普通高等教育“十一五”国家级规划教材

21世纪高等教育计算机规划教材

COMPUTER

软件工程实用教程（第3版）

Software Engineering Practical Tutorial

郭宁 闫俊伢 主编

樊东燕 赵 怡 董妍汝 副主编

- 精心设计案例，体现简单实用目标
- 贯彻实践创新，注重应用能力培养
- 落实教指委精神，方便组织教学

中国工信出版集团

人民邮电出版社
POSTS & TELECOM PRESS

普通高等教育“十一五”国家级规划教材

21世纪高等教育计算机规划教材

COMPUTER

软件工程实用教程（第3版）

Software Engineering Practical Tutorial

郭宁 闫俊伢 主编

樊东燕 赵 怡 董妍汝 副主编

人 民 邮 电 出 版 社

北 京

图书在版编目（CIP）数据

软件工程实用教程 / 郭宁，闫俊伢主编. -- 3版
. -- 北京 : 人民邮电出版社，2015.8
21世纪高等教育计算机规划教材
ISBN 978-7-115-39332-6

Ⅰ. ①软… Ⅱ. ①郭… ②闫… 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(2015)第150010号

内 容 提 要

本书根据软件工程的最新发展，结合目前软件工程教学的需要，围绕软件工程的三大要素——过程、方法和工具，遵循软件开发“工程化”思想，结合大量的应用案例，系统地介绍软件工程的理论、方法以及应用技术。本书内容包括：软件工程引论、软件开发过程模型、需求工程、软件分析与设计、软件测试、软件维护、质量管理、文档技术、软件项目管理、软件开发工具与环境、软件工程课程设计等。

本书强调软件工程的理论与实践相结合，技术与管理相结合，方法与工具相结合。全书语言简练、通俗易懂，采用案例教学方法，注重培养实际开发能力和文档的写作能力，具有很强的实用性和可操作性。书中例题与习题丰富，便于教学和自学。

本书可作为高等院校计算机专业或信息类相关专业高年级本科生或研究生教材，也可作为软件开发人员的参考用书。

◆ 主　　编　郭　宁　闫俊伢
副 主 编　樊东燕　赵　怡　董妍汝
责任编辑　邹文波
责任印制　沈　蓉　彭志环
◆ 人民邮电出版社出版发行　　北京市丰台区成寿寺路 11 号
邮编　100164　　电子邮件　315@ptpress.com.cn
网址　http://www.ptpress.com.cn
北京昌平百善印刷厂印刷
◆ 开本：787×1092　1/16
印张：21　　2015 年 8 月第 3 版
字数：554 千字　　2015 年 8 月北京第 1 次印刷

定价：45.00 元

读者服务热线：(010)81055256　印装质量热线：(010)81055316
反盗版热线：(010)81055315

第 3 版前言

在信息时代，开发软件系统需要掌握大规模软件开发的专业知识。软件工程的主要目标是开发系统模型以及按时并在有限预算下生产出高质量的软件。软件工程是将系统性的、规范化的、可定量的方法应用于软件的开发、运行和维护，它涉及技术、方法、管理等诸多方面。该课程对学生开发能力和管理素质的培养起着重要的作用，因此，它是计算机科学与技术、计算机工程、软件工程以及信息技术相关专业的核心基础课程之一。为适应计算机应用发展的需要，根据教育部高等学校计算机科学与技术教育指导委员会提出的《高等学校计算机科学与技术专业公共核心知识体系与课程》的要求，结合软件工程学科的发展，在分析了国内外多种同类教材的基础上，我们精心编写了本书。

2006 年编者曾经编写出版了《软件工程实用教程》，该书作为国家级“十一五”规划教材正式出版并投入使用。在 2011 年完成了第 2 版的修订，经过 3 年的实际使用，在征求教师和学生意见的基础上，我们对第 2 版进行了适当的修改，现已完成第 3 版的修订任务。《软件工程实用教程（第 3 版）》在第 2 版的基础上做了如下补充与修改。

