全国教育科学“十一五”规划课题研究成果

实用软件工程教程

Practical Software Engineering

刘金安　主　编

高等教育出版社
HIGHER EDUCATION PRESS

全国教育科学“十一五”规划课题研究成果

实用软件工程教程

Shiyong Ruanjian Gongcheng Jiaocheng

刘金安　主编

高等教育出版社·北京
HIGHER EDUCATION PRESS　BEIJING

内容提要

本书是全国教育科学“十一五”规划课题研究成果，针对应用型本科计算机及相关专业而编写，从实用的角度出发，结合大量软件项目的实例分析，以软件的生存周期作为主线索，阐述软件工程方法、应用技术和实用工具。本书主要包括软件工程概述、软件立项与合同、需求分析、系统设计、软件实现、软件测试、软件发布与实施、软件维护、软件配置管理、软件项目管理、软件工程常用工具及开发实例，每章均配有习题，其中很多章节还安排了典型例题解析，最后一章是开发实例，可供学生练习使用。

本书注重基础性、系统性、实用性和新颖性，内容深入浅出，可以作为计算机类或信息类相关专业的教材，也可供从事计算机工程与应用工作的科技工作者参考。

图书在版编目（CIP）数据

实用软件工程教程 / 刘金安主编. -- 北京 : 高等教育出版社, 2012.2
ISBN 978-7-04-033847-8

Ⅰ.①实… Ⅱ.①刘… 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(2012)第008299号

策划编辑 刘 茜　　责任编辑 张海波　　封面设计 张雨微　　版式设计 余 杨
责任校对 窦丽娜　　责任印制 田 甜

出版发行 高等教育出版社
社　　址 北京市西城区德外大街4号
邮政编码 100120
印　　刷 北京铭传印刷有限公司
开　　本 787mm×1092 mm 1/16
印　　张 20
字　　数 490千字
购书热线 010－58581118
咨询电话 400－810－0598
网　　址 http://www.hep.edu.cn
http://www.hep.com.cn
网上订购 http://www.landraco.com
http://www.landraco.com.cn
版　　次 2012年 2月第1版
印　　次 2012年 2月第1次印刷
定　　价 29.30元

物 料 号 33847-00

前　言

进入 21 世纪的信息社会，面对大量复杂的计算机应用需求，如何多（数量）、快（速度）、好（质量）、省（费用）地进行软件开发和减少软件危机的困扰，已成为软件开发人员面临的主要任务。因此，软件工程的方法和技术越来越受到人们的关注。现在，它已经成为计算机科学技术的一个异常活跃的研究领域。

软件工程，着重研究软件过程模型、设计方法、工程开发技术和工具，是指导软件生产和管理的一门新兴的、综合性的应用学科。软件工程是指采用工程的概念、原理、技术和方法来开发和维护软件，把经过时间检验证明是科学的管理技术和当前能够掌握得最好的软件开发技术结合起来，以便经济、有效地开发出高质量的软件。

本书作者结合长期教学经验和工程项目实践经验，从实用的角度出发，参考国内外众多最新（版本）的教材和论文精选内容，注重基础性、系统性、实用性和新颖性，结合大量软件项目的实例分析，深入浅出地阐述软件工程方法、应用技术和实用工具，并将软件开发技术、软件工程环境和软件工程管理等许多方面的知识进行融合，以软件的生存周期作为主线索，按软件工程的过程对软件工程知识进行讲解。全书共分为 12 章，第 1 章进行软件工程的概述；第 2、3、4、5、6、7 章是本书的重点，按软件生存周期过程，分别讲述软件项目立项、需求分析、概要设计、详细设计、实现、测试、发布与实施、运行与维护阶段的各种方法和技术；第 8、9 章介绍软件配置管理和有关软件项目组织、度量、计划、风险、质量等软件项目管理方面的内容；第 11 章介绍项目管理、分析建模、软件测试和版本控制等常用项目管理工具；第 12 章通过一个实例介绍软件工程方法和工具的实际应用。

本书的主要特点如下。

（1）内容清晰：以软件项目为对象，将软件开发技术和软件工程管理等方面的知识结合起来，介绍整个软件生存周期的软件工程活动。

（2）实践性强：融合实践经验和引用大量的典型案例，并对当前流行的和常用的软件工具进行讲解，通用且实践性强。

（3）新颖易读：书中介绍的各种软件工具都是最新版本的，内容新颖，图文并茂，原理、方法与实例相结合，叙述通俗易懂。

本书适用面广，适合初学软件工程的读者使用，既可以作为高等学校、独立学院和高等职业学院的计算机相关专业教材，也可以作为从事计算机软件系统研发等工作的应用型技术管理人员的参考书。建议学时设置为 48。

