普通高等应用型院校“十二五”规划教材

软件工程专业

软件工程

主 编 曾强聪 赵 歆 副主编 阳王东 刘 震 周新民 刘 娜

中国水利水电出版社
www.waterpub.com.cn

普通高等应用型院校“十二五”规划教材——软件工程专业

软件工程

主　编　曾强聪　赵　歆

副主编　阳王东　刘　震　周新民　刘　娜

中国水利水电出版社
www.waterpub.com.cn

内 容 提 要

软件工程是软件研发与维护的工程方法学。本书较好地体现了软件工程的实用性，有较完整的软件工程知识体系，有对工程概念、规则的生动说明，有与工程实践相适应的基于软件生存周期的内容编排，有基于案例的工程方法应用。

全书共三个部分 16 章内容。第一部分工程基础包含两章内容，是对软件工程概念方法、软件工程项目管理的常识性介绍；第二部分工程过程包含 7 章内容，涉及软件工程过程模式，并以软件生存周期为线索，对软件研发全过程进行了说明；第三部分工程方法包含 7 章内容，有基于案例的工程方法说明，涉及主流的结构化工程方法、面向对象工程方法、数据库工程方法，并介绍了敏捷工程等一些非主流方法，第 16 章是一个较完整的面向对象工程案例，基于 UML 建模，并已通过 C++和 Java 进行工程创建。

本书语言精简、通俗易懂、便于自学，而且教学资源完备，书中案例均已基于主流软件工具（Rose、Visio、PowerDesigner）建立模型，可作为教学或自学资源供读者参考（可与出版社或作者联系获取）。

本书可作为高校相关专业本科生、研究生教材，也可作为软件开发人员的技术参考书，还可供广大读者自学软件工程方法。

图书在版编目（CIP）数据

软件工程 / 曾强聪，赵歆主编. -- 北京 : 中国水利水电出版社，2014.10

普通高等应用型院校“十二五”规划教材. 软件工程专业

ISBN 978-7-5170-2574-0

Ⅰ. ①软… Ⅱ. ①曾… ②赵… Ⅲ. ①软件工程－高等学校－教材 Ⅳ. ①TP311.5

中国版本图书馆CIP数据核字(2014)第228599号

策划编辑：周益丹　责任编辑：张玉玲　加工编辑：李　娜　封面设计：李　佳

书　名	普通高等应用型院校“十二五”规划教材——软件工程专业 **软件工程**
作　者	主　编　曾强聪　赵　歆 副主编　阳王东　刘　震　周新民　刘　娜
出版发行	中国水利水电出版社 （北京市海淀区玉渊潭南路 1 号 D 座　100038） 网址：www.waterpub.com.cn E-mail：mchannel@263.net（万水） sales@waterpub.com.cn 电话：（010）68367658（发行部）、82562819（万水）
经　售	北京科水图书销售中心（零售） 电话：（010）88383994、63202643、68545874 全国各地新华书店和相关出版物销售网点
排　版	北京万水电子信息有限公司
印　刷	三河市鑫金马印装有限公司
规　格	184mm×260mm　16 开本　18.25 印张　445 千字
版　次	2014 年 11 月第 1 版　2014 年 11 月第 1 次印刷
印　数	0001—3000 册
定　价	36.80 元

凡购买我社图书，如有缺页、倒页、脱页的，本社发行部负责调换

前　言

软件工程是关于软件产品研发与维护的工程方法学，是软件开发者——软件项目负责人、软件分析师、软件设计师、软件程序员、软件测试员研发与维护软件时的作业指南。

20 世纪 60 年代，软件产业发展遭遇到了危机，出现了软件研发成本进度难以控制，软件产品不能满足用户需求，软件程序容易出错、难于维护等诸多危机现象。这样的软件危机引发了软件研发者的心理恐慌，软件工程即诞生于这样的背景之下，其谋求对软件研发维护能有方法上的指导，能有规则上的约束，以使得面对有待研发的复杂软件系统，软件开发者能更具预见力与自信心。

