软件工程

RUANJIAN GONGCHENG

主　编　郑逢斌

副主编　阎朝坤　房彩丽

罗慧敏　辛　明

科学出版社

软件工程

主　编　郑逢斌

副主编　阎朝坤　房彩丽
　　罗慧敏　辛　明

科学出版社
北京

内 容 简 介

本书全面系统地讲述了软件工程的概念、原理和典型的方法，以及软件项目的管理技术和软件工程的新方法。主要内容包括软件生命周期各阶段的任务、过程、结构化方法和面向对象方法，软件项目管理相关技术及工具，软件工程应用中的新技术等。着重介绍面向数据流的系统分析和设计，面向对象的分析和设计，面向对象基础和 UML。

本书在介绍面向数据流和面向对象的设计时，紧密围绕实例进行阐述，对读者深入理解软件工程学很有帮助，可以作为学生综合实验前的练习。

本书可作为高等院校“软件工程”课程的教材或教学参考书，也可作为软件开发人员和软件项目管理人员的参考书。

图书在版编目(CIP)数据

软件工程/郑逢斌主编. —北京: 科学出版社, 2012. 8

ISBN 978-7-03-034583-7

Ⅰ. ①软… Ⅱ. ①郑… Ⅲ. ①软件工程 Ⅳ. ①TP311.5

中国版本图书馆 CIP 数据核字 (2012) 第 191643 号

责任编辑: 于海云 张丽花 / 责任校对: 刘小梅

责任印制: 闫 磊 / 封面设计: 迷底书装

科学出版社 出版

北京东黄城根北街 16 号

邮政编码: 100717

http://www.sciencep.com

铭浩彩色印装有限公司 印刷

科学出版社发行 各地新华书店经销

*

2012 年 8 月第 一 版 开本: 787×1092 1/16

2012 年 8 月第一次印刷 印张: 21 1/2

字数: 546 000

定价: 46.00 元

(如有印装质量问题, 我社负责调换)

前　言

软件工程是计算机学科中一个年轻并且充满活力的研究领域。自20世纪60年代末以来，人们为克服“软件危机”在这一领域做了大量工作，逐渐形成了系统的软件开发理论、技术和方法，它们在软件开发实践中发挥了重要作用。今天，计算机软件扮演着十分重要的角色，软件工程已成为信息社会高技术竞争的关键领域之一。

“软件工程”是高等学校计算机专业教学计划中的一门核心课程，主要包括支持软件开发和维护的理论、方法、技术、标准，以及计算机辅助工具和环境。这些内容对于软件研制人员、软件项目管理人员都是必需的。本书比较全面、系统地反映了软件工程课程的全貌，既介绍了传统的、实用的软件开发方法，又详细阐述了面向对象的分析和设计这一新兴的技术和方法。

本书共16章，第1章是软件工程概述，第2~14章讲述软件生命周期各阶段的任务、过程、结构化方法和面向对象方法(其中第5章和第9章着重介绍面向数据流的系统分析和设计方法，第7章和第10章介绍面向对象的分析和设计方法，第6章介绍面向对象基础和UML)，第15章讲述软件项目管理技术及管理工具，第16章介绍软件工程应用中的一些新技术。

本书可作为计算机专业本科生和硕士生“软件工程”课程的教材，其内容满足国家教育委员会颁布的计算机专业软件工程课程教学基本要求。同时，本书也可作为软件开发人员与软件项目管理人员的技术参考书。面向数据流和面向对象的分析和设计方法在本书中有详细阐述，占据本书第5~10章。其中第5~7章介绍分析方法，第8~10章介绍设计方法。在教学过程中，可以按章节顺序依次讲授，也可以按照第5章→第8章→第9章，第6章→第7章→第8章→第10章这两个顺序分别进行讲授。

本书根据编者多年在校内外讲授“软件工程”课程所用的讲义改写而成，充分考虑了教学中学生的建议和要求，并增加了一些最新的素材。书中大量引用了国内外众多学者的研究成果与相关资料，没有他们的创造性工作，就不可能有这本书的问世，编者对他们表示崇高的敬意和感谢。

郑逢斌教授组织本书的编写工作，负责统稿和定稿。各部分分工如下：阎朝坤编写第1、6章；房彩丽编写第12、14章；罗慧敏编写第2、10章；辛明编写第5、9章；常继科编写第7、15章；杜莹编写第3、8章；陈小潘编写了第4、16章；孙洁编写第13章；徐素锦、梁文娟编写第11章。感谢学院领导和相关任课老师对本书编写的大力支持。