本书由刘金安、朱德军、高涛、杨宏霞和闵亮编写，其中朱德军编写了第 1、2、3、4 章，高涛编写了第 5、6、7、8 章，刘金安编写了第 9、10、11 章，刘金安、朱德军和闵亮编写了第 12 章，杨红霞编写了 11.2 节。全书由刘金安统稿。陆丽娜教授对本书内容进行了评审，并

提出了一些宝贵的意见。

本书在编写的过程中，参考了大量国内外的优秀教材、著作、学术论文和网上传播的一些优秀作品，并在书中部分地方引用了一些观点，有准确出处的已列入参考文献中，未能找到真实姓名或由于疏忽没有列入参考文献的著作内容，望作者及时与本书作者联系，我们真诚希望能了解和认识更多的软件行业专家和学者，共同交流、探讨和研究，不断地完善和改进本书。在本书的编写过程中，得到了西安交通大学城市学院教材建设项目基金资助，特此致谢！

由于编著者水平有限，时间比较仓促，难免有疏漏和不当之处，敬请专家、同行和广大读者们批评指正。

编 者

2011年7月

目　录

第1章　软件工程概述

本章要点

- 软件、软件危机、软件工程和软件生存周期等概念
- 软件工程的目标、原则、原理和内容
- 软件生存周期各阶段的原则和任务
- 软件开发过程模型及开发方法

1.1　软件与软件危机

1.1.1　软件的定义与特点

现代计算机的思想由来已久，英国皇家学会会员、剑桥大学数学教授巴贝奇（Charles Babbage，1792—1871）最早提出，人类可以制造出通用的计算机来代替大脑计算复杂的数学问题。当时并没有电子技术可以应用，于是巴贝奇的设想就架构在当时日趋成熟的机械技术上。巴贝奇将他设想的通用计算机命名为“分析机”，但当时的科学界都讥笑他是“愚笨的傻瓜”，公然称这是“毫无价值的幻想”。唯一能理解巴贝奇的人是被公认为世界第一位软件工程师、世界计算机先驱中的第一位女性——爱达·奥古斯塔（Augusta Ada Lovelace，1815—1851），她帮助巴贝奇研究分析机，建议用二进制数代替原来的十进制数，并开天辟地地第一次为巴贝奇的分析机编制程序，成为世界上第一位程序员，他们的名字被永远载入了计算机发展的史册。

在20世纪40年代末，软件伴随着第一台通用电子计算机ENIAC在美国的问世而诞生，从此以专门编写软件的职业人陆续出现。20世纪60年代，美国大学里开始出现专门教授软件开发的专业，“软件”一词开始出现，软件产业开始起步并造就了一批亿万富翁。

“软件”的定义是计算机系统中与硬件相互依存的另一部分，包括程序、数据及相关文档的完整集合。其中，程序是软件开发人员根据用户需求开发的、用程序设计语言描述的、适合计算机执行的指令（语句）序列。数据是使程序能正常操纵信息的数据结构（即数据的组织形式）。文档是与程序开发、维护和使用有关的图文资料。可见，软件由两部分组成：一是机器可执行的程序和数据；二是机器不可执行的，与软件开发、运行、维护、使用等有关的文档。

要全面、正确地理解计算机和软件，必须了解软件的特点。

① 软件是一种逻辑产品，它与物质产品有很大的区别。软件产品是看不见、摸不着的，因而具有无形性、抽象性，它是脑力劳动的结晶，它以程序和文档的形式出现。人们可以将其记录在纸上或存储在计算机存储器的磁盘或光盘等介质上，通过计算机的执行才能体现出它的功能和作用。

② 软件的生产与硬件不同，它没有明显的制作过程。一旦某一软件项目研制成功，以后就可以大量地复制同一内容的副本。所以要对软件的数量进行控制，必须在软件开发技术上和法律上采取有力的措施。

③ 软件在运行、使用期间不存在磨损、老化问题。软件虽然在生存周期后期不会因为磨损而老化，但为了适应硬件、环境以及需求的变化也要进行修改，而这些修改又会不可避免地引入错误，导致软件失效率升高，软件退化。