1. 第 3 章将“数据流程图”及“数据字典”部分重新修订，增加了“结构化分析建模”一节，增强读者对结构化分析方法的理解。

2. 第 5 章改写了面向对象分析中的部分案例；修订了部分概念的定义。

3. 第 6 章在介绍软件逻辑架构设计建模之前增加了“常用软件架构风格”方面的内容，使读者对软件架构和不同类型软件的架构风格有一定的了解和认识。

4. 第 7 章修订了“黑盒测试方法”中“因果图方法”的内容，补充了因果图的基本符号、因果关系、约束关系等内容，并重新改写了对应案例，补充了“错误推测法”测试用例的相关内容，使读者对该方法有一个更直观的理解和认识。根据软件测试流程的工程实践对本节结构进行了重新编排，对本节内容进行了精练和补充。

5. 第 9 章增加了“软件质量管理”的内容，使读者全面了解软件质量的知识；第 10 章对软件工程国家标准进行了更新。

6. 第 11 章补充了“软件项目时间管理”中“前导图法和箭线图法”的相关内容，增加了“软件项目成本管理”中成本预算和控制的相关例题，便于读者对该方法的理解和掌握。

7. 第 12 章增加了“软件开发环境的概念”“软件开发工具的概念”“软件开发工具的功能”“常见工具的使用”等内容，使读者对软件开发环境和工具的基本知识有一个全面的认识和了解；同时增加了对 PowerBuilder 开发工具的介绍以及 NetBeans 集成开发环境、Rational Rose 建模工具、Microsoft Project 软件项目管理工具使用方法的介绍，加深读者在后续学习中对软件工具使用的理解和掌握。

8. 第 13 章增加了“网上书店系统开发案例”“手机购物网站 APP 开发案例”，

按照软件工程的基本流程，从需求分析、系统分析与设计到系统的实现、系统测试进行了详细的介绍，力求使读者能够从整体上对软件开发有个全面的认识和了解，加深读者对软件工程的理解。

9. 根据软件工程应用实践重新编排了练习题，方便学生对整章内容进行整体性的把握，使学生在学习、消化和运用知识的过程中能够获得更多的启示。

本书具有以下特色。

1. 每章重新设计了练习题。题型有判断题、选择题、简答题和应用题，使读者在理解知识的基础上，熟练掌握课程所要求的基本能力和基本知识。练习题设计力求做到针对性强、内容全面、考查细致。

2. 本书对软件工程的基本理论进行了系统的介绍，在案例选取和内容组织时注重实践性和可操作性。对具体知识点配有丰富的例题，极大地方便了读者对抽象、枯燥的软件工程理论的理解和掌握。

3. 理论联系实际，案例丰富，启发性强。在各章中，针对每个知识点都有形象具体的举例，同时，配合软件工程相关核心理论的方法的阐述，每章中一般都配有一到两个贯穿整个章节的完整案例。本书的第5章和第6章以“网上计算机销售系统”为例详细阐述了采用面向对象方法进行软件系统开发的分析、设计和实现过程；第12章介绍了常用的软件开发环境和工具的基本知识及使用方法；第13章介绍了“嵌入式软件系统应用实例”“网上书店系统开发案例”“手机购物网站 APP 开发案例”3个软件系统开发实例，详细阐述了分析和设计的重要过程，可以启发读者思考，从中学会发现问题并解决问题的方法。

4. 结合目前软件工程领域的新发展，对书中的案例、软件工程方法、软件工程标准等进行了更新，使教材的组织结构更加合理，内容更加完整。

本书由郭宁、闫俊伢任主编，樊东燕、赵怡、董妍汝任副主编。其中，首都经济贸易大学的郭宁编写第4章，山西大学商务学院的闫俊伢编写第2章、第3章，樊东燕编写第9章，赵怡编写第5章、第6章，董妍汝编写第8章、第10章，马晓慧编写第7章、第11章，杨森编写第12章、第13章，郑文娟编写第1章和附录。全书由郭宁、闫俊伢统稿。

