国家级精品课程系列教材

软件开发技术基础

主编 顾 刚

西安电子科技大学出版社
http://www.xduph.com

国家级精品课程系列教材

软件开发技术基础

主编　顾　刚

西安电子科技大学出版社

内 容 简 介

本书旨在介绍计算机软件技术领域中最基本、最实用的原理和方法。本书从当前高等院校计算机教育的实际出发，充分结合计算机技术本身的发展状况，在内容取舍、篇章结构、叙述方式、实用性编程案例等方面都进行了精心的设计和组织。

本书共 9 章，内容分为：软件工程、线性表、堆栈与队列、树和图、查找和排序、操作系统及 Windows 程序设计、数据库及应用程序开发、网络软件开发技术、多媒体编程技术。网络、多媒体和数据库这三个方面的编程方法在本书中有较详细的介绍。

本书着眼于提高学生对软件本质的理解和软件设计的能力。本书可作为高等院校非计算机专业的本科生、研究生学习计算机软件技术课程的教材，也可作为广大从事计算机软件开发人员学习计算机技术的参考书。

图书在版编目(CIP)数据

软件开发技术基础/顾刚主编. —西安：西安电子科技大学出版社，2010.12

国家级精品课程系列教材

ISBN 978-7-5606-2479-2

Ⅰ. ① 软… Ⅱ. ① 顾… Ⅲ. ① 软件开发—高等学校—教材 Ⅳ. ① TP311.52

中国版本图书馆 CIP 数据核字(2010)第 189663 号

策　　划　曹　昳

责任编辑　李文娟　曹　昳

出版发行　西安电子科技大学出版社(西安市太白南路 2 号)

电　　话　(029)88242885　88201467　　邮　　编　710071

网　　址　www.xduph.com　　电子邮箱　xdupfxb001@163.com

经　　销　新华书店

印刷单位　陕西华沐印刷科技有限责任公司

版　　次　2010 年 12 月第 1 版　2010 年 12 月第 1 次印刷

开　　本　787 毫米×1092 毫米　1/16　印 张 19.75

字　　数　468 千字

印　　数　1～3000 册

定　　价　28.00 元

ISBN 978 - 7 - 5606 - 2479 - 2/TP • 1234

XDUP 2771001-1

前　言

计算机已成为人类社会生活不可或缺的工具。大多数人使用计算机都是以获取信息为主要目的，通过互联网、物联网来解决各自的生活、工作、学习等方方面面的问题。然而，对计算机更广泛、更深层次的应用则是通过设计编制各式各样的软件来解决人们各自的问题。计算机在这些方面的应用前景更广阔、更美妙，仍有待于各行各业的人员掌握和提高软件开发技术去进一步开拓。计算机软件可以模拟当今世界任何自然现象和人工现象，这已是不争的事实，从这个意义上讲，软件开发能力已成为现代人应具有的基本素质之一。

何为软件开发技术？这个问题上“仁者见仁，智者见智”，目前没有统一的定义。本书中列出了六大部分：软件工程、数据结构、操作系统及其编程技术、数据库编程技术、网络编程技术和多媒体编程技术。编者认为这些是软件开发人员所必备的最基本、最实用的技术。只有掌握了这些技术和方法，才能增强软件开发人员的“后劲”和对软件的“悟性”，从而切实提高软件开发能力。本书具有如下特点：

(1) 基础性和原理性并重。着重介绍计算机软件开发技术领域中最基本的原理，增强学生对软件本质的理解和软件开发环境的适应能力。

(2) 实用性和先进性并存。侧重于网络、数据库、多媒体三方面软件开发技术的阐述，软件开发环境选取流行的工具和平台，并阐述了当今先进的技术方法。

本书共分为 9 章，第 1 章为软件工程概述；第 2 章至第 5 章主要涉及数据结构方面的理论知识，其中在 Hash 查找方法上给出了所有实用、有效的算法程序；第 6 章介绍了操作系统方面的程序设计；第 7 章介绍了数据库软件开发方法；第 8 章介绍了网络软件开发方法；第 9 章介绍了多媒体软件开发方法。在具体教学安排上，各校可以根据教学学时、学生的程度等具体情况，选取教学内容，教学顺序可以不按本书的章节次序灵活安排。

本书由顾刚教授主编，顾刚教授编写了第 2 章和第 5 章，薛涛副教授编写了第 1 章、第 6 章和第 8 章，贾应智副教授编写了第 7 章和第 9 章，最后由顾刚教授统稿。

