普通高等教育“十一五”规划教材

软件工程与实践

张 凯 编著

中国电力出版社
www.infopower.com.cn

普通高等教育“十一五”规划教材

软件工程与实践

张 凯 编著

中国电力出版社
www.infopower.com.cn

内容简介

本书是一本比较好的软件工程设计与实践教材，作者一直在日本从事软件设计开发15年，根据软件工程的基本概念结合自己的开发经验和心得体会来编写的。本书主要介绍了软件与软件工程的基本概念，结构化设计开发方法，面向对象开发方法，制定计划与管理，从需求到设计，编程工程，从测试到运行，质量管理，设计文档，设计评审和软件开发新话题等内容。从理论到实践对软件工程作了透彻明了的解说。最后加上了常见到的设计模板，使大家可对软件整个开发流程有一个清楚的认识。

图书在版编目（CIP）数据

软件工程与实践/张凯编著. —北京：中国电力出版社，2007.6

ISBN 978-7-5083-5374-6

普通高等教育“十一五”规划教材

Ⅰ.软… Ⅱ.张… Ⅲ.软件工程-高等学校-教材 Ⅳ.TP311.5

中国版本图书馆CIP数据核字（2007）第046999号

丛 书 名：普通高等教育“十一五”规划教材

书　　名：软件工程与实践

出版发行：中国电力出版社

地　　址：北京市三里河路6号　　邮政编码：100044

电　　话：（010）68362602　　传　　真：（010）68316497，88383619

本书如有印装质量问题，我社负责退换

服务电话：（010）88515918（总机）　　传　　真：（010）88518169

E-mail：infopower@cepp.com.cn

印　　刷：北京密云红光印刷厂

开本尺寸：185×260　　印　　张：17.5　　字　　数：427千字

书　　号：ISBN 978-7-5083-5374-6

版　　次：2007年6月北京第1版

印　　次：2007年6月第1次印刷

印　　数：0001—4000册

定　　价：27.00元

序

软件工程无论是在预算上还是在工期上，其设计工程、制造工程和测试工程之比应为2:1:2，尤其是文档资料的编写上需花费相当大的代价。因为文档除了能起到向下一步工程传达设计意图和设计内容的作用以外，还有以下三点重要作用：

（1）用户方和开发方的责任依据。

（2）大规模团队开发的信息依据。

（3）软件维护和再开发的设计依据。

正因为如此，国外软件中设计工程和测试工程的人均单价为制造工程的人均单价的 1.45倍以上，自然也构成了软件业的价值观。

与国外软件业相比，我国软件业中普遍存在轻视设计工程和测试工程而重视制造工程的倾向。虽然近十年来，我国很重视培养计算机软件开发人才，但是90%都是位于制造工程层次的人才。这种人才结构，首先决定了我国独自研发的软件产品市场成功率低，市场寿命短的现状。解决我国软件开发人才结构问题的关键在于高等院校的计算机软件专业的教学改革和人才结构的调整以及IT企业人才教育类型的改革。

随着计算机应用的普及和发展，软件开发呈现出国际化的特征。最近几年，我国的软件出口增长较快，特别是沿海地区对日本的软件出口更加显著，其中绝大部分属于承接来自国外特别是日本的软件外包业务，许多软件企业在这种势态下不断发展壮大。因此，外包型软件企业迫切需要掌握国外软件开发技术的人才。

张凯编写的“软件工程与实践”一书，以作者在国外软件业工作的丰富经验，比较详细地介绍了在国外实际应用的软件工程方法，以软件开发为主线分别阐述了系统分析、设计、编码和测试等内容。内容紧贴工程实际应用，体现了工程中成熟的技术和工程案例，是一本在软件工程领域能够理论联系实际，有一定理论水平和工程使用价值的好书。这种内容及其组织方法在我国已出版的软件工程专著和教材中还不多见。因此，此书的出版可以为我们树立应用型著作的典范