同时，感谢科学出版社的各位领导和编辑，他们为本书早日与读者见面耗费了大量的心血。

鉴于编者的水平有限，书中疏漏和不足在所难免，敬请读者批评指正。

编　者

2012年5月

目　录

第 1 章　软件工程概述

随着计算机技术的飞速发展，计算机已经被广泛地应用于科学计算、工程设计、数据处理、自动控制、通信、互联网、计算机辅助设计与制造、计算机辅助教学等涉及人们工作和日常生活的广大领域。相对于计算机硬件的迅猛发展，软件技术的发展明显滞后且存在“瓶颈”，软件的可靠性无法得到保障、维护费用不断上升、开发进度和成本难以控制。人们用“软件危机”一词来形容当今软件开发行业面临的困境。为了扭转这种局面，人们期望用类似工程的方法来解决软件开发中碰到的各种困难，软件工程学应运而生。

1.1　软 件 概 述

1.1.1　软件的定义

“软件”一词是在 20 世纪 60 年代初从国外的组合词 software 翻译过来的，有人翻译成“软质品”，也有人翻译成“软体”，目前国内统一称为“软件”。一些人认为“软件就是程序”，将软件定义为“是一系列按照特定顺序组织起来的计算机数据和指令的集合”。在一些词典中将软件解释为“控制计算机硬件功能及致使其运行的程序、自程序和符号语言”。

软件是计算机系统中与硬件相互依存的另外一个重要部分，是包括程序、数据及其相关文档的完整集合，这是早期对软件的一种普遍解释。其中，程序是依据事先设计的功能和性能要求研制、执行的一组指令序列；数据是使程序能正常操纵信息的数据结构；文档是与程序开发、维护和使用相关的一系列图文材料。

在 IEEE 软件工程词汇标准中，软件的定义是：软件是计算机程序、规程以及运行计算机系统可能需要的相关文档和数据。其中，计算机程序是计算机设备可以接受的一系列指令和说明，为计算机执行提供所需的功能和性能；数据是事实、概念或指令的结构化表示，能被计算机接收、理解或处理；文档是描述程序研制过程、方法及使用的图文材料。

Wirth 在结构化程序设计中指出：程序=算法+数据结构。在软件工程中，软件=程序+文档。

虽然以上对软件定义的表述不同，但本质上它们是基本一致的，只是各自的侧重点不一样而已。

1.1.2　软件的特点

在一个计算机系统中，软件是与硬件相互依存的。相对于硬件来说，软件也有一些自身特点。

(1) 软件和硬件不同，它不是看得见、摸得着的物理实体，而是一种逻辑实体，具有抽象性。软件可以记录并存储在不同的物理介质中，但无法直接看到其具体形态，必须通过观察、分析、思考、判断，利用抽象思维能力去了解其功能、性能等特征。

(2) 软件的生产不存在明显的制造过程，不像硬件那样，一旦研制成功就可以批量重复

制造，在制造过程中控制硬件产品的质量。而软件需要通过人们的智力活动，把知识与技术转化成信息，通过研制、开发的过程被创造出来。我们常说的软件故障往往就是在开发阶段产生的、而在测试阶段没有被检测出来的问题。所以要保证软件的质量，必须着重在软件开发方面下功夫。

(3) 在软件的运行和使用的过程中，也不会存在像硬件那样的机械磨损及老化问题。任何机械、电子设备在运行和使用中，其失效率的趋势大都遵循一种U形曲线的形态，一开始制造出来的新设备往往具有较高的失效率，但随着不断运行、磨合、调整，失效率逐步降低，随后处于相对稳定状态。但是当硬件设备使用年限过久时，由于存在设备磨损和老化的问题，失效率又大幅度上升。软件不存在磨损及老化的问题，但存在“退化”问题，软件在真正投入使用后的一段时间内，还需要多次对软件进行修改以满足用户新的需求。这种修改过程是多次反复发生的，一直到软件彻底不用为止。由于在每次修改后可能又会引入一些新的问题，如软件bug，因而软件的失效曲线往往表现为一种“锯齿状”的走势。