本书主编是国家级精品课程“软件开发技术基础”第一负责人，在精品课程网站 http://202.117.35.252 中展示了本课程大量的教学资源。

由于认识水平的局限，关于软件开发技术教学方面存在的一些问题，有待进一步探索和深层次的总结，欢迎读者批评指正。

编　者

2010 年 7 月

目　录

第 1 章　软件工程概述

计算机系统经历了四个不同的发展阶段，对软件产品的数量、种类、功能、性能的需求在不断攀升。但是由于“软件危机”的困扰，软件已经成为限制计算机系统发展的瓶颈。

为了更有效地开发与维护软件，从 20 世纪 60 年代末开始认真研究解决软件危机的方法和途径，从而逐渐形成了一门新兴的工程学科——计算机软件工程学。

1.1　软件和软件危机

著名计算机科学家 Roger S. Pressman 在谈到软件时曾说过：“计算机软件已经成为一种驱动力。它是进行商业决策的引擎，是现代科学研究和工程问题寻求解答的基础，也是鉴别现代产品和服务的关键因素。它被嵌入在各类系统中：交通、医疗、电信、军事、工业生产过程、娱乐、办公、……难以穷举。软件在现代社会中确实是必不可少的，而进入 21 世纪，软件将成为从基础教育到基因工程的所有领域新进展的驱动器。”

1.1.1　软件的概念

如今无论是在各行各业还是在人们的日常生活中，计算机几乎无处不在、无所不能，小到个人电子文档排版、下棋娱乐，大到指挥控制“勇气”号火星探测器登陆火星。为什么计算机能发挥如此神奇的作用呢？是计算机系统中的“软件”在发挥着作用。

在计算机系统中，软件是一个逻辑部件，软件所具有的特征包括：

1．软件产品与硬件产品的表现形式不同

软件和硬件是截然不同的两种产品。硬件是看得见、摸得着的物理部件或设备；而软件是以程序和文档的形式存在，它通过在计算机上运行来体现其作用。

在研制硬件产品时，人的创造性活动表现在把原材料转变成有形的物理产品。例如研制出一种新型的计算机主板、CPU 芯片、可重写的激光磁盘、高速路由器等。

在研制软件产品的过程中，人们的生产活动表现在：要创造性地抽象出问题的求解模型，然后根据求解模型写出程序，最后经过调试、运行程序得到求解问题的结果。整个生产、开发过程是在无形化方式下完成的，其能见度极差，这给软件开发、生产过程的管理带来极大的困难。

2．软件产品质量的体现方式与硬件不同

硬件产品在整个生产过程中，从原材料的选购、设计图纸的绘制，到每一个加工、生产的工艺、工序过程都有严格的质量控制和检验标准，不合格的零部件及产品通常在还没有流入到下一道工序前就被检测出来，从而可以有效地保证产品的质量。

软件产品只有在实际问题求解过程中被证实是可行的且被用户接受，才能成为产品，也才有存在的价值。软件产品一旦定型，其生产过程很简单，只是复制、拷贝而已。但是软件在设计、编程和实现过程中的各个阶段其质量难以保证和检验。

3．软件产品的成本构成与硬件不同

硬件产品的成本构成中有形的物质占了相当大的比重，例如，工厂、矿山、设备、运输机械、原材料等，人力资源占的比例相对较小。就硬件产品生存周期而言，成本构成中设计、生产环节占绝大部分，而售后服务只占少部分。

软件生产主要靠脑力劳动。软件产品的成本构成中人力资源占了相当大的比重。软件产品的生产主要是研制，生产成本主要是开发和研制的费用。研制成功后，产品生产就简单了，通过复制就能批量生产。

随着计算机应用领域的不断拓宽，对软件的需求越来越多，软件的生产费用也在不断增加，导致生产成本的不断增加。例如，1979 年美国国防预算为 1258 亿美元。其中 9%用于计算机领域，约 113 亿美元。在这 113 亿美元经费中，91 亿美元用于软件投资(约占 80%)，仅有不到 23 亿美元用于购买硬件设备。图 1-1 显示了在计算机系统中软件和硬件费用所占比例的变化趋势。