本书既可作为高等院校计算机科学与技术、计算机工程、软件工程以及信息技术相关专业的教材，也可供企事业单位和信息系统相关人员参考使用。

在本书的编写、修订过程中得到了首都经济贸易大学信息学院、山西大学商务学院信息学院的领导和同事们的支持与帮助，在此一并表示感谢。

由于编者水平有限，书中难免存在不妥与疏漏之处，敬请广大读者批评指正。

编　者

2015年4月

目 录

第1章 软件工程引论

21世纪是高度依赖计算机信息系统的时代，面对大量的计算机应用需求，怎样才能更有效地开发出各种不同类型的软件，是软件开发技术与软件工程所要解决的问题。软件工程是应用计算机科学、管理学、数学、项目管理、质量管理、软件人类工程学及系统工程的原则和方法来创建软件的学科，它对指导软件开发、质量控制以及开发过程的管理起着重要的作用。本章将概括地介绍软件的基本概念与特点，软件工程学科的诞生背景与形成，软件工程学研究的内容与对象等，使读者对软件工程与软件开发技术有所认识。

本章学习目标：

1. 掌握软件的定义与特点
2. 了解软件危机以及软件危机产生的原因
3. 掌握软件工程的定义、目标和原则
4. 了解软件工程的研究内容与对象
5. 了解软件工程的知识体系
6. 明确学习软件工程的意义

1.1 软件及软件危机

随着计算机应用领域的扩大，软件规模也越来越大，复杂程度不断增加，使得软件生产的质量、周期、成本难以预测和控制，从而出现了软件危机。软件工程正是为了解决软件危机而提出的，其目的是改善软件生产的质量，提高软件的生产效率。经过几十年的实践与探索，软件工程正在逐步发展成为一门成熟的专业学科，在软件产业的发展中起到重要的技术保障和促进作用。

1.1.1 软件及其特性

软件是计算机系统的思维中枢，是软件产业的核心。作为信息技术的灵魂，计算机软件在现代社会中起着极其重要的作用。

1. 软件

计算机软件是由计算机程序的发展而形成的一个概念。它是与计算机系统操作有关的程序、规程、规则及其文档和数据的统称。软件由两部分组成：一是机器可执行的程序和有关的数据；二是与软件开发、运行、维护、使用和培训有关的文档。

程序是按事先设计的功能和性能要求执行的语句序列。数据是程序所处理信息的数据和数据

结构。文档则是与程序开发、维护和使用相关的各种图文资料，如各种规格说明书、设计说明书、用户手册等。在文档中记录着软件开发的活动和阶段成果。

2. 软件的特点

软件是一种逻辑产品而不是实物产品，软件功能的发挥依赖于硬件和软件的运行环境，没有计算机硬件的支持，软件毫无实用价值。若要对软件有一个全面而正确的理解，我们应从软件的本质、软件的生产等方面剖析软件的特征。

（1）软件固有的特性

① 复杂性。软件是一个庞大的逻辑系统。一方面在软件中要客观地体现人类社会的事务，反映业务流程的自然规律；另一方面在软件中还要集成多种多样的功能，以满足用户在激烈的竞争中对大量信息及时处理、传输、存储等方面的需求，这就使得软件变得十分复杂。

② 抽象性。软件是人们经过大脑思维后加工出来的产品，一般寄生在内存、磁盘、光盘等载体上，我们无法观察到它的具体形态，这就导致了软件开发不仅工作量难以估计，进度难以控制，而且质量也难以把握。

③ 依赖性。软件必须和运行软件的机器（硬件）保持一致，软件的开发和运行往往受到计算机硬件的限制，对计算机系统有着不同程度的依赖性。软件与计算机硬件的这种密切相关性与依赖性，是一般产品所没有的特性。为了减少这种依赖性，有关人员在软件开发中提出了软件的可移植性问题。

④ 软件使用特性。软件的价值在于应用。软件产品不会因多次反复使用而磨损老化，一个久经考验的优质软件可以长期使用。由于用户在选择新机型时，通常会提出兼容性要求，所以一个成熟的软件可以在不同型号的计算机上运行。

（2）软件的生产特性

① 软件开发特性。由于软件固有的特性，使得软件的开发不仅具有技术复杂性，还有管理复杂性。技术复杂性体现在软件提供的功能比一般硬件产品提供的功能多，而且功能的实现具有多样性，需要在各种实现中做出选择，更有实现算法上的优化带来的不同，而实现上的差异会带来使用上的差别。管理上的复杂性表现在：第一，软件产品的能见度低（包括如何使用文档表示的概念能见度），要看到软件开发进度比看到有形产品的进度困难得多；第二，软件结构的合理性差，结构不合理使软件管理复杂性随软件规模增大而呈指数增长。因此，领导一个人数众多的项目组织进行规模化生产并非易事，软件开发比硬件开发更依赖于开发人员的团队精神、智力，以及对开发人员的组织与管理。