软件工程影响着软件产业的发展，软件工程人才已越来越受到软件业界的重视。可以看到的是，当前的软件工程师已代替了早期的软件程序员成为了软件研发中的技术骨干。然而，软件产业的发展不只是要求软件工程师需要具备软件工程知识与技能，还要求软件研发的所有参与者——技术人员、管理人员、产品客户都应该具备一定的软件工程知识与技能，由此软件工程规则方法才能更顺利地得以应用。

软件工程在随软件产业的发展而进步。目前软件工程已是计算机科学领域中的一个重要分支，其已有了结构化、面向对象等比较成熟的工程方法学体系，并已有了对技术、管理、经济的比较全面的工程方法支持。然而，直至今天，软件工程还仍是处于成长过程中，仍然非常年轻，需要工程探索，需要逐步完善。

软件工程的真正价值在于工程应用与产生工程实效。本书编写即充分考虑到了软件工程的实用性，有对软件工程概念、规则的生动说明，有与工程实践相适应的基于软件生存周期的内容编排，并有基于工程案例的工程方法应用指导。

本书内容由三个部分共 16 章组成。

第一部分是工程基础（第 1、2 章），涉及软件工程概况与软件项目管理两个方面的内容。我们认为要学好软件工程，首要的问题是学习者能对软件工程有一个好的学习态度，这个部分即用于培养与树立学习者的工程意识，并因此激发起学习者的工程学习热情。

第二部分是工程过程（第 3～9 章），涉及一些主流的过程模式，并以软件生存周期为线索，对软件研发全过程进行了细节说明，包括：软件需求分析、软件概要设计、程序算法设计、软件测试、软件维护与进化等内容。我们认为，以软件生存周期为线索的内容编排将有利于基于软件项目的工程实践与学以致用。

第三部分是工程方法（第 10～16 章），既包括主流的结构化工程方法、面向对象工程方法、数据库工程方法，也涉及一些非主流的工程方法，如敏捷工程方法、净室工程方法等。诸多方法都通过案例进行了应用说明。这个部分的核心内容是面向对象工程方法，有比较全面的基于 UML 的建模说明。第 16 章是一个较完整的面向对象工程案例，不仅有基于 UML 的分析设计建模，并通过 C++、Java 进行工程创建。

本书的重点是软件分析与设计，并主要体现在分析设计建模上。对于分析设计建模，本书有以下方面的教学考虑，这有利于学习者更好地把握建模方法：

（1）学习建模不能仅限于识别图形元素或看懂图表，而应该学会如何基于一定的流程，并通过一定的建模方法，对软件问题进行模型演变，由此使软件问题得解。因此，本书中无论结构化建模还是面向对象建模，其都有较好的基于案例的由分析到设计的全过程推演，以反映软件系统构建的严密性。

（2）结构化分析中的数据流细化建模一直是一个学习难点，其实它也是一个实际建模中的应用难点，即不知道如何进行功能细化。对此，本书基于功能树的功能逐级分解，可使这个难点得到较好的处理，其具有教学意义，同时也有一定的工程参考价值。

（3）统一开发过程中的面向对象建模是基于用例驱动的，由此可使对系统的业务分析能延伸到系统的结构设计、过程设计、界面设计、安装部署。然而，以往的大多数教科书，仅局限于纲领性说明，而无实际建模演示，因此也就看不到用例驱动的实际价值。对此，本书有较好的基于案例的用例驱动建模过程说明，如分析中基于用例的活动建模，设计中基于用例的类结构建模、对象协作建模，诸多案例无不体现出用例驱动的价值。