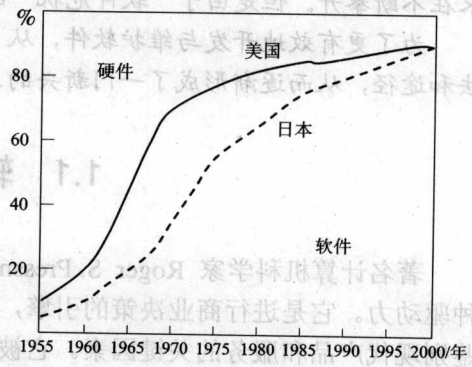

图 1-1　软件与硬件费用之比示意图

4．软件产品的生命周期和失效曲线与硬件不同

硬件产品存在老化和折旧问题。当一个硬件部件磨损时，可以用一个新部件去替换它。硬件会因为主要部件的磨损而最终被淘汰。

对于软件而言，不存在折旧和磨损问题，如果需要的话可以永远使用下去。但是软件故障的排除要比硬件故障的排除复杂得多。软件故障主要是因为软件设计或编码存在错误所致，必须重新设计和编码(设计、测试和调试)才能解决问题。

软件在其开发初始阶段存在很高的失效率，这是由于需求分析不切实际，或设计错误等引起的。当开发过程中的错误被纠正后，其失效率便下降到一定水平并保持相对稳定，直到该软件被废弃不用。在软件进行大的改动时，也会导致失效率急剧上升。

图 1-2 是软件和硬件的失效率曲线示意图。

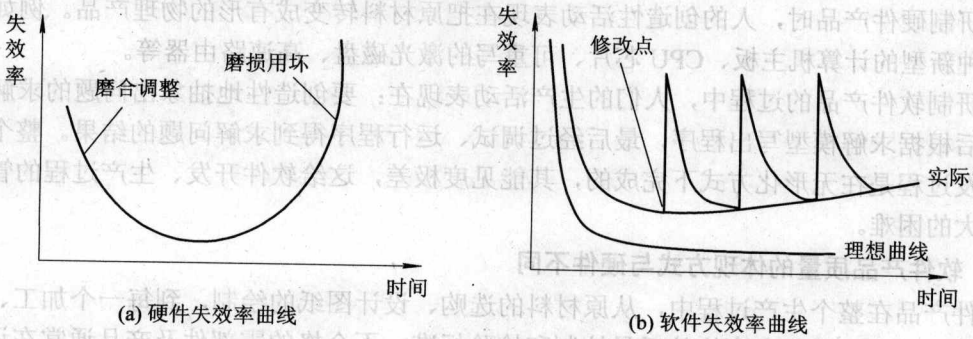

图 1-2　软件与硬件的失效率曲线

5. 大多数软件仍然是定制生产的

硬件产品的生产技术和工艺已经成熟，可以做到标准化、系列化成批生产。由于硬件产品具有标准的框架和接口，不论哪个厂家的产品，用户买来都可以集成、组装和替换使用。

尽管软件产品复用是软件界孜孜不倦追求的目标，在某些局部范围内几家领军软件企业和组织也建立了一些软件组件复用的技术标准，例如，OMG 的 CORBA、Microsoft 的 COM、SUN 的 J2EE，但是，目前还做不到大范围使用软件替代品，大多数软件仍然是为特定任务或用户定制的。

6. 软件产品的不同属性

为便于软件产品的研制、生产、维护和使用，软件还必须具有可维护性、独立性、效率性和可用性四个属性。

1.1.2　软件危机

从 20 世纪 60 年代开始，软件界在感受计算机应用造福人类的成功喜悦的同时，更经常遭受软件危机的袭扰。

以 IBM 公司的 OS/360 操作系统为例。它共有 4000 多个模块、100 万行指令，共投入 5000 人/年，耗资 5 亿美元，但在交付使用的系统中仍能找出 2000 个以上的错误。这个项目的负责人 F.D.Brooks 事后总结了他在组织开发过程中的沉痛教训时说："……正像一只逃亡的野兽落到泥潭中做垂死的挣扎，越是挣扎，陷得越深，最后无法逃脱灭顶的灾难。……程序设计工作正像这样一个泥潭，……一批批程序员被迫在泥潭中拼命挣扎，……谁也没有料到问题竟会陷入这样的困境……"。

在客观上，软件不同于硬件。软件开发实质上是逻辑思维的过程，在写出程序并拿到计算机上运行之前，软件开发的进展情况难以掌握，质量也难以评价，因此管理软件开发过程十分困难。同时，软件规模和复杂度呈指数状增长。成百上千人共同开发一个大型系统时，大量的通信、后勤工作成为问题。这常常是造成软件开发失败多，费用高的重要原因。人们面临的不光是技术问题，更重要的是管理问题。管理不善必然导致失败。