(4) 软件的开发和运行往往受限于具体的计算机系统，对计算机系统有着不同程度的依赖性，对软件的通用性造成了一定影响。为了消除这种依赖性，“软件移植”的概念被提出并应用到软件开发中。

(5) 目前，软件的开发模式还没有完全脱离手工开发方式，大部分软件产品都属于“定制”类型，还不能完全采用组装的方式进行软件开发。伴随着软件技术的发展，一些提高软件开发效率和质量的新方法应运而生，如软件复用、设计模式、模型驱动体系结构(Model Driven Architecture，MDA)、基于构件的软件体系结构与开发方法等。然而，由于软件开发工作本身就是一种高强度的脑力劳动，开发人员必须依靠自己的智力去理解需求，满足需求，并综合运用软件技术来提高开发效率和质量，无法完全使软件开发过程自动化。

(6) 软件作为一种提高人类工作效率的逻辑产品，本身是非常复杂的。软件开发，特别是应用软件的开发常常涉及多种领域的知识，这对软件人员的知识素养提出了较高要求。另外，软件的复杂性与软件开发技术的发展越来越不相适应，表现为：软件开发技术的发展远远落后于软件复杂性需求，且随着时间的推移，这种差距日趋明显。

(7) 软件成本相当昂贵。软件的研发需要投入大量的、高强度的智力劳动，成本比较高。相对而言，计算机硬件技术的不断发展导致在一个计算机系统中硬件成本的比例逐步下降，而软件成本的比例逐步增多，即软件的价值比例在不断提高。

(8) 相当多的软件工作涉及社会因素。许多软件的开发和运行涉及机构、体制及管理方式等问题，甚至涉及人的观念和人们的心理，这些直接影响项目的成败。

1.1.3 软件的分类

事实上，很难对软件进行科学有效的分类，但在实际生活中，不同类型的软件在开发过程、方法和工具的选择上确实存在一些差异，有必要对各类软件的特征有一定了解。因此，下面从不同的角度对软件类型进行划分。

1. 按功能特征进行划分

(1)系统软件。系统软件是计算机系统必不可缺的组成部分，在计算机实际运行过程中，系统软件需要频繁地与硬件交互，负责各资源共享及进程管理、复杂数据结构的处理等工作。典型的例子有操作系统、数据库管理系统、设备驱动程序等。

(2)支撑软件。支撑软件是协助用户软件开发的工具软件，其中包括帮助程序人员开发软件产品的工具，也包括帮助管理人员控制开发进程的工具。

(3)应用软件。应用软件是在特定领域内开发，为特定目的服务的一类软件。现在计算机已经渗透到各行各业、各个领域。这些领域的应用软件种类繁多，如商业数据处理、计算机辅助设计/制造(CAD/CAM)、中文信息处理、计算机辅助教学(CAI)、计算机网络等软件。

2. 按规模大小进行划分

软件研发所面临的问题和领域是各不相同的，如科学计算的软件(如天气预报软件、材料分析软件)与一般应用软件(酒店管理系统、学生管理系统等)在问题复杂型、计算数据量、可靠性需求等方面都是不一样的。相应地，软件的研发过程、人员财力的投入等也会有较大的区别。根据开发软件过程中所需的人力、研制期限以及最终软件程序包含的源代码行数，可将软件划分成 6 种不同规模的软件，如表 1-1 所示。

表 1-1　软件规模的分类

类别	所需人力	研制期限	产品规模(源代码行数)
微型	1	1~4 周	0.5k
小型	1	1~6 月	1k~2k
中型	2~5	1~2 年	5k~50k
大型	5~20	2~3 年	50k~100k
甚大型	100~1 000	4~5 年	1M(=1 000k)
极大型	2 000~5 000	5~10 年	1M~10M

规模大、开发周期长、项目参与人员多的软件项目，其开发工作必须得有软件工程的理论知识做指导。而规模小、开发周期短、参加人员少的软件项目也必须有软件工程的概念，遵循一定的开发规范。总的来说，基本原则都是一样的，本质上的不同在于它们对软件工程技术依赖程度的多少。

3. 按工作方式进行划分

(1)实时处理软件。在一些应用领域中，一旦事件发生或数据产生，就需要对其及时处理并反馈，这类监测和控制系统执行过程的软件称为实时处理软件。一般来说，实时处理软件主要包括数据采集、数据分析、结果输出三大部件，处理时间是有严格限定的，如果实际处理时间超出了这一限制，将影响后续系统的运行，甚至造成严重事故。