一些初学者认为，软件工程是教条式理论。然而，它却不应该成为教条，只是有可能出现教条式教学。实际上，软件工程有很强的工程实践性，其一系列的方法规则就建立在工程实践基础上。无疑，这个实践性必然会对软件工程的教学提出要求，这就是有用、能用、实用。显然，经过软件开发者长期的实践与努力探索而获得的工程方法肯定是有用的，值得认真学习。然而，有用还只能体现出软件工程的知识特性，若要使这些知识转化为学习者的能力，则必须能用。通常的看法是，案例教学有利于培养学习者能力。本书提供了较丰富的案例，其目的就在于培养学习者的能力，以使书中的知识能够被用起来。软件工程教学还要求实用，可用来解决实际问题。对此，可采取项目方式进行工程实践，这有利于知识、能力与实际应用的结合，由此可使软件工程学习具备实用性。

还需要注意的是，基于项目的工程实践的成效依赖于教师的精心组织。我们的建议是，可3～5人一组进行项目实践，并以组为单位进行整体评价，但各成员应有特定任务，以方便对各成员做个体评价。这种教学模式的好处是，既可考察学习者的团队协作，又可体现学习者的个人学习成就。还由于是团队合作学习，有利于调动学习者的工程实践热情，能产生更好的工程实践效果。

本书主要由曾强聪、赵歆编写，阳王东、刘震、周新民、刘娜承担了部分内容的编写工作。无疑，本书的创作是一件非常耗费心力的事情。首先是结构，它应该是严谨的。接着是语言，其应该既严密又畅快，以便于阅读，并能尽量吸引学习者去阅读。接下来就是取舍内容、定义概念、描述方法、设计案例等。

实际上，本书中的每一个细节都经过反复推敲，因为毕竟是教科书，很担心因工作疏忽而给学习者带来误导。然而，由于作者学识水平与时间都是有限的，因此书中难免有一些缺点和不足，对此特请广大读者批评指正，以使本书再版时能够不断完善。值得庆幸的是，软件工程仍在不断发展，因此本书的再版与完善也是一件很自然的事情。

编　者

2014年9月

目　录

第 1 章　软件工程综述

软件工程是工程化软件开发与维护的方法论。软件的开发者、维护者或软件项目管理者都将是软件工程的实践者，并都需要掌握与应用软件工程方法。

- 软件工程及其发展
- 软件特点与分类
- 软件危机现象与原因
- 工程技术与工程管理
- 主流工程方法学

1.1　什么是软件工程

软件工程是关于软件开发、使用与维护的工程方法学，是一门涉及工程技术、工程管理与工程经济等诸多内容的综合性工程学科。软件工程建立在与软件有关的工程概念、原理与方法基础上，它是对现实软件问题的工程方法探索，具有鲜明的工程实用性。

软件工程是软件产业发展的必由之路。通常看来，软件产业的发展涉及软件技术的进步、软件应用域的拓宽、软件生产方式的改进，其犹如三匹骏马，拉动着软件在产业化大道上奔驰。大体上看，软件的产业化发展经历了程序设计、程序系统和软件工程三个时代。

程序设计时代（20 世纪 50 年代）是软件发展的早期阶段，这时的软件仅体现为程序，一般使用密切依赖于某特定计算机硬件的机器语言或汇编语言，直接面对机器编制。因此，这时的程序很难由一台设备移植到另一台设备。这时的程序则主要用于科研机构的科学工程计算，设计中通常没有设计文档的支持，而且程序的任务单一、规模小、结构简单，其主要设计问题是程序算法改进。这时的程序还没有成为产品，程序编制者大多就是程序使用者，基本上是个人设计、个人使用、个人操作、自给自足的个性化的软件生产方式。实际上，由于缺乏统一的程序标准，设计者完全可按照自己对问题的理解去构造程序，目标则是实现功能与追求高性能，硬件是唯一需要重视的条件限制，此外则似乎再无其他原则性约束。

程序系统时代（20 世纪 60 年代）是软件产业发展的关键阶段，这时的计算机无论硬件还是软件都出现了显著的进步。体现计算机硬件指标的半导体材料、集成电路迅速发展，并获得了很有成效的应用，其不断改善计算机的计算速度、可靠性和存储容量。计算机软件也在迅速发展，高级程序语言获得了有效应用，其提高了编程效率，并方便了大规模复杂程序系统的创建。操作系统也在这个时期出现了，其有效地改善了软件的工作环境，并使软件具有了很好的可移植性。程序系统时代的技术进步有力地推动了计算机应用的发展。一些大型商业机构开始使用计算机进行商业数据处理。应该说，这个时期的软件需求在飞速增长，软