经研究发现，研制软件系统需要投入大量的人力、物力和资金，但是系统的质量却无法保证。开发软件所需的高成本与软件产品的低质量之间存在尖锐的矛盾，致使软件开发陷入不可自拔的恶性循环之中。这种现象被称为"软件危机"(Software Crisis)。

1. 软件危机的表现

软件危机主要体现在如下几个方面：

(1) 软件开发进度难以预测。拖延工期几个月甚至几年的现象并不罕见。据一项研究统计结果表明：只有 15%的项目是按计划进度完成的。

(2) 软件开发成本难以控制，投资一再追加，令人难以置信。据同样的研究统计结果表明：仅有 10%的项目是按费用计划完成的。

(3) 用户对软件产品的功能要求难以满足。

(4) 软件产品的质量无法保证，系统中的错误难以消除。软件是逻辑产品，质量问题很难以统一的标准度量，因而造成质量控制困难。据统计数据表明，在美国，软件开发项目

的开发时间平均超出计划时间的 50%。软件项目越大，情况就越坏。所有大型系统中，大约有 3/4 的系统有运行问题，要么不像预料的那样起作用，要么就根本不能使用。

(5) 软件产品难以维护。软件产品本质上是开发人员的逻辑思维活动的代码化描述，他人难以理解和替代。开发过程中因规范、标准、编程风格、文档资料、检测手段等很难统一，导致开发出的软件系统可维护性差。

(6) 软件通常缺少文档资料。软件的文档是开发组织和用户之间权利和义务的合同书，是系统总体设计者向开发人员下达的任务书，是系统维护人员的技术指导手册，是用户的操作说明书。缺乏必要的文档或者文档不合格，将给软件开发、维护带来严重的后果。

(7) 软件开发生产率的提高速度难以满足社会需求的增长。软件产品“供不应求”的现象是不能充分利用现代计算机硬件提供的巨大潜力的重要原因。

2. 产生软件危机的原因

产生软件危机的因素很多，概括起来有两个方面的主要因素：软件的特殊性和软件开发技术及管理的复杂性。前者是软件产品的内在因素，是客观存在的，要认识其规律，研究解决；而后者是外部因素，可以在工程实践中不断总结经验，摸索新途径，推出新技术和新方法。

1) 软件特点的因素

软件是一种逻辑产品，具有无形性。其实质是在软件运行以后，动态变化地处理过程。而软件在开发过程中，能见度差，这给开发过程的管理带来极大的困难，进度难以控制，开发质量难以评价和保证。例如，美国第四代宇宙飞船的软件包含 4000 万行目标代码。假如一个人一年可以开发出一万行的程序，该软件是否集中 4000 人的力量一年就可以完成呢？绝对做不到！因为代码长度增加了 4000 倍，复杂度的增加远远超过 4000 倍。

2) 软件开发技术、管理的因素

软件本身的特点决定了软件开发技术和管理的复杂性。但是导致产生软件危机的主要原因之一是相当多的软件开发和管理人员对所从事的工作缺乏正确的认识，在软件开发和管理过程中有意或无意地采用了错误的方法和技术所造成的。主要表现为：

- 开发人员和用户之间的矛盾。在软件开发初期，由于各种原因用户往往不能准确地提出需求描述。这就形成了开发人员和用户之间的屏障，双方缺乏共同语言和相互了解，因此无法紧密配合。
- 软件产品从定义、开发、使用和维护，直到最终被废弃，要经历一个漫长的时期，称之为软件生存周期。在生存周期的各个阶段有其具体的任务，为完成各个阶段的任务，又有各自不同的技术方法和操作步骤。只有科学地按生存周期各个阶段的任务、技术方法和操作步骤去实施，才能保证软件产品的质量。而急于求成，不按科学方法实施，不愿采用新的开发技术和开发工具，则势必造成“事倍功半”，甚至断送一个软件产品的“生命”。
- 重编程，轻需求分析；重开发，轻维护；重程序，轻文档。这样做的后果就是在软件系统中“埋藏”了许多故障隐患，直接危害着系统的可靠性和稳定性。经验证明，修改一个软件错误的代价与修改滞后时间呈正比关系。
- 忽视工程管理，是软件工程项目失败的主要原因之一。软件工程项目不同于一般工程，软件开发管理的对象是逻辑部件(求解模型、流程、算法、数据结构等)，这就给工程项目管理带来极大的难度。