(2)分时软件。分时软件可以允许多个联机的用户同时使用计算机，系统将处理机时间划分成一个一个的时间片，将这些时间片轮流分配给各联机用户，每个用户都感到只有自己完全占用计算机。

(3)交互式软件。一些软件系统在运行过程中需要接受用户的信息，这种能实现人机通信的软件称为交互式软件。不同于实时处理软件，这类软件在时间上没有严格的限定。随着计算机逐渐深入到人们工作生活的方方面面，交互式软件的设计在考虑人机界面友好性的同时不得不考虑界面配置的灵活性，特别是基于 Web 的应用。

(4)批处理软件。批处理软件是把一组输入作业或一批数据以成批处理的方式一次运行，按顺序逐个处理完的软件，这是一种最为传统的工作方式。

4. 按应用范围进行划分

(1)通用软件。通用软件是指由一些专门的软件开发组织开发的、面向市场用户公开销售的独立运行系统，像操作系统、数据库管理系统、字处理软件、绘图软件包和项目管理工具等都属于这种类型。

(2)定制软件。与通用软件不同，定制软件是受某个特定客户委托、软件开发组织按照与客户签订的开发合同进行开发的软件，像一些商业 ERP 系统、卫星控制系统和交通指挥系统等都属于这种类型。

5. 按使用频度进行划分

软件的使用频度指的是对某一软件使用的频率，有的软件开发出来只能使用一次，不具备再次使用的价值，例如用于人口普查、工业普查的软件。在实际生活中，这类相关普查工作往往间隔时间很长，下一次普查时，以往开发的普查软件往往会因为新环境、新因素的变化很难再适用。对于一些企业部门来说，有的统计资料或试验数据是按照年度进行统计分析的，因而统计分析软件也只是每年运行一次。相对而言，有些软件具有较高的使用频度，如天气预报软件。显然，不同使用频度的软件，有不同的要求，不可同等对待。

6. 按失效影响进行划分

不同的软件由于其应用的场景不同，对软件失效的关注程度也会不同。通常，一些软件如果在运行中出现了问题，造成软件失效，可能会丢失一些数据或耽误业务的进展，这对客户来说，影响不是很大。但对于一些特殊的软件，如果出现失效，将有可能酿成灾难性的后果，例如财务金融、交通通信、航空航天等软件，称这类软件为关键软件。

1.1.4　软件的发展

计算机诞生已经有 60 多年了。过去几十年来，计算机硬件技术发展迅速，计算机的性能不断提升，计算机的体系结构也发生了较大变化。同时，计算机软件系统也变得更加复杂，计算机软件的角色也在不断发生变化。总体上说，软件的发展历程大致可以分为如图 1-1 所示的 4 个阶段。

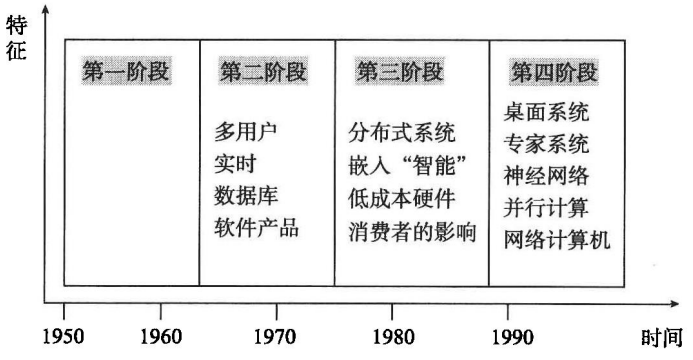

图 1-1　计算机软件发展的 4 个阶段

第一阶段指的是从 20 世纪 50 年代初期到 60 年代初期的十余年。这段时期属于计算机系统开发的初期阶段，其间的软件开发都是针对某个具体应用专门编写的。另外，编写软件

的人和最终的用户也常常是相同的。这种个体化的软件开发环境，使得软件设计的概念只是人们头脑中的隐含过程，不能对其有深入的理解。最终，软件呈现出来的只有程序清单，没有其他文档资料保存下来。

