有职工

的调入、调出、增加工资、发工资时打印工资单等功能。

分析：该系统含多项功能，每次选择一种功能进行处理，因而属于事务型数据流。

在一个大型系统的数据流图中，我们会发现变换型和事务型两类结构往往同时存在。有时系统的总体结构具有多种事务处理能力，但在它们的某（几）条通路上，可能出现变换型结构。有时系统的整体结构为变换型，其中某些部分又可能具有事务处理的特点。对于两类结构同时存在的系统，我们可以将系统分级处理。系统的高层数据流图属于哪一类就按这一类进行处理，在某一部分内部属于另一类结构的数据流时再换一种办法处理，总之要按实际情况灵活处理，把两种类型的分析应用到同一系统的不同部分。

例 1-3　某校医疗费管理系统。

要求数据库中存放每个职工的职工号、姓名、所属部门。职工报销时填写所属部门、职工号、姓名、日期。医疗费分校内门诊费、校外门诊费、住院费、子女医疗费 4 种。该校规定，每年每个职工的医疗费有一个限额（如 380 元），限额在年初时确定，每个职工一年内报销的医疗费不超过限额时可全部报销；超过限额时，超出部分只可报销 90%，其余 10% 由职工个人负担。职工子女的医疗费也有限额（如 240 元）。医疗费管理系统每天记录当天报销的若干职工或职工子女的医疗费的类别、金额，在当天下班前让系统自动结账，统计当天报销的医疗费总额，供出纳员核对。每笔账要保存备查，每天所报销的费用要和各个职工已报销的金额累计起来，以便检查哪些职工已超额，系统要配有适当的查询功能。年终结算后，下一年度开始时要对数据库文件进行初始化。职工调离本单位、职工调入本单位或在本单位内部部门间调动，数据库文件要及时修改。

该系统的主要功能是 数据输入、查询、统计及系统维护(初始化、职工调动、改医疗费限额等），因而总体上看属于事务型数据流。但其中数据输入子功能先输入当日数据，然后结算当日累计数据供出纳员核对；每笔账要存入明细账中去，再和每个职工已报销的数据累加。在这个子功能中，数据进行多次处理，因而属于变换型数据流。在这个系统中，事务型、变换型两种特性同时存在。

1.3.1.2　完成软件结构设计

对于变换型和事务型数据流，结构化设计方法的任务是把数据流图表示的逻辑模型转换为对软件结构的描述。面向数据流设计方法的过程是，首先分析数据流类型，划分流程段，确定变换中心或事务中心；然后对两种类型分别进一步分析，转换为软件结构中的模块，对模块进行划分或合并，完成软件结构设计(见第 3 章）。下面分别介绍对两种类型数据流进行分析设计的步骤。

1. 变换型分析

变换型分析是经以下几个步骤把具有变换特点的数据流图映射成软件结构的。

(1) 复查系统逻辑模型，确保系统的输人数据和输出数据符合实际情况。例如，例 1-1 中输入数据是考生基本情况(姓名、地址、报考专业)和考生成绩，其中包括政治、语文、专业课(行政、法律或财经）。输出数据是考生成绩单和录用通知书。

(2) 复查、细化数据流图。确保系统逻辑模型的正确性，把处理细化为相对独立的子功能。

如在例 1-1 中“输入考生基本情况”，可以分为给数据库中的记录赋予准考证号和其他数据(姓名、地址、专业)两个步骤：

(a) 同一专业同一地区的考生准考证号码是连续的，我们可按数学公式让计算机自动给各个考生记录的“准考证号”赋值，免去了人工逐个输入准考证号的单调、繁重的手工操作。

(b) 考生的姓名、地址、专业必须逐个输入。

计算考生成绩总分，可让计算机自动进行，是一个单独的处理过程，在计算出考生成绩总分后才可得到输出数据“考生成绩单”。按总分排序则是另一个单独的处理过程，有了前一个过程的结果才能进入后一过程。按考生成绩总分排序后才可进入下一步“录用”处理，进而可得到需要输出的数据“录用通知书”。该系统的输出结果不止一个。

(3) 确定数据流具有变换特性还是事务特性。例如，例 1-1 某市招干考试成绩统计系统具有变换特性。

(4) 把系统划分为输入数据、输出数据及数据变换 3 大部分，再把数据变换划分为几个具体步骤。

通过以上分析，系统的处理过程就很清楚了，软件结构也可确定了。

2. 事务型分析

任何系统的数据进入系统后都要进行加工处理，最后得到输出结果，因而都可使用变换型分析法来设计软件结构。但是在数据流具有明显的事务型特点时，应采用事务型分析法为宜。事务型数据流，通常有一个输入数据项，它的不同处理结果会导致系统在下一步进入多个不同的处理分支中的某一个分支。这个数据项称为事务项。事务型数据流映射成软件结构时，数据流的事务中心对应软件结构的控制中心。把数据流图中事务中心分出的各个处理通路映射成控制中心下属的各个子模块。