谭浩强

2007年6月

前　言

自1960年所提倡的“软件危机”以来，软件工学已诞生约40年了。随着软件开发的需要，各种研究不断地被推进，为软件开发所活用。软件开发面临着软件大规模化危机到来，就要追求生产性向上。因此，就造成开发工作标准化、包装化、软件工具自动化和软件的大规模化及交货期的短缩。

本书将软件工学理论与软件开发实践相结合，详述了其关键重要的技术。不仅适用于学习软件工程的本科及研究生，也面向有2～3年编程经验的程序员及由程序员走向系统工程师的工程技术人员。本书的目的并不是单纯追求软件工程理论的前沿，而是一本为培养能胜任软件开发和应用工作的各层次实用型、复合型人才编写的应用性教材。其五大特点总结如下：

（1）基于软件工学基础技术事项，融合现场的软件开发技术，以国外软件开发的基础和实践经验为中心来说明。

（2）本书贯穿了软件开发全过程，包括需求分析、外部设计、内部设计、编码、测试和维护。并对面向对象的基本事项、项目管理及质量管理等重要内容进行了介绍。

（3）各章配有习题，可以帮助读者总结、归纳。

（4）本书作为软件工程的实用型教材，内容比较完整。

（5）本书配有设计文档模版和实训项目，以便学生能掌握本书教学内容，并达到使之和实践真正结合的目的。

此外，本书由以下六篇组成。其中第一、二篇为后续各章进入具体设计流程之前，而为读者介绍了关于软件、软件工程及软件开发方法的基础；第三篇用了4章内容分别介绍了软件项目为了使各种不同角色的工作人员相互合作，需建立团队、制定计划和管理，并详细介绍了从需求分析、外部设计、内部设计、编码、测试到维护的各软件设计过程及设计中的实战经验；第四篇介绍了软件质量保证建立了一套有计划，有系统的方法及完备的设计文档、评审，来向管理层保证拟定出的标准、步骤、实践和方法能够正确地被所有项目所采用；第五篇列举了软件开发面象对象开发的动向、XP编程、网上服务、安全技术、活用软件包开发、净室方法、软件构成管理等最近的话题。第六篇主要介绍了实践项目。

本书是高校研究生、本科生、高职高专学生和在职专业人员学习软件工程的教材。可以全学，也可以根据读者所学要求的不同，选学其中各部分章节。建议研究生和在职专业人员可全选，本科生、高职高专学生可以第三篇、第四篇为重点进行教学；建议本书教学为30学时，本书所配项目实训为48学时。

欢迎使用本书。本书作者期待您的意见、批评、指正和建议。

编　者

2007年6月

目　　录

第三篇 软件工程设计与实践

第四篇 软件质量与质量保证

第五篇 软件开发的发展

第六篇 实 践 练 习

第一篇　软件与软件工程

第1章　软　　件

软件提供了我们这个时代最重要的产品——信息。软件可以处理个人数据，使这些数据在局部范围内更为有用；它可以管理商业信息以增强竞争力，它提供了通往全球信息网络的通路，它还提供了可以用各种形式获取信息的手段。

1.1　软件的发展

人们常说的计算机系统是由硬件和软件两大部分组成的。如果用计算机求解问题，程序是不可缺少的，程序被称为软件的实体部分，程序只有在硬件载体上运行才可获得所求问题的解，因此硬件和程序是求解问题的最基本条件。然而，只有程序也会给使用者带来很多不便，好的程序应有相应的文字资料，如各种规格说明书、设计说明书、用户手册等，我们称这些文字资料为文档。文档不但对使用者是必要的，而且对程序开发者更是至关重要的。特别是由多个人经过多年才能完成开发任务的大型程序，开发者与使用者之间都需要有规范的书面文档规定程序的功能、使用环境、使用方法等。所以，严格地说，程序和软件是两个不同的概念。程序是指计算机可识别的源程序代码或机器可直接执行的程序代码；而软件是指程序加上开发、使用和维护该程序所需要的全部文档。程序是软件的实体，而文档是软件的重要组成部分，这个公认的浅显认识囊括了软件发展的三个阶段。