除此之外，早期所开发的计算机系统一般采用了批处理技术，虽然提高了计算机的使用效率，但是不利于程序的设计、调试和修改。在当时，人们认为计算机的主要用途是快速计算，当时的软件编程也相对比较简单，不存在任何系统化的方法，没有任何开发管理，最终软件质量的好坏完全依赖于编程人员的个人技巧。

第二阶段是指从 20 世纪 60 年代中期到 70 年代中期的十余年。在这个阶段的软件有不同的特点，如多用户系统引入了人机交互的概念，实时系统能够从多个数据源收集数据并加以分析和转换，在线存储的发展产生了第一代数据库管理系统。

在这个阶段，软件产品和“软件作坊”的概念开始出现，软件开发人员不再像早期阶段那样只因个人工作需要而开发，而是为了用户更好地使用计算机。然而，“软件作坊”仍然沿用了早期“个体式”软件开发方法。程序运行时发现的错误必须设法更正；用户有了新需求时，必须相应地修改或添加程序；硬件或操作系统更新时，又可能要修改程序以适应新的环境。在这种情况下，软件的维护工作以惊人的比例耗费资源，更严重的是，程序设计的个体化和作坊化特性使软件最终成为不可维护的，从而出现了早期的“软件危机”。

第三阶段是指 20 世纪 70 年代中期到 80 年代末期这段时间。在这个阶段，分布式系统极大地提高了计算机系统的复杂性，网络的发展对软件开发提出了更高的要求，特别是微处理器的出现和广泛应用，孕育了一系列的智能产品。硬件的发展速度已经超过了人们对软件的需求速度，使得硬件价格下降，软件的价格急剧上升，导致了软件危机的加剧，致使更多的科学家着手研究软件工程学的科学理论、方法和时限等一系列问题。软件开发技术的度量问题受到重视，最著名的有软件工作量估计 COCOMO 模型、软件过程改进模型 CMM 等。

第四阶段是从 20 世纪 80 年代末期开始的。这个阶段是强大的桌面系统和计算机网络迅速发展的时期，计算机体系结构由中央主机控制方式变为客户机/服务器方式，专家系统和人工智能软件终于走出实验室进入了实际应用，虚拟现实和多媒体系统改变了与最终用户的通信方式，出现了并行计算和网络计算的研究，面向对象技术在许多领域迅速取代了传统软件开发方法。

1.2　软 件 危 机

1.2.1　什么是软件危机

在 20 世纪 60 年代，许多开发的软件项目都未能按照预期计划实施或完成，甚至以失败告终。一些软件项目的开发时间大大超出了预期，导致了资金的浪费。一些软件甚至导致了人员伤亡，软件复杂度日益加剧使得软件人员发现软件开发的难度越来越大。

以 IBM/360 系统为例，该系统总共由 4 000 多个软件模块组成，约 100 万条指令，开发总投资金额达到 5.5 亿美元(硬件 2 亿美元，软件 3.5 亿美元)，可见这是当时一个很大的投资项目，差不多相当于美国研究原子弹的曼哈顿计划资金投入的 1/4。当时年仅 29 岁的布鲁克斯(首届计算机先驱奖、1999 年图灵奖获得者)主持了这个项目，率领 2 000 名程序员夜以继

日地工作，单单操作系统的开发就用了 5 000 个人年(1 人年为一个人工作一年的工作量)。IBM/360 系统一经推出就以系列化、通用化、标准化的特点受到追捧，被认为是划时代的杰作，也使 IBM 在两年内，资本积累就增加到 45 亿美元。但据统计，其操作系统每次发行的新版本都是从前一版本中找出上千个程序错误而修正的结果。如今经历了数十年，这个极度复杂的软件项目甚至产生了一套不包括在原始设计方案之中的工作系统。后来，布鲁克斯在他的《人月神话》一书中对这个项目的开发研制过程进行分析及总结，承认由于软件管理方面的原因，项目在某些方面来说是失败的，甚至犯了一个价值数百万美元的错误。

某些软件的错误可能导致不可弥补的巨大的损失，如 2002 年 12 月欧洲阿里亚娜火箭的爆炸。伴随着计算机软件在医院等与生命息息相关的行业的广泛应用，软件的错误还可能导致人员的伤亡。在工业生产中，某些嵌入式系统的故障会导致机器无法正常运转，降低了生产效率。