● 开发工具落后导致软件开发效率低下。

3. 导致软件开发项目失败的原因

Standish Group 在 1995 年作了大量的调查研究后得出软件项目失败的原因，并按其重要程度进行了排序，如表 1-1 所示。

表 1-1 软件项目失败的主要原因

Factors(因素)	所占百分比
1. Incomplete Requirements(不完整的需求)	13.1%
2. Lack of User Involvement(缺乏用户参与)	12.4%
3. Lack of Resources(缺乏资源)	10.6%
4. Unrealistic Expectations(不实际的期望)	9.9%
5. Lack of Executive Support(缺乏执行的支持)	9.3%
6. Changing Requirements & Specifications(需求和规格的变化)	8.7%
7. Lack of Planning(缺乏计划)	8.1%
8. Do not Need It Any Longer(不再需要)	7.5%
9. Lack of IT Management(缺乏 IT 管理)	6.2%
10. Technology Illiteracy(技术落后/技术盲区)	4.3%
11. Others(其他)	9.9%

从表的 1、2、4、6、8 项可以看出，用户需求不完整、不清晰、不稳定是项目失败的主要原因，应引起足够的重视。软件开发方法的研究，应针对项目失败的原因系统地提出解决办法。

4. 解决软件危机的途径

如何解决软件危机，如何提高软件的生产效率和软件产品的可维护性，一直是困扰软件界的难题。人们经过长期的研究和探索，开始找出解决软件危机的途径。

解决软件危机要从组织管理措施和技术方法两个方面综合考虑，才能从根本上解决问题，两者缺一不可。

硬件生产和软件生产的效率之所以有如此巨大的差别，除了这两类产品的特征因素外，主要原因之一是组织管理方式。硬件生产早已采用了现代化工程管理方式组织生产，对生产环节各要素实现统一管理、优化调度、最佳组合；充分发挥有限资源的最大潜能；产、供、销总体平衡；机械化、自动化的生产线取代了人的体力操作，严密的质量检测仪器取代了人的脑力劳动等。而软件生产还是个体劳动方式，自产自销，手工劳动，不成规模，生产效率低下，质量检测还是凭个人经验。

软件开发不应是某种个体劳动的神秘技巧，而应该是一种组织良好、管理严密、各类人员协同配合、共同完成的工程项目。软件生产也必须采用现代化、社会化的组织管理方式，必须充分吸取和借鉴人类长期以来从事各种工程项目所积累的行之有效的原理、概念、技术和方法，特别要吸取几十年来人类从事计算机硬、软件研究和开发的经验教训。

软件开发本身是高度智力密集型劳动，使用先进、得心应手的工具可以“放大”人的智能和体能。在软件开发过程中，要坚持不断引入新技术、新开发工具，将软件工程方法

学与自动化软件工具相结合，使得开发与维护过程中繁杂重复性的手工劳动能够智能化、自动化，从而提高软件的开发效率和开发质量。

总之，为了解决软件危机，既要有技术措施(方法和工具)，又要有严谨的组织管理措施。软件工程正是从管理和技术两方面，研究如何更好地开发和维护软件的一门新兴学科。

1.2　软件工程的基本概念

由于软件危机，使人们认识到软件开发有其独特的内在规律，遵循这个规律，才能掌握软件生产的自由。什么是软件生产的规律呢？从1968年北大西洋公约组织在联邦德国召开的一次专门研讨解决软件危机的国际会议上，正式提出了“软件工程”的概念。为研究解决软件开发过程中的开发技术、思想、方法、工具等问题，软件工程逐步发展成为一门独立的学科——软件工程学。

1.2.1　软件工程的定义

什么是“软件工程”？

著名软件工程专家B. W. Boehm给软件工程下的定义是：运用现代科学技术知识来设计并构造计算机程序，以及为开发、运行和维护这些程序所必需的相关文件资料。

IEEE给出的定义为，将系统化的、规范的、可度量的方法应用于软件的开发、运行和维护过程。即将工程化方法应用于软件开发与维护的过程中。

教科书中的定义为，运用系统的、规范的和可定量的方法来开发、运行和维护软件。

从这些不同的定义和描述中不难看出，软件工程具有极其丰富的内涵，上述定义都无法表达它涉及的全部实际内容。

软件工程是一门交叉学科，涉及到计算机科学、管理科学、工程学和数学。软件工程的理论、方法、技术都是建立在计算机科学的基础上，它是用管理学的原理、方法进行软件生产管理；用工程学的观点进行费用估算、制定进度和实施方案；用数学方法建立软件可靠性模型以及分析各种算法。