② 软件产品形式的特性。软件产品在设计阶段成本高昂，而在生产阶段成本极低。硬件产品试制成功之后，批量生产需要建设生产线，投入大量的人力、物力和资金，生产过程中还要对产品进行质量控制，对每件产品进行严格的检验。然而，软件是把人的知识与技术转化为信息的逻辑产品，开发成功之后，只需对原版软件进行复制即可；大量人力、物力、资金的投入，以及质量控制，软件产品检验都是在软件开发中进行的。由于软件的复制非常容易，软件的知识产权保护就显得极为重要。

③ 软件维护特性。软件在运行过程中的维护工作比硬件复杂得多。首先，软件投入运行后，总会存在缺陷甚至暴露出潜伏的错误，需要进行“纠错性维护”。其次，用户可能要求完善软件性能，对软件产品进行修改，进行“完善性维护”。当支撑软件产品运行的硬件或软件环境改变时，也需要对软件产品进行修改，进行“适应性维护”。软件的缺陷或错误属于逻辑性的，因此不需要更换某种备件，而是修改程序，纠正逻辑缺陷，改正错误，提高性能，增加适应性。当软件产品规模庞大、内部的逻辑关系复杂时，经常会发生纠正一个错误而产生新错误的情况，因此，软件产品的维护要比硬件产品的维护工作量大而且复杂。

1.1.2　软件危机

20 世纪 60～70 年代，由于软件规模的扩大、功能的增强和复杂性的增加，使得在一定时间内仅依靠少数人开发一个软件变得越来越困难。在软件开发中经常会出现时间延迟、预算超支、质量得不到保证、移植性差等问题，甚至有的项目在耗费了大量人力、财力后，由于实际产品离目标相差甚远而宣布失败。这种情况使人们认识到“软件危机”的存在。

1. 软件危机的表现

（1）软件生产率低。软件生产率提高的速度远远跟不上计算机应用迅速普及和深入的趋势。落后的生产方式与开发人员的匮乏，使得软件产品的供需差距不断扩大。由于缺乏系统有效的方法，现有的开发知识、经验和相关数据难以积累与复用。另外，低水平的重复开发过程浪费了大量的人力、物力、财力和时间。人们为不能充分发挥计算机硬件提供的巨大潜力而苦恼。

（2）软件产品常常与用户要求不一致。开发人员与用户之间的信息交流往往存在障碍，除了知识背景的差异，缺少合适的交流方法及需求描述工具也是一个重要原因。这使得获取的需求经常存在二义性、遗漏，甚至是错误。由于开发人员对用户需求的理解与用户的本意有所差异，以致造成开发中后期需求与现实之间的矛盾集中暴露。

（3）软件规模的增长，带来了复杂度的增加。由于缺乏有效的软件开发方法和工具的支持，过分依靠程序设计人员在软件开发过程中的技巧和创造性，所以，软件的可靠性往往随着软件规模的增长而下降，质量保障越来越困难。

（4）不可维护性突出。软件的局限性和欠灵活性，不仅使错误非常难改正，而且不能适应新的硬件环境，也很难根据需要增加一些新的功能。整个软件维护过程除了程序之外，没有适当的文档资料可供参考。

（5）软件文档不完整、不一致。软件文档是计算机软件的重要组成部分，在开发过程中，管理人员需要使用这些文档资料来管理软件项目；技术人员则需要利用文档资料进行信息交流；用户也需要通过文档来认识软件，对软件进行验收。但是，由于软件项目管理工作的不规范，软件文档往往不完整、不一致，这给软件的开发、交流、管理、维护等都带来了困难。

2. 产生软件危机的原因

软件危机是指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。这些问题不仅局限于那些“不能正确完成功能”的软件，还包括我们如何开发软件，如何维护大量已有软件，如何使软件开发速度与对软件需求增长相适应等问题。产生软件危机的原因主要有以下几点。