④ 软件的开发、运行对计算机系统具有依赖性，受计算机系统的限制，这导致了软件移植性差的问题。

⑤ 软件产品的生产主要是脑力劳动，还未完全摆脱手工开发方式，大部分产品是“定做”的，很少能做到利用现成的部件组装成所需的软件。

⑥ 软件复杂性高，成本昂贵。软件是人类有史以来生产的复杂度最高的工业产品之一。软件涉及人类社会的各行各业、方方面面，软件开发常常涉及其他领域的专业知识。软件开发需要投入大量、高强度的脑力劳动，成本高，风险大。

⑦ 相当多的软件开发还涉及诸多的社会因素，如很多软件的开发、运行和维护都会涉及相关机构、体制、管理方式等方面的因素，同时也涉及人们的观念和心理、软件知识产权及法律等问题。

1.1.2 软件的发展

程序可以认为是软件的前身，伴随着计算机硬件性能的大幅度提高、计算机体系结构的不断完善，软件经历几十年的发展也日趋成熟和复杂，具体来说，其发展经历了以下 4 个阶段。

1. 程序设计时代（20 世纪 50 年代初期至 20 世纪 60 年代初期）

这个阶段的生产方式是个体手工劳动，大多数软件由使用者自己开发、编写，自己应用，基本上没有标准可遵循，开发者之间也没有明确分工，使用的工具是机器语言、汇编语言，程序开发完成后，除了程序清单外，没有其他文档保存下来。开发方法是追求编程技巧，追求程序运行效率，因而使得程序难读、难懂以及难修改。硬件特征是价格贵、存储容量小、运行可靠性差。软件特征是只有程序、程序设计概念，没有科学的程序设计方法和管理方法。

2. 软件系统时代（20 世纪 60 年代中期至 20 世纪 70 年代中期）

这个阶段的生产方式是作坊式的小集团合作生产，生产工具是高级语言，开发方法仍旧靠个人技巧，但已开始提出结构化方法。硬件特征是速度、容量、工作可靠性有明显提高，价格降低，销售量呈爆炸式增长。软件特征是程序员数量猛增，大量其他行业人员进入这个行业，因为缺乏训练，所以开发人员素质差，这时人们已意识到软件开发的重要性，但开发技术没有新的突破，大量软件开发的需求已提出，但开发人员的素质和落后的开发技术不适应规模大、结构复杂的软件开发，产生了尖锐的矛盾，导致软件危机的产生。

3. 软件工程时代（20 世纪 70 年代中期至 20 世纪 80 年代中期）

这个阶段的生产方式是工程化的生产，使用数据库、开发工具、开发环境、网络、分布式技术开发软件。硬件特征是性能不断提高而价格不断下降。软件特征是开发技术有很大进步，需求变得更高、更快、更复杂，以软件产品化、工程化、系列化、标准化为特征的软件产业迅

猛发展，推动了软件工程学的进步，但是未能获得突破性进展，软件价格不断上升，软件危机加剧。

4. 现代软件工程时代（20 世纪 80 年代中期至今）

这个阶段计算机网络迅速发展，软件开发不再侧重于单台计算机的应用，计算机体系结构从中央主机控制方式转变为分布式的客户-服务器方式和浏览器/服务器方式，由复杂的操作系统控制强大的桌面机、广域网和局域网。同时专家系统和人工智能也开始得以实际应用，多媒体系统改变了用户之间的通信方式，出现了并行计算和网络计算，软件技术向产业化方向发展。一些新技术蓬勃兴起，面向对象技术在许多领域迅速取代了传统的软件开发方法。

1.1.3 软件危机

1. 软件危机的产生

如果开发的软件隐含错误，可靠性得不到保证，那么在运行过程中很可能会造成十分严重的后果，轻则影响系统的正常工作，重则导致整个系统瘫痪，乃至造成无可挽回的恶性事故。如银行的存款可能被化为乌有，甚至变成赤字；工厂的产品全部报废，导致工厂破产；军事指挥系统瘫痪，造成全军溃败。1963 年，美国用于控制火星探测器的计算机软件中的一个“,”号被误写为“.”，而致使飞往火星的探测器发生爆炸，造成高达数亿美元的损失。