软件的发展大致可分为三个时期，即程序时期（20 世纪 50 年代～60 年代初）、软件=程序+说明时期（20 世纪 50 年代末～70 年代初）、软件=程序+文档时期（20 世纪 70 年代初至今，也称软件工程时期）。如表 1-1 所示。

表 1-1　计算机软件发展的三个时期及其特点

时间 / 特点	程序设计 20 世纪 50～60 年代	程序系统 20 世纪 60～70 年代	软件工程 20 世纪 70 年代以后
软件所指	程　序	程序及说明书	程序、文档、数据
主要程序设计语言	汇编语言及机器语言	高级语言	软件语言
软件工作范围	程序编写	包括设计和测试	软件生存期
需求者	程序设计者本人	少数用户	市场用户

续表

特点 \ 时间	程序设计 20 世纪 50～60 年代	程序系统 20 世纪 60～70 年代	软件工程 20 世纪 70 年代以后
开发软件的组织	个　人	个人、开发小组	个人、开发小组、大中型软件开发
软件规模	小　型	中小型	大中小型
决定质量的因素	个人程序技术	小组技术水平	管理水平
开发技术和手段	子程序	结构化程序设计	数据库、开发工具、开发环境、工程化开发方法、标准和规范、网络及分布式开发、面向对象技术
维护责任者	程序设计者	开发小组	专职维护者
硬件特征	价格高、存储容量小、工作可靠性差	价格低、速度、容量及工作可靠性有明显提高	向超高速、大容量、微型化及网络化方向发展
软件特征	完全不受重视	软件技术的发展不能满足需要，出现软件危机	开发技术有进步，但没取得突破性进展，价格高，未完全摆脱软件危机

这里所说的程序是指在计算机上可执行的代码序列。在程序时期，使用者需直接操作计算机硬件，因而一开始就形成了计算机硬件就是计算机这一概念。这一时期，计算机工作者致力于改善硬件结构和可靠性研究，而仅把程序作为机器运行时必须进行的准备工作。每个程序都是为求解某个特定问题而专门设计的，不考虑程序的通用性，更没意识到程序同计算机硬件一样，也是计算机系统不可分割的部分。这一时期的程序设计工作全凭借设计者个人经验和技术独立进行，是一种典型的手工艺智力劳动。程序中出现的错误只能由设计者本人才能修改，因而也无需向他人作任何交待和说明，当时人们脑海中只有程序概念而无软件概念。

20 世纪 50 年代末至 70 年代初这段时期，由于硬件技术的飞速发展，与此同时相继问世了一批高级程序设计语言（如 FORTRAN、ALGOL60、COBOL、BASIC 等）的编译程序、解释程序以及操作系统等支持程序（也称系统程序）和具有较强通用性的应用程序，使得计算机应用从单一的科学计算，迅速发展到数据处理和实时控制等各个应用领域。为使用户方便地使用这些支持程序和通用应用程序，必须提供相应的技术指南、用户手册等说明性文字资料。因而出现了“软件=程序+说明”这一概念。这个时期的程序特征主要表现在三个方面：

（1）由于程序规模较大，需多人协作才能完成程序编写，我们把这种方式称为“作坊式”生产方式。

（2）程序的设计与运行维护再也不能由一个人来承担。

（3）人们已认识到程序再也不是计算机硬件的附属成分，而是计算机系统中与硬件相互依存的不可缺少的部分。

这个时期软件技术取得了很大发展，比较有代表性的是多道程序设计技术、多用户人

机交互系统、实时系统以及文件管理等。同时出现了更多的通用和专用程序设计语言，在形式语言理论、编译理论、数据库理论等方面也取得了重大突破，从而诞生了真正的计算机科学。