件规模也在不断扩大。“软件作坊”在这个时期应运而生，它们专业从事软件生产，以满足迅速膨胀的软件需求。

程序系统时代出现了软件危机。软件作坊带来了不同于个人开发的工业化软件产品，但仅仅只是工业化软件生产的起步，成员合作需要依赖的行为准则、技术标准并没有建立起来，以致产品随意性大、质量难以保证。

应该说，软件产业发展过程中爆发的软件危机促进了软件工程的快速成长。1968 年在前联邦德国召开的计算机国际会议上，来自原北大西洋公约组织的计算机科学家针对软件危机问题进行了专题讨论。软件工程这个学术名词也在这次学术会议上被正式提出，其用来代表基于工程模式的软件开发，以谋求对软件危机的有效克服。

软件工程使身处软件危机黑暗中的软件开发者看见了曙光，并使软件产业发展步入到了软件工程时代（20 世纪 70 年代起）。无须质疑的是，软件的产业化发展也确实需要有软件工程方法的支持。实际上，自从软件成为了工业化产品，它所面临的就已不只是软件技术问题了，还必须考虑如管理、经济、应用等其他因素。软件工程即是基于工程角度对有关软件的诸多因素的综合考虑。

软件工程的最核心问题是，如何更有效率并更高质量地开发软件，其作用范围则贯穿于自软件的最初定义，到软件的开发实现，直到软件的使用与维护的整个软件生命过程。因此，无论是软件开发者，或是软件维护修复者，还是软件使用者，都有可能成为软件工程实践者。有许多人在专门从事软件工程研究，他们是专业的软件工程研究者，他们一直在探讨如何给软件工程下一个恰如其分的定义，以使该学科研究有一个较为明确的发展目标。

20 世纪 60 年代末，软件工程早期研究者 FritZ Baner 给软件工程的定义是：“为了经济地获得能在实际机器上可靠运行的软件，而需要合理建立，并有效应用的工程原则”。显然，这是一个在今天看来很不全面的定义，其工程特征仅体现为工程原则。但该定义把软件工程学科研究中所将面临的实质问题提出来了，即软件的成本、软件的质量、软件生产应该采用的工程方法。因此，该定义成为了以后软件工程方法学成长发展的基线。

1993 年，国际权威机构 IEEE 给出了关于软件工程的更加全面的定义，即“软件工程是：①将系统的、受规范约束的、可量化的方法应用于软件的开发、运行与维护，即将工程方法应用于软件；②对①中工程方法的研究”。应该说，这个定义对软件工程的工程学科特性有了更完整清晰的表述，如对软件工程的作用范围、软件工程基于工程应用的研究途径等都有了比较具体的说明。

1.2 软件有什么特点

计算机系统由硬件、软件两部分组成。硬件是物理部件，如处理器、存储器、主板总线，具有一定的物理形态，能够独立存在。软件则是物理硬件以外的逻辑部件，如程序、数据、文档，抽象无形，不能独立存在。

软件工程需要研究如何更有成效地研发软件、维护软件，要达到这个目标，则必然需要对软件有很好的认识。

20 世纪 50 年代，计算机主要用于解决复杂科学工程计算，核心问题是计算过程，因此，这时的软件就是程序指令。20 世纪 60 年代，为了适应计算机的大批量数据处理应用，数据从

早期程序中分离出来，因此，软件又多了数据这个逻辑要素。20世纪70年代，为了适应软件系统的工程模式研发，用于描述软件结构、说明软件使用的软件文档又成为了需要独立对待的新的软件要素。