20 世纪 60 年代中期以后，计算机硬件技术日益进步，存储容量、运算速度和可靠性明显提高，生产硬件的成本不断降低。计算机价格的下跌为其得到广泛应用创造了极好的条件。在这种形势下，迫切要求计算机软件也能与之相适应。因而，一些开发大型软件系统的要求被提了出来。然而，软件技术的进步一直未能满足形势发展的需要，在大型软件的开发过程中出现了复杂程度高、研制周期长、正确性难以保证的三大难题。遇到的问题得不到解决，致使问题堆积起来，形成了人们难以控制的局面，出现了所谓的“软件危机”。

2. 软件危机的定义

所谓软件危机（Software Crisis）是指计算机软件在开发和维护过程中所遇到的一系列严重问题。概括地说，主要包含两方面的问题：如何开发软件才能满足用户对软件日益增长的需求，如何维护数量不断膨胀的已有软件。实际上，几乎所有的软件都不同程度地存在这些问题。

3. 软件危机的表现

（1）对软件开发成本和进度的估计常常无法控制

实际成本比估计成本有可能高出一个数量级，实际进度比预期进度拖延几个月甚至几年的现象并不罕见。这种现象降低了开发组织的信誉。为了赶进度和节约成本所采取的权宜之计往往会损害软件产品的质量，进而不可避免地引起用户的不满。

（2）用户对“已完成的”软件系统不满意的现象经常发生

软件开发人员常常在对用户需求只有模糊的了解，甚至对所要解决的问题还没有确切认识的情况下，就匆忙着手编写程序。软件开发人员和用户之间的交流往往很不充分，“闭门造车”必然导致最终产品不符合用户的实际需要。

（3）软件产品的质量常常不可靠

软件可靠性和质量保证的确切定量概念刚刚出现，软件质量保证技术（审查、复审和测试）

还没有被坚持不懈地应用到软件开发的全过程中，这些都会导致软件产品发生质量问题。

（4）软件常常是不可维护的

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

软件不仅是程序，还应该有一整套文档资料。这些文档资料是在软件开发过程中产生的，而且应该是“最新的”（与代码相符），这些文档资料对于软件开发人员和维护人员来说是至关重要的。缺乏文档必然给软件的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

软件生产是一种脑力劳动，大型软件开发投入人员多，周期长，费用高。

（7）软件开发速度跟不上计算机发展速度和人们对软件的需求

硬件生产率的提高速度远远高于软件，用户对软件的要求也越来越高，软件结构越来越复杂，这种复杂的程度甚至超过了软件行业所能接受的程度。

4. 产生软件危机的原因

软件危机的产生，一方面与软件本身的特点有关，另一方面与开发和维护软件的方式、方法、技术有关。

（1）软件开发规模越来越大，结构越来越复杂

在软件应用范围扩大的同时，软件开发规模也变得更复杂，过去的手工作坊式软件开发模式已经不能适应大型软件的开发。大型软件的开发需要组织大量人员参加，彼此之间很难协调得恰到好处，而且多数程序员没有开发大型软件的经验，参与软件开发的管理者的水平也有待提高。另外，软件开发和维护人员往往对要开发的软件所应用的行业领域并不熟悉，从某个角度讲是外行人在给内行人开发产品。

（2）用户需求不明确且在不断变化

在软件被开发出来之前，用户自己对软件开发的具体需求也不能做到很明晰；用户对软件开发需求的描述不精确，可能有遗漏、有二义性，甚至有错误；软件开发人员对用户需求的理解与用户本来的愿望有差异；有时用户对需求的调整变化可能会导致已完成的工作被抛弃，软件需要重新设计，这不但会影响参与者的积极性，还会增加复杂性，带来更多的错误。

（3）缺乏正确的理论方法指导

缺乏有力的方法学和工具方面的支持。由于软件开发过程是复杂的逻辑思维过程，其产品极大程度地依赖于开发人员个人的技巧和创造性，即便开发人员再谨慎也存在出错的概率，开发人员一点点小的疏忽可能会导致灾难性的后果。软件产品的质量也较难评价。

（4）代码文档贫乏

贫乏或者不规范的文档使得代码维护和修改变得异常艰辛，其结果是带来许多错误。事实上，在许多机构并不鼓励其程序员为代码编写文档，也不鼓励程序员将代码写得清晰和容易理解，相反他们认为少写文档可以更快地进行编码，无法理解的代码更易于工作的保密（“写得艰难必定读得痛苦”）。