在 20 世纪 60 年代末到 20 世纪 70 年代初这段时间，软件危机一词在计算机界得以流传。事实上，软件危机几乎是从计算机诞生的那一天起就存在了，只是到了 1968 年，北大西洋公约组织的计算机科学家在联邦德国召开的国际学术会议上才第一次提出了“软件危机”(software crisis)这个名词。

所谓软件危机，是指在计算机软件的开发和维护过程中所遇到的一系列严重问题。这些严重问题绝不仅仅是某些失败的软件才具有的，几乎所有软件都不同程度地存在这类问题。具体地说，软件危机主要有以下表现：

(1)对软件开发成本和进度的估计常常不准确。软件开发工作的开始，主观盲目地制订计划，但在执行过程中发现计划与预期有较大差距，造成经费的增加。另外，对软件开发的工作量估计不足，开始制订的进度计划根本无法遵循，造成开发工作的完成期限一拖再拖，严重降低了开发组织的信誉。有时候，一些项目组为了加快软件开发的进度以按期交付软件而增加开发人员，结果造成适得其反的效果。有时为赶进度和节约成本所采取的权宜之计往往却损害了产品的质量。

(2)用户对“已完成”系统不满意的现象经常发生。对软件的需求来说，往往在开发的初期阶段不能明确提出，或是不能对其表达确切，在开发工作开始后，软件人员和用户若不能及时交换意见，会使得一些问题无法得到及时解决，造成了软件开发的后期出现大量问题，这种“闭门造车”的状态必然导致最终产品不符合用户实际需求。

(3)软件产品的质量往往不能得到保证。软件作为一种逻辑产品，其质量问题很难用统一的标准去度量，因而造成对质量的控制比较困难。软件产品不可能是完美的，总会存在一些错误，但是盲目检测是很难发现错误的，而最终隐藏下来的错误往往是造成重大软件故障的隐患。

(4)软件的可维护程度非常低。程序中的错误很难改正，实际上不可能使这些程序适应新的硬件环境，也不能根据用户的需求在原有程序中增加新的功能。

(5)软件通常没有适当的文档资料。文档资料是软件必不可少的重要组成部分。缺乏必要的文档资料或者文档资料不合格，将给软件开发和维护带来许多严重的困难和问题。

(6)软件成本在计算机系统总成本中所占比例逐年上升。软件开发的生产率每年只以 4%~7%的速度增长，远远落后于硬件的发展速度；而且软件开发需要大量的人力，软件成本随着通货膨胀以及软件规模和数量的不断扩大而逐年上升。

1.2.2　产生的原因及解决途径

根据软件本身固有的逻辑结构和以上软件危机的各种表现，可以发现软件危机产生的原因，常常表现为以下几点：

(1)用户对软件需求的描述常常是不精确的，可能存在遗漏、二义性或错误。甚至在软件开发过程中，用户还会提出修改软件功能、界面等方面的要求。

(2)软件开发人员对用户需求的理解与用户的本来愿望之间有差异，这将导致开发出来的软件产品不符合用户的要求。

(3)大型软件项目需要组织一定的人力共同完成。但在实际研发人员的团队构成上，多数管理人员缺乏开发大型软件系统的经验，而许多软件开发人员又缺乏管理方面的经验。项目团队中各类人员的信息交流不及时、不准确。

(4)软件项目开发人员往往不能有效地、独立自主地处理好大型软件的全部关系以及其中的各个分支，容易产生一些疏漏和错误。

(5)缺乏有力的方法和工具支撑，过分地依靠程序设计人员在软件开发过程中的技巧和创造性，加剧了软件产品的个性化。

(6)软件产品是人类智力活动的一种表现，但软件产品的特殊性和人类智力的局限性，导致人们无力处理“复杂问题”。

为了解决软件危机，许多计算机专家和软件学家尝试着将其他工程领域中有实效的工程学知识应用到软件的研发过程中来。

在长期的软件研发过程中，人们逐渐认识到，要解决软件危机，既要有一些技术措施(方法和工具)，也要有必要的组织管理措施。一方面，一些先进的软件开发方法和工具，不仅可以提高软件开发及维护的效率，也能在一定程度上保证软件的质量。另一方面，由于软件开发活动并不是简单的个体行为，要保证软件开发活动顺利、有效、高质量地完成，必须严密组织、管理和协调各类人员，这是必不可少的重要工作。尤其是大型的软件开发，软件的复杂性、人员的流动性都会对软件开发的组织和管理带来困难。这种情况下，经验丰富的组织管理人员，行之有效的原理、概念、技术和方法的应用，都将对软件的有效开发起到极其重要的作用。