软件工程具有方法、工具和过程三个要素。软件工程方法为软件开发提供了“如何做某项工作”的技术指南。软件工程工具为软件工程方法提供了自动的或半自动的软件支撑环境。软件工程的过程是将软件工程的方法和工具综合起来以达到合理、高效、高质量地进行计算机软件开发的活动步骤。

1.2.2　软件工程的原则

自1968年提出“软件工程”的概念以来，专家学者们根据自己从事软件工程的实践经验和体会，提出了许多软件工程的原则。1983年B. W. Boehm在他发表的一篇论文中提出了软件工程的七条基本原则。这些基本原则可以说是对众说纷纭的软件工程原则的归纳和总结，适用于大型软件工程项目。这七条基本原则简述如下。

1．严格管理分阶段的项目计划

据统计数据表明，在失败的软件项目中有一半左右是由于计划不周或虽然有计划但执行不严造成的。因此，要根据工程项目各个阶段不同性质任务的需要，制定详细、可行的项目实施计划，并严格按计划进行管理，才能保证工程项目顺利地进行。

在项目实施过程中，往往会发生因客户或上级人为因素干扰而背离项目预定计划的情况。这里要强调指出的是：项目计划可以变更，但必须是有科学根据的、有组织的、有计划的变更，决不允许主观随意地修改计划。项目组各级人员只有严格按计划各尽其职的义务，没有随意修改计划的权利。

Boehm 认为，在整个项目实施过程中，应该制定并严格执行六类计划，它们包括项目概要计划、里程碑计划、项目控制计划、产品控制计划、验证计划和运行维护计划。

2．坚持进行阶段评审

根据 Boehm 等人的统计，软件的大部分错误是在编码之前造成的，设计错误占 63%，编码错误只占 37%。而且，由于软件特点的因素，错误发现和改正得越晚，所付出的代价越大。因此，在各个阶段都要进行严格的评审，以便尽早发现并改正错误。

3．实行严格的产品控制

在软件开发过程中不应随意修改需求(修改需求往往要付出很高的代价)，但在客观条件及外部环境变化后，相应地修改需求是必要的。在改变需求时，为了保持软件配置的一致性，必须实行严格的产品控制。这里主要是指各个阶段产生的文档与软件系统的一致性。所有修改必须严格按规程进行评审，获准后才能实施。任何修改的内容必须记录入档备案。

4．采用现代化程序设计技术

如何提高软件产品的生产效率，一直是软件界人士追求的目标。采用先进的程序设计技术不仅可以提高软件的开发效率，还可以提高软件维护的效率。结构化程序设计方法曾在 20 世纪 60、70 年代创造过辉煌的业绩；90 年代成熟起来的面向对象 OO 程序设计方法已成为程序设计人员追求的“宠儿”，并且由此引发了一场程序设计方法的革命。

5．结果应能清楚地审查

软件产品是逻辑产品，很难进行管理和质量监控。为了提高软件开发过程的可见性以及开发质量的可测性，应该根据软件开发项目的总目标及完成期限，规定开发组织的责任和产品标准，从而使得开发过程中应得到的结果能够清楚地审查。

使软件开发过程和开发质量有形化的重要途径是文档。详细地规定工程各阶段的文档格式和内容，并通过文档实现对人员组织、项目进度、质量控制、系统配置的管理，而文档的内容是可以清楚地审查的。

6．开发小组的人员应该少而精

开发小组人员的素质和数量是影响软件产品质量和开发效率的重要因素。实践证明：素质高的开发人员的开发效率高(可能高几倍甚至几十倍)，而错误明显较少。另一方面，开发小组人员越多，通信路径越长(人员交流、信息反馈周期长)，管理、协调的难度就越大。因此，开发小组成员少而精是软件工程的一条基本原理。

7．承认不断改进软件工程实践的必要性

一般而言，上述六条基本原则已经能够保证实现软件的工程化生产了。但 Boehm 认为，

应该把“承认不断改进软件工程实践的必要性”作为第七条基本原则。因为，社会的巨大需求促进了软件技术的飞速发展，不能因循守旧，要把积极主动地采用新的软件技术、不断总结经验、改进工作、提高生产效率作为软件工程实践中的主线，牢牢把握，不断创新进取。