因此，目前看来，软件是由程序、数据、文档三个部分组成的。

1.2.1 软件特点

可以将软件与硬件进行比较，从中可以发现软件具有以下方面的特点：

（1）软件有对硬件不可缺失的依赖。

任何时候，软件都离不开硬件的支持。显而易见的是，必须要有磁盘、光盘、磁带这些有形的物理介质，抽象的逻辑软件才能存储；必须要有CPU、内存的支持，抽象的逻辑软件才能进行物理运行；必须要有键盘、鼠标、显示器、打印设备的支持，软件中的抽象数据才能有物理的输入、输出。

（2）软件有不同于硬件的生产流程。

软件不是制造出来的，而是开发出来的。硬件生产的最大成本是复杂的制造工艺，而软件生产的最大成本则是分析与设计。例如，产品质量监控，硬件主要体现为对产品制造工艺的流程监控，而软件则主要体现为对产品的分析设计的流程监控。

软件生产还有不同于硬件生产的资源需求。通常，硬件生产对原材料、能源、生产设备等有较高的依赖度，并难免对周围环境带来污染。软件生产则主要依赖于人的智慧，而对原材料、能源、生产设备等有很低的依赖度，不会给周围环境带来污染。

（3）软件有不同于硬件的生命过程。

硬件在使用过程中往往会经历：磨合期、正常使用期、老化期三个阶段。

- 磨合期：最初使用阶段。由于硬件中的一些难以预料的缺陷，这个阶段往往会有一个相对较高的故障频率。
- 正常使用期：硬件在磨合之后的一个较长的稳定工作过程，故障频率相对较低。
- 老化期：硬件寿命接近终点时的阶段。由于长久使用，硬件将出现过度磨损，因此故障频率会越来越高，并逐渐失效而不能工作。

图1-1所示为硬件的生命过程曲线。显然，高质量硬件产品应有较低的磨合期故障频率与较长久的正常使用期。

软件也有类似于硬件的磨合期与正常使用期。

软件磨合期的高故障频率通常由软件中隐藏的错误或缺陷所致，并随着这些错误或缺陷的排除而进入正常使用期。然而，软件是用不坏的，因此不会出现磨损现象。实际上，只要软件的工作环境没有变化，并且软件本身不做改动，则在其进入正常使用期后，可以一直使用下去。

软件无被动磨损，但软件可主动进化。通常，软件在使用一定周期后，需要通过改版进化而获得新的生命力，以使软件能适应新的工作环境，能提供新的更完善的应用服务。必须注意的是，软件可因改版进化而引入新的错误或新的缺陷，并使软件需要重新经历磨合期。