例如,例 1-2 工资管理系统具有事务特性，我们可建立一个主菜单让用户选择执行菜单中列出的几个功能：职工调动、工资变动、打印工资单等。程序中的菜单变量的值就是控制中心，取不同的值就使程序调用不同的子模块，执行不同的子程序。

1.3.2 面向数据结构设计方法

面向数据结构的设计方法是按输入、输出以及内部存储信息的数据结构进行设计的，把对数据结构的描述变换为对软件结构的描述。

在许多应用领域中，信息的结构层次清楚，输入数据、输出数据以及内部存储的信息有一定的结构关系。数据结构不仅影响软件的结构设计，还影响软件的处理过程。如重复出现的数据通常由循环结构来控制；一个数据结构具有选择特性，既可能出现也可能不出现，就采用条件选择程序来控制；如果一个数据结构为分层次的，软件结构也必然为分层次的。所以，数据结构充分地揭示了软件结构。使用面向数据结构的设计方法，首先需要分析确定数据结构，并用适当的工具清晰地描述数据结构，最终得出对程序处理过程的描述。

下面分别介绍两种面向数据结构的设计方法： Jackson 方法和 Warnier 方法。

1.3.2.1 Jackson 方法

Jackson 方法由英国的 M. Jackson 提出，在欧洲较为流行。它特别适合于设计企事业管理类的数据处理系统。

Jackson 设计方法分以下 4 个步骤：

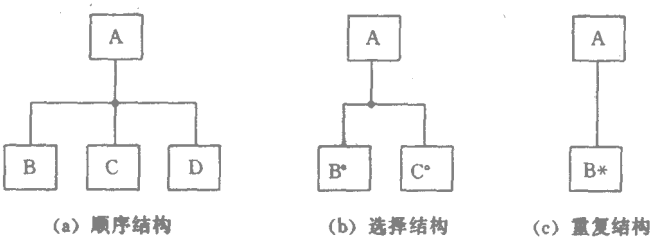
(1) 分析并确定输入数据和输出数据的逻辑结构；

(2) 找出输入数据结构和输出数据结构中有对应关系的数据单元；

(3) 从描述数据结构的 Jackson 图导出描述程序结构的 Jackson 图；

(4) 列出所有的操作和条件，并把它们分配到程序结构图中去。

Jackson 把数据结构分为以下 3 种基本类型（见图 1-6 ）。



(a) 顺序结构　(b) 选择结构　(c) 重复结构

图 1-6　数据结构的 3种基本类型

(1) 顺序结构。

顺序结构的数据由一个或多个元素组成，每个元素依次出现一次。图 1-5 (a)中数据 A 由 B,C,D 3 个元素顺序组成。

(2) 选择结构。

选择结构的数据包含两个或多个元素，每次使用该数据时按一定的条件从这些元素中选择一个。图 1-5 (b) 中根据条件在 B 或 C 中选择一个。B 和 C 的右上方加符号‘ ° ’表示从中选择一个。

(3) 重复结构。

重复结构的数据，根据条件由数据元素出现零次或多次组成。图 1-5 (c)中数据 A 由 B 出现零次或多次组成。数据 B 后加符号‘ * ’表示重复。

Jackson 图可以清晰地表示数据的层次结构，形象直观易读。既可表示数据结构也可表示程序结构。

下面结合具体例子进一步说明 Jackson 结构程序设计方法。

在例 1-1 某市招干考试成绩统计系统中，不同专业考试课程不同：法律专业考政治、语文、法律；行政专业考政治、语文、行政学；财经专业考政治、语文、财经学。每个考生有准考证号、姓名。考生成绩统计系统按准考证号的顺序输入每个考生的准考证号、姓名、专业和各门课程的成绩，并统计出每个考生 3 门课程考试成绩的总分。按准考证号的顺序打印出考生考试成绩单。各专业分别将考生按成绩总分从高到低的次序排序，以便决定录取名单。

成绩单格式如表 1-1、表 1-2、表 1-3 所示。

表 1-1　招干考试成绩单

准考证号	姓　名	专　业	政　治	语　文	法　律	总　分
110001	王　勇	法　律	80	78	90	248

表 1-2　招干考试成绩单

准考证号	姓　名	专　业	政　治	语　文	行政学	总　分
210002	陈　民	行政学	60	68	70	198