随着投入市场运行的计算机数量的日益增加，计算机应用领域越来越广，软件需求量越来越大。因而，美国和西欧一些发生国家相继建立起庞大的软件公司，专门生产计算机软件，一种新兴产业——软件产业应运而生。

虽然软件需求量不断增加，其复杂度也越来越高，但其生产方式大多停留在手工“作坊式”生产阶段。这种方式不仅效率低，而且软件质量差，如 IBM 公司 20 世纪 60 年代开发的 OS/360 系统，耗资几千万美元，用了 5000 多人，拖延几年才交付使用，交付使用后每年发现近 100 个错误。OS/360 系统开发负责人 Brooks 生动地描述了研制过程中的困难和混乱：“……像巨兽陷入泥潭作垂死挣扎一样，挣扎得越猛，泥浆就沾得越多，最后没有一个野兽能逃脱淹没在泥潭中的命运……。程序设计就像是这样的泥潭，一批批程序员在泥潭中挣扎……。没有料到问题会这样棘手……”。比 OS/360 更糟的软件系统并不少，即花费大量的人力、财力，结果半途而废，或完成之日即是遗弃之时。这就是人们常说的“软件危机”期。

为了摆脱这一困境，软件工作者一方面研究程序设计方法和程序正确性验证方法，另一方面寻找工程化的软件系统开发方法。

以 E. W. Dijkstra 提出的“结构化程序设计方法”为代表，引导人们对程序设计方法的广泛研究。在短短几年时间内就提出了模块化、结构化、自顶向下逐步求精、程序变换、程序的推理与综合、数据类型抽象、符号测试、程序正确性证明等各种程序设计和验证方法。虽然有些方法至今仍作为理论上的探讨，但这一时期的研究成果使软件设计者深刻认识到，没有科学的方法作指导，不可能生产出高质量的软件产品，同时还提出了以正确性、结构清晰、便于设计、易于测试和维护作为衡量软件质量优劣的重要标准。

1.2　软件的定义

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，它是现代科学研究和工程问题寻求解答的基础，它也是鉴别现代产品和服务的关键因素。它被嵌入到各种类型的系统中，如交通、医疗、电信、军事、工业生产过程、娱乐、办公……。软件在现代社会中是必不可少的。而且进入 21 世纪软件将成为所有领域新进展的驱动器。

1.2.1　软件的含义

软件一词具有三层含义。一为个体含义，即指计算机系统中的程序及其文档；二为整体含义，即指在特定计算机系统中所有上述个体含以下的软件的总称，亦即计算机系统中硬件除外的所有成分；三为学科含义，即指在研究、开发、维护以及使用前述含义下的软件所涉及的理论、方法、技术所构成的学科。

1. 软件的特点

理解软件（以及最终理解软件工程）前，先了解软件的特点是很重要的，这样能够明白软件与人类建造的其他事物之间的区别。当建造硬件时，人的创造性过程（分析、设计、建造、

测试）最终被转换成有形的形式。如果我们建造一个新的计算机，初始的草图，正式的设计图纸和试验版的原型一步步演化成为一个有形的产品（芯片、线路板、电源等等）。

而软件是逻辑的而不是有形的系统元件，因此，软件具有与硬件完全不同的特征。

软件是被开发或设计的，而不是传统意义上被制造的。虽然在软件开发和硬件制造之间有一些相似之处，但两种活动在本质上是不同的。对于这两种活动，都可以通过良好的设计获得高质量，但硬件在制造过程中可能会引入质量问题，这种情况对于软件而言几乎不存在（或是很容易改正）；两者都依赖于人，但参与的人和完成的工作之间的关系不同；两种活动都要建造一个“产品”，但方法不同。

软件成本集中在开发上，这意味着软件项目不能像制造项目那样管理。

2. 软件不会“磨损”