图1-2所示为软件的生命过程曲线。显然，如同硬件一样，高质量软件也应该有较低的磨合期故障频率与较长久的正常使用期。

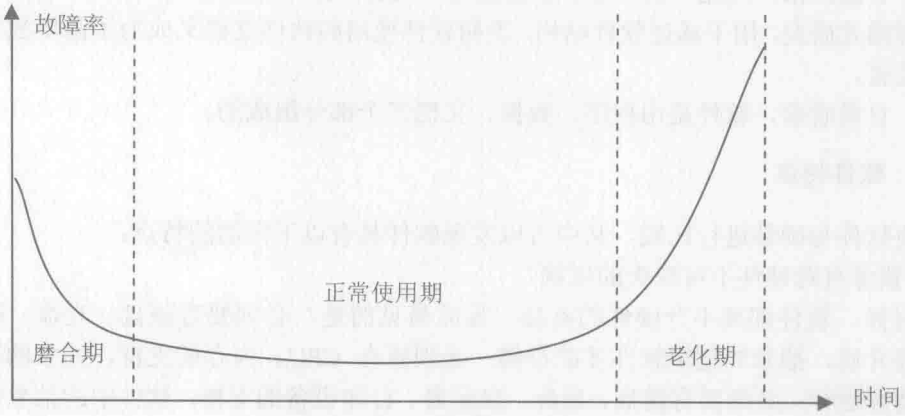

图 1-1　硬件产品生命过程曲线

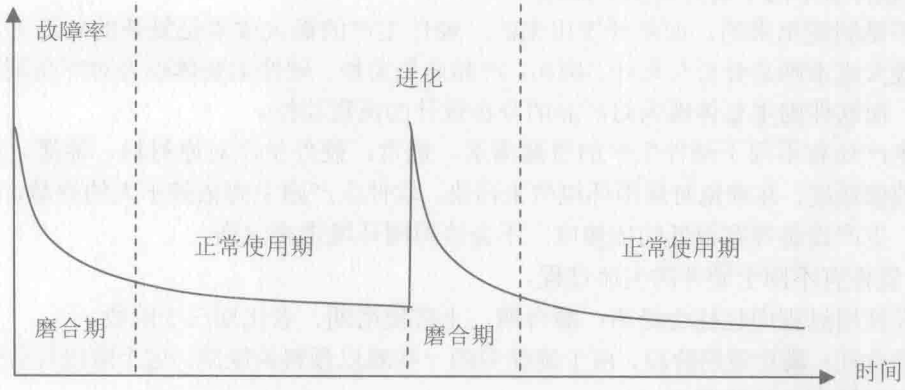

图 1-2　软件产品生命过程曲线

1.2.2　软件分类

软件分类有利于更好地认识软件。然而，同一个软件可以有多个不同的认识视角，如技术人员视角、管理人员视角、用户视角，因此也就可以按不同方式进行分类。

1. *按功能层次分类*

软件可依据功能地位而分为系统软件、支撑软件与应用软件，其关系如图 1-3 所示。

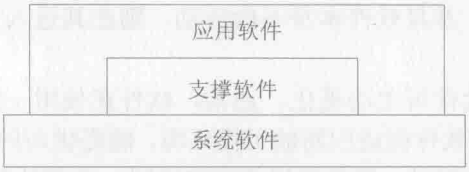

图 1-3　系统软件、支撑软件与应用软件之间的关系

（1）系统软件：为计算机底层软件，如操作系统、设备驱动程序、数据库引擎等。系统软件的特点是与计算机硬件紧密相关，并通过与硬件的频繁交互而对其他软件的运行提供底层

服务支持。例如操作系统，其建立在硬件环境基础上，并通过文件服务、进程服务、存储服务而给其他软件提供更加适宜的运行环境。

（2）支撑软件：介于系统软件与应用软件之间。工具软件是最常见的支撑软件，如程序编译器、程序编辑器、错误检测程序、程序资源库等。中间件软件也是支撑软件，根据用途不同可分为：数据中间件、对象中间件、通信中间件。可通过中间件的嵌入、组合或二次开发构造应用软件。

（3）应用软件：为最终用户提供应用服务的软件，通常由工具软件开发，并依靠系统软件的支持运行，如财务处理系统、生产控制系统、自动办公系统。早期的应用软件大多为离散结构，功能相互独立。但目前的应用软件则一般为多功能集成系统，如企业资源管理系统，即集成有企业中的各种资源的管理与协调，涉及人力、设备、资金、客户、原材料、生产线、半成品、产品等诸多方面的管理与控制。

2. 按工作方式分类

（1）实时软件：能够对外部事件或外部环境变化做出及时响应的软件，如飞机自动导航程序、导弹目标跟踪程序、自动生产线监控程序。实时软件一般由数据收集器、数据分析器、响应控制器等诸多部件组成，如图 1-4 所示。通常，数据收集器负责从外部环境获取格式化信息；数据分析器负责对获取的格式化信息进行分析、判断；响应控制器则负责按分析器对外部事件的判断进行有效的行为应对。例如导弹目标跟踪程序，当导弹跟踪目标发生移动时，导弹可通过热感应方式获取移动信息，并通过数据收集器将移动信息格式化，然后通过分析器的分析、判断确定目标新的位置，再由响应控制器调整运行轨迹，以达到有效跟踪的目的。