1.2.3 软件生存周期

软件生存周期是指一个软件从提出开发要求直到该软件报废为止的整个过程，如图 1-3 所示。针对不同的开发模型、开发对象以及开发方法，软件生命周期可以有不同的划分。如果不考虑上述不同的因素以及应用领域、项目规模和复杂性，软件生存周期可以划分为软件定义、软件开发和软件维护三个时期，每个时期又进一步划分为若干个阶段。

软件定义时期的核心任务是“做什么”。即要确定软件开发工程的总目标；确定工程的可行性；提出实现工程目标应该采用的策略及系统应实现的功能；估计完成该项工程需要的资源和成本，并且制定工程进度表。这个时期的工作通常又称为系统分析，由系统分析员负责完成。软件定义时期通常可进一步划分为 3 个阶段：问题定义、可行性研究和需求分析。

软件开发时期的核心任务是“怎么做”，即要具体设计和实现在软件定义时期定义的软件系统。它又包括 4 个阶段：总体设计、详细设计、编码和测试。其中前两个阶段又称为系统设计，后两个阶段又称为系统实现。

维护时期的核心任务是“改变”，即要保证软件持久地满足用户各个方面改变的需要。维护时期不再进一步划分阶段，但是每一次维护活动本质上都是一次压缩和简化了的定义和开发过程。例如，如果提出的变更要求是更加新的功能，那就要涉及新一轮的软件开发过程：需求分析、系统设计、编码及测试。

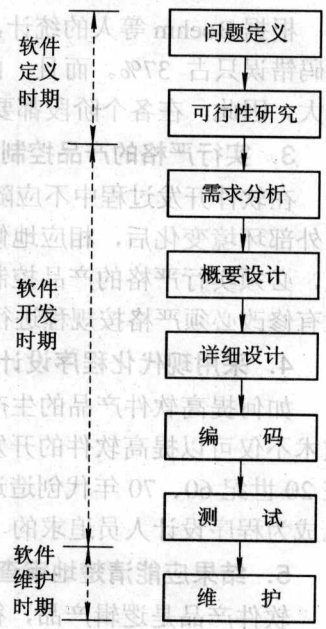

图 1-3　软件生命周期图

下面简要介绍软件生命周期每个阶段的基本任务。

1. 问题定义

问题定义阶段必须明确回答“要解决的问题是什么”。开发人员首先要进行有关工程项目的调查研究，收集、分析有关工程的信息资料。然后根据现有对情况的了解和用户的要求写出有关问题的性质、工程目标和工程规模的书面报告。对产生的报告经过必要讨论和修改，再经用户确认后形成正式的问题报告。

2. 可行性研究

可行性研究阶段要回答“用户提出的问题是否可解”的问题。这项工作实际上是一次简化和抽象的工程项目分析和设计过程，当然时间不能长，投入不能大，因此多是由有经验的系统分析员来完成本阶段的工作。该阶段工作的结果是产生出可行性报告。报告必须明确回答该工程项目可行否，必须提供可供决策的依据和论据。

3. 需求分析

在前两个阶段完成后，即可转入需求分析阶段。这个阶段的任务是准确地确定“为了解决提出的问题，系统必须做什么”，即要确定系统必须实现哪些功能。这个阶段工作往往非常困难。通常是用户不了解计算机能做什么，因此不能提出准确的问题描述，这就给了解需求的开发人员带来很大的困难。因此，如何准确了解用户的需求，既有技术问题、方法问题，也有工具问题。例如，可以借助图形、逻辑框图、文字等手段和工具与用户交流，引导用户准确地表达他们所要做的工作和所要解决的问题。

该阶段结束时要产生软件项目计划、需求分析说明书(系统规格说明书)等文档。对于小型软件来说，可以不进行可行性研究，相应文档也可省，但需求分析说明书是不可缺少的。

4. 总体设计

这个阶段要回答的问题是，“概括地说，应该怎样实现系统”。即要设计出系统的总体结构、系统全部的逻辑功能及总体的数据结构等。

首先，应该设计出实现目标系统的几种可能的方案。通常至少应该设计出高、中、低三种成本的方案来。开发人员应该用适当的表达工具描述每种方案，分析每种方案的优缺点，并在充分权衡各种方案的利弊的基础上，推荐一个最佳方案。此外，还应该制定出实现最佳方案的详细计划。如果客户接受所推荐的方案，则转入下一步工作——系统结构设计和系统功能模块分解。软件结构设计的一条基本原理是系统的模块化分解；即将一个大的、复杂的系统分解为小的、简单的若干个功能模块，这些模块按求解问题的逻辑关系组成合理的层次结构，得到的结果是由模块及模块连接关系组成的软件结构图。