经过不断实践和总结，最后得出一个结论：按工程化的原则和方法组织软件开发工作是有效的，是摆脱软件危机的一个主要出路。

1.3　软 件 工 程

1.3.1　软件工程定义

1968 年 10 月，NATO 的科技委员会召集了近 50 名一流的程序工程人员、计算机科学家和工业界巨头，讨论和制订如何摆脱“软件危机”的对策，Fritz Bauer 首次提出了软件工程的概念，他认为：软件工程是为了经济地获得能够在实际机器上高效运行的可靠软件而建立和使用的一系列好的工程化原则。

之后，人们对软件开发是否符合工程化思想这一核心问题进行了长达数年的探索，针对软件工程这一学科及其自身特点等问题开展了一系列的讨论和研究，从而形成了软件工程的

各种各样的定义。

P. Wegtner 和 B.Boehm 认为，软件工程是运用现代科学技术知识来设计并构造计算机程序及其开发、运行和维护这些程序所必需的相关文件资料。这里的“设计”是一个广义的概念，它包括软件的需求分析和对软件进行修改时所进行的再设计活动。

F.L.Baner 认为，为了经济地获得软件，使得这个软件是可靠的，并且能在实际的计算机上工作，需要确立健全的工程原理(方法)和过程。

1993 年，IEEE 计算机学会定义软件工程是将系统化、规范化、可度量的方法应用于软件的开发、运行和维护过程，即将工程化应用于软件中的方法的研究。

随后，又有一些人提出了许多更为完善的定义，但主要思想都是强调在软件开发过程中应该以工程化思想为指引。总之，为了解决软件危机，既要有技术措施(包括方法和工具)，又要有必要的组织管理措施。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计算机软件的一门新兴学科。

1.3.2　软件工程的研究内容

软件工程主要研究软件生产的客观规律性，建立与系统化软件生产有关的概念、原则、方法、技术和工具，指导并支持软件系统的生产活动。软件工程属于一种层次化的技术模式，如图 1-2 所示，过程、方法和工具是软件工程的三个要素。位于底层的质量焦点内容表明软件工程要以质量为关注的焦点、重心，全面的质量管理和质量需求是促进软件过程不断改进的源动力，也就是这种动力导致更加成熟有效的软件工程方法可以不断涌现出来，为软件工程奠定强有力的根基。

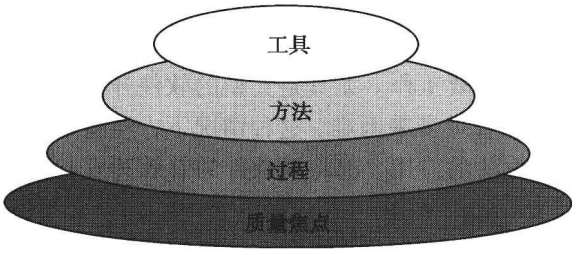

图 1-2　软件工程的层次

软件工程的基础层是过程层。软件工程过程是为获得软件产品，在软件工具支持下由软件开发人员完成的一系列软件工程的活动。过程层将方法和工具结合起来，定义了一组关键过程区域的框架，定义了方法使用的顺序、要求交付的文档资料、为保证质量和协调变化所需要的管理以及软件开发各个阶段完成的里程碑。最终目的是保证软件工程技术被有效地应用，使得软件能够被及时、高质量和合理地开发出来。

方法层为软件开发的各个阶段提供所需的各种方法。方法这一概念本质上包含了许多内容，如项目计划与估算方法，需求分析和设计方法，编程、测试方法及维护的方法等。软件工程方法依赖于一组基本原则，这些原则控制了每一个技术区域，包括建模活动和其他方面的各种描述技术。

工具层为软件工程方法提供了一种自动或半自动的软件支撑环境。目前，已经出现了多种软件工具，这些软件工具集成起来，建立起被称为计算机辅助软件工程(CASE)的软件开发支撑系统。CASE 将各种软件工具、开发机器和一个存放开发过程信息的工程数据库组合起来形成一个软件工程环境。使用软件工程工具可以有效地改善软件开发过程，提高软件开发