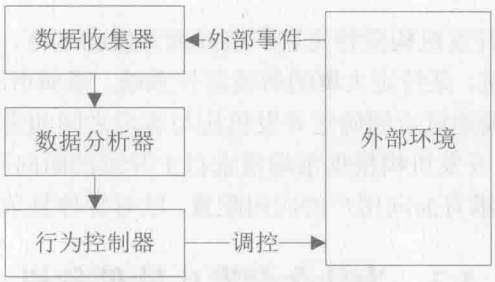

图 1-4 实时软件部件组成

（2）分时软件：能够把一个 CPU 工作时间段切片分配给多项任务的软件。如 Windows 操作系统，它即是一个具有分时功能的多任务操作系统，能使多个程序同步运行。与分时软件相关的软件技术是多线程机制，即在一个程序进程内执行多个线程任务。

（3）交互式软件：用于实现人机对话的软件，大多具有图形界面，如窗体程序、Web 页面。交互式软件的工作空间一般在客户端。窗体程序的特点是无论安装还是运行都在客户端。Web 页面则是以 HTML、XML 等文档格式安装于 Web 服务器，但运行空间仍在客户端，图 1-5 所示即为 Web 页面的工作特征。

（4）嵌入式软件：专门为特定智能设备提供自动控制的软件，如洗衣机流程控制程序、微波炉流程控制程序、汽车定速导航程序。嵌入式软件一般驻留在智能设备拥有的只读存储器中，因此与智能设备融为一体。

（5）批处理软件：能够成批处理大量数据或成批处理多项事务的软件。批处理软件通常

需要面对大量数据，并往往以某个时间点（如年终、月尾）为任务触发事件。可通过批处理而提高系统工作性能，如将可能影响系统日常性能的大量数据的处理安排在机器相对空闲时进行成批处理。可通过批处理而使事务操作延后，一些涉及多方协作而不便于同步操作的事务即可通过批处理而获得异步操作。

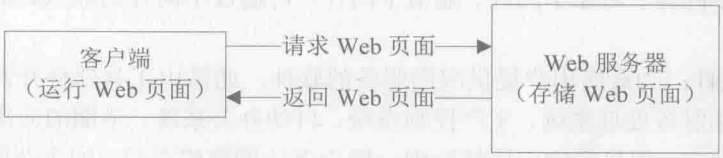

图 1-5　Web 页面的工作特征

3. 按规模分类

（1）微型软件：源程序代码行数 500 行语句以内的程序，通常一个人几天内即可完成开发。

（2）小型软件：源程序代码行数 2000 行以内的程序，通常一个人半年之内即可完成开发，大多为用户自主开发。

（3）中型软件：源程序代码行数 5 万行以内，并具有一定复杂度的软件系统，通常一年内可完成开发。

（4）大型软件：源程序代码行数 5 万行以上，具有综合功能，并涉及多用户任务协作的相当复杂的软件系统。

4. 按服务对象分类

（1）用户定制软件：开发机构受特定客户委托而开发的软件，如某特殊设备的控制系统、某专门企业的业务管理系统、某特定大厦的智能监控系统、某城市的交通监管系统。大多以招投标方式委托开发，并需要通过合同确定开发机构与客户之间的责权。

（2）通用商业软件：开发机构根据市场需求自主开发的面向广大用户的软件，如通用办公系统、通用财务系统。要求有面向用户的应用配置，以对各种复杂工作环境有良好的适应性。

1.3　为什么会发生软件危机

20 世纪 60 年代中期发生了软件危机，其严重影响了软件开发者的自信心。通常看来，是软件危机催生了软件工程，并通过软件工程克服了软件危机，重建了软件开发者的自信心。因此，软件危机一直被软件工程研究者重视。

软件危机主要表现在以下几个方面：

（1）软件开发成本、进度失控。许多软件尽管在开发之前已有了看似周密的成本估算，但实际开发费用则往往远远超出前期预算。开发进度难以控制，已定的进度计划可能是三个月提交产品，但实际则是半年过去了，而能够真正投入使用的最终产品却仍是遥遥无期。