图 1-1 描述了作为时间的函数的硬件故障率。其关系常常被称作“浴缸曲线”，表明了硬件在其生命初期有较高的故障率（这些故障主要是由于设计或制造的缺陷）。这些缺陷修正之后，故障率在一段时间内会降到一个稳定的水平上（理想情况下相当低）。然而，随着时间的流逝，故障率又提升了，这是因为硬件构件由于种种原因会不断受到损害，例如灰尘、振动、不合理的使用、温度的急剧变化以及其他许多环境问题。简单地讲，硬件已经开始磨损了。

软件并不受到这些引起硬件磨损的环境因素的影响。因此，理论上，软件的故障率曲线呈现出如图 1-2 所示的“理想曲线”形式。未发现的错误会引起程序在其生命初期具有较高的故障率，然而，当这些错误改正以后（理想情况下，不引入其他错误），曲线就如图所示那样趋于平稳。理想曲线是软件的实际故障模型非常粗略的简化，但其含义很清楚——软件不会磨损，不过它会退化。

这个说法表面上似乎是矛盾的，我们可以通过图 1-2 中的实际曲线来解释清楚。在其生存期间，软件会经历修改，随着这些修改，有可能会引入新的错误，使得故障率曲线呈现为图 1-2 所示的锯齿形。在该曲线能够恢复到原来的稳定状态的故障率之前，又需要新的修改，又引起一个新的锯齿。慢慢地，最小故障率水平就开始提高了——软件的退化是由于修改引起的。

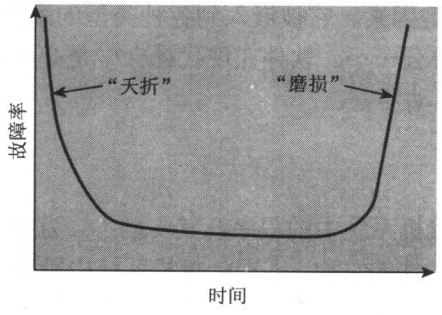

图 1-1　硬件的故障曲线

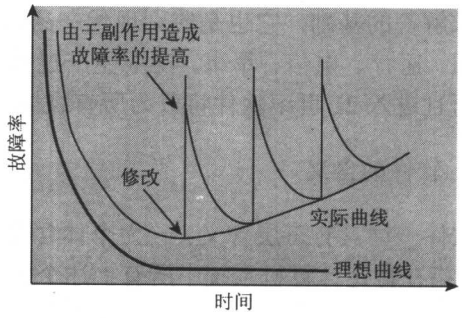

图 1-2　软件的理想故障曲线和实际故障曲线

关于磨损的另一方面也表明了硬件和软件之间的不同。当一个硬件磨损时，可以用另外一个备用零件替换它，但对于软件就没有备用零件可以替换了。每一个软件故障都表明了设计或是将设计转换成机器可执行代码的过程中存在错误。因此，软件维护要比硬件维护复杂得多。

3. 大多数软件仍是定制的

考虑一个基于计算机的产品的控制硬件被设计和建造的方式。设计工程师画一个简单的数字电路图，做一些基本的分析以保证可以实现预定的功能，然后查询所需的数字零件的目录。每一个集成电路（通常称为“IC”或“芯片”）都有一个零件编号，一个定义的和确认过的功能，定义好的接口和一组标准的集成指南。每一个选定的零件，你都可以在货架上买到。

标准的螺丝钉和购买的集成电路仅仅是机械和电气工程师设计新系统时所使用的数千种标准构件中的两种。可复用的构件被创建，使得工程师可以专注于设计真正创新的部分，即设计中表示某种新东西的部分。在硬件世界，构件复用是工程过程自然的一部分。而在软件世界，它是刚刚开始起步的事物。

软件构件应该被设计和实现已使得它能够被复用于很多不同的程序。在 20 世纪 60 年代，我们创建了科学子例程序，它们是在一个广泛的工程和科学应用领域内可复用的。这些子例程序以一种有效的方式复用定义良好的算法，但是，其应用有限。今天，我们已经扩展我们复用的视角到包含算法和数据结构。现代的可复用构件封装了数据和应用于数据的处理，使得软件工程师能够从可复用的部件创建新的应用。例如，今天的图形用户界面是使用可复用的构件创建的，这些构件设计图形窗口、下拉菜单和大量的交互机制。建造界面所需的数据结构和处理细节被包含在界面构造的可复用构件库中。