（5）软件开发工具

可视化工具、类库、编译器、脚本工具等常常会将自身的错误带到应用软件中。

5. 解决软件危机的途径

（1）技术措施

寻找、创造、使用更好的、更科学的、更适合的软件开发方法、软件开发技术和软件开发工具。

（2）组织管理措施

软件开发不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。

1.2 软件工程简述

为了克服软件危机，人们从其他产业的工程化生产中得到启示。1968 年秋季，北大西洋公约组织的世界著名计算机科学家在联邦德国召开国际会议，讨论软件危机问题，在这次会议上正式提出“软件工程”一词，一门新兴的学科诞生了。如果把开发软件技术比作工匠的盖房技术，那么软件工程学就可比作一整套的现代建筑学体系。

1.2.1 软件工程定义

软件工程是指导计算机软件开发和维护的工程学科，以提高软件质量、降低软件开发成本为主要目的。它是一门综合性的交叉学科，涉及计算机科学、工程科学、管理科学和数学科学等领域。计算机科学和数学用于构造模型与算法，工程科学用于制定规范、分析与设计、评估成本及确定权衡，管理科学用于计划、资源、质量和成本的管理。

关于软件工程（Software Engineering，SE）的定义，国标（GB）中指出，软件工程是应用于计算机软件的定义、开发和维护的一整套方法、工具、文档、实践标准和工序。

1968 在北大西洋公约组织会议（NATO 会议）上，德国人 Fritz Bauer 认为：“软件工程是建立并使用完善的工程化原则，以较经济的手段获得能在实际机器上有效运行的可靠软件的一系列方法。”

1993 年，IEEE（Institute of Electrical & Electronics Engineers，电气和电子工程师学会）给出了一个更加综合的定义：“软件工程是：①将系统的、规范的、可度量的方法应用于软件的开发、运行和维护过程，即将工程化应用于软件；②对①中所述方法进行研究。”

这些主要思想都强调在软件开发过程中需要应用工程化原则。

软件工程包括 3 个要素，即方法、工具和过程，方法是完成软件工程项目的技术手段，工具支持软件的开发、管理、文档生成，过程支持软件开发各个环节的控制、管理。

软件工程的核心思想是把软件产品看做其他工业产品一样处理。把需求分析、可行性研究、工程审核、质量监督、后期维护等工程化的概念引入到软件生产当中，以确保进度、经费、质量等目标。同时，软件还不同于其他工业产品，人们基于软件的特性还提出了一些新的技术方法，如结构化方法、面向对象方法和软件开发模型等。

1.2.2 软件工程目标与原理

1. 软件工程目标

软件工程的目标是成功创建一个软件构造系统，该软件构造系统具有如下几个特性：花费

较低的成本，获得较好的软件性能；开发的软件满足用户的需求、易于移植、可靠性高、维护方便且费用较低；能按时完成开发任务，及时交付使用。

事实上，如图 1.1 所示，上述软件工程目标之间存在着相互制约的关系，要想使所有目标都达到理想的程度是非常困难的，软件工程的总体目标就是力求相互平衡。

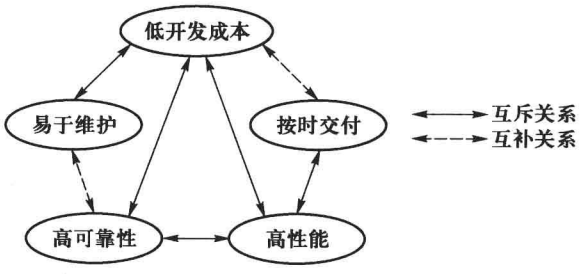

图 1.1　软件工程目标之间的关系

2. 软件工程原理

为了达到上述软件工程目标，在开发软件的过程中必须遵循软件工程的基本原理。这些基本原理包括抽象、信息隐藏、模块化、局部化、确定性、一致性、完备性和可验证性。

（1）抽象（Abstraction）

抽取事物最基本的特性和行为，忽略非基本的细节。采用分层次抽象，自顶向下、逐层细化的办法控制软件开发过程的复杂性。

（2）信息隐藏（Information Hiding）

将模块设计成“黑箱”，使用封装技术，将实现的细节隐藏在模块内部，不让模块的使用者直接访问。使用者只能通过简单的模块接口访问模块中封装的数据。

（3）模块化（Modularity）

模块是程序中逻辑上相对独立的成分，是独立的编程单位，应有良好的接口定义。如 C 语言程序中的函数过程、C++语言程序中的类。模块化有助于实现信息隐藏和抽象，有助于表示复杂的系统。模块大小要适中，模块过大会使模块内部复杂性增加，不利于模块的理解、修改、调试和重用；模块太小会使整个系统表现过于复杂，不利于控制软件系统的复杂性。模块之间相互连接的紧密程度用耦合度（Coupling）来度量，模块内部各元素结合的紧密程度用内聚度（Cohesion）来度量。