（2）软件质量不能获得有效保证。花费大量资金、人力开发出来的软件可能不能正常使用，经常出故障，性能不稳定。不重视分析设计，只希望能尽快编写出可运行程序交用户使用，而为了压成本、赶进度，一些必须要有的软件测试也被省去了。显然，以这种方式开发出来的软件，其质量很难有保证。

（3）软件不能满足用户应用需要。尽管开发者自己觉得开发出了一个还算不错的软件，然而用户不买账，认为提交给他们的软件与开发者在合同中承诺的软件并不一致，一些必须要有的功能给遗漏了，程序工作流程与实际业务流程也可能不一致。

（4）软件可维护性差。软件难免需要进行错误修正或功能修补。然而，当开发者需要修正或改进软件时，则发现这是一件非常头痛的事情，原因是维护中需要依赖的设计文档很不完整，很多程序根本就找不到对应的设计说明。有些程序尽管有设计说明，但源程序已有多处改动，与设计说明严重不符。

为什么会发生软件危机呢？通常看来，有主客观两个方面的原因。

1. 客观原因

（1）软件的逻辑特性。软件是计算机系统中的逻辑部件，这使得软件缺陷具有了较大的隐蔽性，缺少硬件缺陷所具有的直观表象，难以发现。通常，硬件缺陷会有发热、噪音等不正常现象相伴随，它有一个逐渐变坏的过程，大多可以在没有引发严重故障之前被发现。软件缺陷则缺乏硬件缺陷的直观表象，必须等到错误发生前提条件成立时故障才会出现，如软件中的编码错误就必须等到出错的这行代码被执行时才有可能被发现。

（2）软件的复杂性。软件成为了工业产品之后，其规模越来越大，复杂程度越来越高。实际上，许多庞大的软件系统，其复杂程度已经远远超出了常人的理解力。人们寄希望于软件工具支持高度复杂软件系统的开发。然而直到今天，用于软件开发的工具仍很不完善，许多复杂软件问题的解决仍还不得不依靠人基于经验的决策判断。

（3）软件的产业发展状况。20 世纪 60 年代，软件产业才刚刚起步，缺乏工程规范、技术标准。就算是今天，即使有了一定的工程规范，但实施起来仍还是难度很大，并没有发挥其应有的作用。以 CMM 认证体系为例，这是一个有关软件工程过程的达标认证，用来反映软件企业的产业化完善程度，有 5 个认证级别，要求逐级认证。然而，在世界范围内，能够通过 3 级 CMM 认证的软件企业也不是很多。

2. 主观原因

（1）漠视用户需求。软件技术与用户需求是软件开发的两个支撑点，但软件开发者往往是一头热，他们的热情几乎都落到了软件技术上，因此用户需求成了冰点，被忽视甚至漠不关心。然而，软件毕竟是为用户开发，漠视用户需求必然使软件面临应用上的困难。漠视用户需求还给开发者带来了关门搞开发的陋习，很多通用软件就是按照这种方式开发出来的，开发之前没有进行广泛有效的市场调研，以致开发出来的产品不能获得市场的热情响应。这个问题也反映到了委托方式的定制软件上，形式上虽有一个需求分析期，但实质上也就是要求用户填写一封需求调研表格，紧接着就是软件开发，必须要有的现场调研、资料收集、需求论证被忽视了，以致开发出来的软件与用户的实际业务有冲突。

（2）重结果轻过程。软件开发的过程透明度一直很低，这既与软件的逻辑本性有关，也与早期开发中形成的个人创作工作模式有关。由于软件过程的低透明度，以致软件项目负责人、评审人员、用户都只能凭借最后结果对软件作出评价。问题的关键是，由于软件过程不透明，因此我们不关心过程，于是过程变得越来越不透明。但实质上，软件开发过程的透明度是可以提高的，只是我们为此需要花费更多的时间与精力，需要做更多的事情，如记录工作日志、加强配置管理。提高软件过程的透明度也是很重要的，过程决定结果，透明的过程将带来透明的结果，其有利于软件变更、维护与质量追踪。