1.2.2　软件的种类

软件可以应用于任何情况，只要定义了一组预先定义好的程序步骤（即一个算法，但也有例外，如专家系统和人工神经网络软件）。信息的内容和确定性是决定一个软件应用特性的重要因素。

信息的内容指的是输入/输出信息的含义和形式，例如，许多商业应用使用高度结构化的输入数据（一个数据库）且产生格式化的输出“报告”，而控制一个自动化机器的软件（如一个数控系统）则接受限定结构的离散数据项并产生快速连续的单个机器指令。

信息的确定性指的是信息的处理顺序及时序的可预定性。一个工程分析程序接受预定顺序的数据，不间断地执行分析算法，并以报告或图形格式产生相关的数据。这类应用是确定的。而一个多用户操作系统则接受可变化内容和任意时序的数据，执行可被异常条件中断的算法，并产生随环境和时间的某个函数变化的输出，具有这些特点的应用是非确定的。

在某种程度上我们难以对软件应用给出一个通用的分类。随着软件复杂性的增加，其间已没有明显的差别。下面是列出的常用软件应用领域。

（1）系统软件。系统软件是一组为其他程序服务的程序。一些系统软件（如编译程序、编辑程序和文件管理实用程序）处理复杂的但也是确定的信息结构，其他的系统应用（如操作系统、驱动程序等）则处理大量的非确定的数据。无论哪种情况，系统软件均具有以下特点：与计算机硬件频繁交互，多用户支持，需要精细调度、资源共享及灵活的进程管理的并发操作，复杂的数据结构，多外部接口。

（2）实时软件。管理、分析、控制现实世界中发生的事件的软件成为实时软件。实时软件的组成包括一个数据收集部件，负责从外部环境获取和格式化信息；一个分析部件，负责将信

息转换成应用所需的形式；一个控制/输出部件，负责相应外部环境；一个管理部件，负责协调其他各部件使得系统能够保持一个可接受的实时响应时间。

（3）商业软件。商业信息处理是最大的单个软件应用领域。离散的“系统”（如工资管理、账目支付和接收、库存管理等）均已演化为管理信息系统（MIS）软件，它们可以访问一个或多个包含商业信息的大型数据库。该领域的应用将已有的数据重新构造，变换成一种能够辅助商业操作和管理决策的形式。除了传统的数据处理应用之外，商业软件应用还包括交互式计算（如 POS 事务处理）。

（4）工程和科学计算软件。工程和科学计算软件的特征是其“数值集中”的算法。此类应用覆盖面很广，从天文学到地质学、从汽车压力分析到动力学、从生物学到自动化制造。不过，现代工程和科学计算软件已不仅限于传统的数值算法。计算机辅助设计、系统仿真和其他交互应用已经开始具有实时软件和系统软件的特征。

（5）嵌入式软件。智能产品几乎在每一个消费或工业市场上都已经变成寻常产品。嵌入式软件驻留在只读内存中，用于控制这些消费和工业市场上的智能产品和系统。嵌入式软件能够执行很有限但专职的功能（如微波炉的面板控制）或提供重要的功能及控制能力（如汽车中的数字控制，包括燃料控制、仪表版显示、刹车系统等）。

（6）个人计算机软件。个人计算机软件市场是在过去 20 年中产生和发展起来的。文字处理、电子表格、计算机图形、多媒体、娱乐、数据库管理、个人及商业金融应用、外部网络和数据库访问等。

（7）基于 Web 的软件。浏览器检索的 Web 页面是软件，它结合了可执行的指令（如 CGI、HTML、Perl 和 Java）和数据（如超文本和一系列可视及音频格式的数据）。本质上，网络变成了一个巨大的计算机，提供了一个几乎是无限的、可被任意人访问的软件资源。