（4）局部化（Localization）

要求在一个物理模块内集中逻辑上相互关联的计算机资源，保证模块之间具有松散的耦合度，模块内部具有较强的内聚度，这有助于控制分解的复杂性。

（5）确定性（Certainty）

在软件开发过程中所有概念的表达应是确定的、无歧义性的、规范的，这有助于人们交流时不会产生误解、遗漏，保证整个开发工作协调一致。

（6）一致性（Consistency）

整个软件系统（包括程序、文档和数据）的各个模块应使用一致的概念、符号和术语。程序内部接口应保持一致，软件、硬件和操作系统的接口应保持一致，系统规格说明与系统行为应保持一致。

（7）完备性（Completeness）

软件系统不丢失任何重要成分，可以完全实现系统所要求的功能。为了保证系统的完备性，在软件开发和运行过程中需要进行严格的技术评审。

（8）可验证性（Verifiability）

开发大型的软件系统时需要对系统按自顶向下的方法逐层进行分解。系统分解应遵循系统易于检查、测试、评审的原则，以确保系统的正确性。

1.2.3 软件工程的基本原则

美国著名软件工程专家 Boehm 综合有关专家和学者的意见并总结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 条基本原则，这是软件工程的基本准则和信条。Boehm 认为，这 7 条原则是确保软件产品质量和开发效率原理的最小集合。它们是相互独立的，是缺一不可的最小集合，同时它们又是相当完备的。

1. 用分阶段的生存周期计划严格管理

有人经统计发现，在不成功的软件项目中有一半左右是由于计划不周造成的，可见建立完善计划的重要性。在软件开发与维护的漫长生存周期中，有各种性质的工作需要完成。应该把软件生存周期划分成若干个阶段，并相应地制定出切实可行的计划，然后严格按照计划对软件的开发与维护工作进行管理。Boehm 认为，在软件的整个生存周期中应该制定并严格执行 6 类计划，分别是项目概要计划、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。不同层次、不同工作性质的人员都必须严格按照计划各尽其职地工作，绝不能受其他因素影响而擅自背离预定计划。

2. 坚持进行阶段评审

软件的质量保证工作绝对不能等到编码阶段结束之后再进行。这样说至少有两个理由：第一，大部分错误是在编码之前造成的，例如，根据 Boehm 等人的统计，设计错误占软件错误的 63%，编码仅占 37%；第二，发现错误后，改正得越晚，所需付出的代价就越大。因此，在每个阶段都要进行严格的评审，以便尽早发现在软件开发过程中所犯的错误，是一条必须遵循的重要原则。

3. 实行严格的产品控制

在软件开发过程中不应随意改变需求，因为改变一项需求往往需要付出较高的代价，但是，在软件开发过程中改变需求又是难免的。由于外部环境的变化，用户相应地改变需求是一种客观需要，显然不能硬性禁止用户提出改变需求的要求，而只能依靠科学的产品控制技术来顺应这种要求。也就是说，当改变需求时，为了保持软件各个配置成分的一致性，必须实行严格的产品控制，其中主要是实行基准配置管理。基准配置又称为基线配置，它们是经过阶段评审后的软件配置成分（各个阶段产生的文档或程序代码）。基准配置管理也称为变动控制：一切有关修改软件的建议，特别是涉及对基准配置的修改建议，都必须严格按照规程进行评审，获得批准以后才能实施修改。不允许任何人随意修改软件（包括尚在开发过程中的软件）。

4. 采用现代程序设计技术

从提出软件工程的概念开始，人们一直将主要精力放在研究各种新的程序设计技术上。20 世纪 60 年代末出现的结构化程序设计技术已经成为绝大多数人公认的先进的程序设计技术。之后又进一步发展出各种结构分析（Structured Analysis，SA）与结构设计（Structured Design，SD）技术。实践表明，采用先进的技术既可提高软件开发的效率，又可提高软件维护的效率。