5. 详细设计

总体设计阶段用比较抽象、概括的方式提出了解决问题的办法。详细设计阶段的任务就是把解法具体化，即要回答“应该怎样具体地实现这个系统”的问题。但是这个阶段实现的系统不是在计算机中可以实现的代码系统，而是供编程人员实现编码的程序的详细规格说明书——工程蓝图。

详细设计也称为模块设计，在这个阶段将详细地设计每个模块，确定实现模块功能所需要的算法和数据结构。这个阶段结束后，要交付概要设计说明书和系统设计说明书。

6. 系统编程

系统编程阶段的任务是根据设计说明书中每个模块的算法描述，用指定的程序设计语言编写出相应的程序，该阶段要交付的是源程序及其文档。

7. 系统测试

系统测试阶段的任务是尽可能地检查出程序中存在的错误，提高软件系统的可靠性，其目的是检验系统“做的怎样”。这阶段又可分为三个步骤：模块测试、组装测试和确认测试。模块测试是每个模块程序写出后，单独测试该模块是否能实现设计的功能要求。组装测试是以一组相关模块组合在一起成批方式进行，目的是测试模块之间的接口及参数传递是否正确。确认测试又称验收测试，是按系统规格说明书的规定，由用户对系统进行测试。测试以整体方式(整个系统的全部程序)进行，目的是测试整个系统的功能和性能是否能满足设计要求。前两种测试是开发组织内部进行的测试，而后一种测试是以用户为主进行的。该阶段结束后应交付测试报告，说明测试数据的选择、测试用例以及测试结果是否符合预

期结果。测试发现问题之后要经过调试，找出错误原因和位置，然后进行改正。必要时还可以通过现场测试或平行运行等方法对系统做进一步的测试检验。

8. 系统维护

系统维护的任务是改正软件系统在使用过程中发现的隐含错误，扩充在使用过程中用户提出的新的功能及性能要求，其目的是维持软件系统的“正常运行”。这阶段的文档是软件问题报告和软件修改报告，它记录发现软件错误的情况以及修改软件的过程。

1.3　软件工程过程模型

软件工程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，是对软件开发过程中各个阶段工作和活动准则的一种描述。

过程定义了运用方法的顺序、应该交付的文档资料、为保证软件质量和协调变化所需要采取的管理措施，以及标志软件开发各个阶段任务完成的里程碑。为获得高质量的软件产品，软件过程必须科学、有效。通常使用软件工程过程模型(也称为软件生存周期模型)简洁地描述软件过程，它规定了把生存周期划分成哪些阶段及各个阶段的执行顺序。

1.3.1　瀑布模型

1970 年 Winston Royce 提出了著名的“瀑布模型”，直到 20 世纪 80 年代早期，它一直是唯一被广泛采用的软件开发模型。瀑布模型将软件生存周期中各活动规定为依线性顺序连接的若干阶段，包括需求分析、系统设计、编码、测试和维护。

瀑布模型如图 1-4 所示，它将软件生命周期划分为制定计划、需求分析、软件设计、程序编写、软件测试和运行维护等六个基本活动，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。

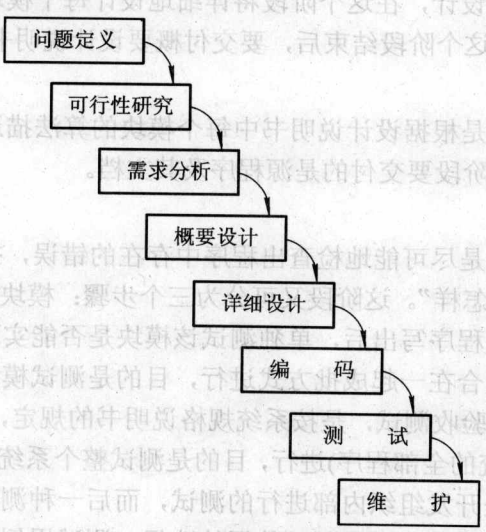

图 1-4　瀑布模型