（8）人工智能软件。人工智能（AI）软件利用非数值算法去解决复杂的问题，这些问题不能通过计算或直接分析得到答案。专家系统（也称为基于知识的系统）、模式识别（图像或声音）、人工神经网络、定理证明和游戏是该应用领域的典型代表。

1.3 软 件 危 机

软件危机是指在计算机软件开发和维护过程中所遇到的一系列问题，例如：

（1）不能正确地估计软件开发成本和进度，导致实际开发成本往往高出预算很多。

（2）软件产品不可靠，满足不了用户的需求，甚至无法使用。

（3）交付使用的软件不易演化，以至于人们不得不重复开发类似的软件。

（4）软件生产率低下，远远满足不了社会发展的需要。

……

产生软件危机的原因很多，除了与软件本身固有的特征有关以外，还与软件开发模型、软件设计方法、软件开发支持以及软件开发管理等有关。

程序设计方法和程序正确性验证方法的研究，使程序设计走向更科学的道路，在一定程度上提高了软件可靠性，但设计过程采用的低效手工方式，周期长、代价高，远远不适应软件生产的需要。为此，北大西洋公约组织成员国的软件工作者于 1968 年、1969 年连续两年召开了软件研究会（即 NATO 会议）。集中研究对策，讨论如何摆脱软件危机。考虑到软件

系统开发过程同制造一台机器或建造一栋大厦有许多相似之处，于是提出了“软件工程”这个术语，试图用“工程化”的思想作指导来解决软件研究中面临的困难和混乱，从而解决软件危机的困境。

习　题

1. 描述软件的含义。
2. 什么是软件危机？
3. 软件的种类有哪些？

第2章 软件工程

2.1 软件工程的定义

软件工程（Software Engineering）这个名词是北大西洋公约组织（NATO）科学技术委员会于1968年秋天在当时的联邦德国召集了近50名一流的编程人员、计算机科学家和工业界巨头，制定摆脱软件危机的办法时所提出的。尽管当时专家们无法设计出一张指导软件业走向更牢固阵地的详细路线图，但他们借鉴硬件工程的办法来解决软件这一难题，这不仅创造了一个新名词——软件工程，还使软件工程有了方向。从1968年到现在已经30多年了，应该说今天软件工程已发展成为一门独立的学科。

软件工程可以理解为运用工程学的原理和方法来组织和管理软件的生产和维护，以保证软件产品开发、运行和维护的高质量和高生产率。

1993年，IEEE在《IEEE Standard Collection》对“软件工程”这个概念给出了以下全面的定义：

用系统、规范及可量化的方法去开发、运行和维护软件，即软件的工程化应用及研究。

2.2 软件工程开发模式

软件工程应用计算机科学、数学及管理科学的原理，借鉴传统工程的原则、方法，创建软件以达到提高质量，降低成本的目的。软件工程也是一门指导计算机软件开发和维护的工程学科。

软件开发过程模型确定的是软件开发的宏观过程框架，要保证开发活动的高质量，还必须有相应的软件开发方法作为技术支持。软件开发方法是具体软件开发活动中应用的技术。

软件开发过程模型是指开发软件项目的总体过程思路。最传统的最早的软件开发模型是瀑布模型，随着软件工程技术的不断发展，在软件开发实践中，出现了许多新的或改进的软件开发过程模型，比较常见的有原型模型、螺旋模型等。

2.2.1 瀑布模型

图2-1是典型的瀑布模型（Waterfall Model）示意图。瀑布模型主要包括开发和确认两个过程。

（1）开发过程是严格的下导式过程，各阶段间具有顺序性和依赖性，前一阶段的输出是后一阶段的输入，每个阶段工作的完成需要审查确认。

（2）确认过程是严格的追溯式过程，后一阶段出现了问题要通过前一阶段的重新确认来解