（1）软件独有的特点给开发和维护带来困难。由于软件的抽象性、复杂性与不可预见性，使得软件在运行之前，开发过程的进展情况较难衡量，软件的错误发现较晚，软件的质量也较难评价，因此，管理和控制软件开发过程相当困难。此外，软件错误具有隐蔽性，往往在很长时间里软件仍可能需要改错。这在客观上使得软件维护较为困难。

（2）软件人员的错误认识。相当多的软件专业人员对软件开发和维护还有不少的错误观念。例如，认为软件开发就是编写程序，忽视软件需求分析的重要性，轻视文档的作用，轻视软件维护等。这些错误认识加重了软件危机的影响。

（3）软件开发工具自动化程度低。尽管软件开发工具比 30 年前已经有了很大的进步，但直到今天，软件开发仍然离不开工程人员的个人创造与手工操作，软件生产仍不可能向硬件设备的生产那样，达到高度自动化。这样不仅浪费了大量的财力、物力和宝贵的人力资源，无法避免低水平的重复性劳动，而且软件的质量也难以保证。此外，软件生产工程化管理程度低，致使软件项

目管理混乱，难以保障软件项目成本、开发进度按计划执行。

1.2 软件工程的形成与概念

为了克服“软件危机”，1968年在北大西洋公约组织（NATO）召开的计算机科学会议上，Fritz Bauer首先提出了“软件工程”的概念，试图用工程的方法和管理手段将软件开发纳入工程化的轨道，以便开发出成本低、功能强、可靠性高的软件产品。几十年来，人们一直在努力探索克服软件危机的途径。

1.2.1 软件工程的形成与发展

自1968年NATO会议上提出软件工程这一概念以来，人们一直在寻求更先进的软件开发的方法与技术。当出现一种先进的方法与技术时，就会使软件危机得到一定程度的缓解。然而，这种进步又促使人们把更多、更复杂的问题交给计算机去解决，于是又需要探索更先进的方法与技术。几十年来，软件工程研究的范围和内容也随着软件技术的发展不断变化和拓展。软件工程的发展经历了以下3个阶段。

第一阶段：20世纪70年代，为了解决软件项目失败率高、错误率高以及软件维护任务重等问题，人们提出了软件生产工程化的思想，希望使软件生产走上正规化的道路，并努力克服软件危机。人们发现将传统工程学的原理、技术和方法应用于软件开发，可以起到使软件生产规范化的作用。它有利于组织软件生产，提高开发质量，降低成本和控制进度。随后，人们又提出了软件生命周期的概念，将软件开发过程划分为不同阶段（需求分析、概要与详细设计、编程、测试、维护等），以适应更加复杂的应用。人们还将计算机科学和数学用于构造模型与算法上，围绕软件项目开展了有关开发模型、方法以及支持工具的研究，并提出了多种开发模型、方法与多种软件开发工具（编辑、编译、跟踪、排错、源程序分析、反汇编、反编译等），并围绕项目管理提出了费用估算、文档评审等一些管理方法和工具，基本形成了软件工程的概念、框架、方法和手段，成为软件工程的第一代——传统软件工程时代。

第二阶段：20世纪80年代，面向对象的方法与技术受到了广泛的重视，Smalltalk-80的出现标志着面向对象的程序设计进入了实用和成熟阶段。20世纪80年代末逐步发展起来的面向对象的分析与设计方法，形成了完整的面向对象技术体系，使系统的生命周期更长，适应更大规模、更广泛的应用。这时，进一步提高软件生产率、保证软件质量就成为软件工程追求的更高目标。软件生产开始进入以过程为中心的第二阶段。这个时期人们认识到，应从软件生命周期的总费用及总价值来决定软件开发方案。在重视发展软件开发技术的同时，人们提出软件能力成熟度模型、个体软件过程、群组软件过程等概念。在软件定量研究方面提出了软件工作量估计COCOMO模型等。软件开发过程从目标管理转向过程管理，形成了软件工程的第二代——过程软件工程时代。

第三阶段：进入20世纪90年代以后，软件开发技术的主要处理对象为网络计算和支持多媒体信息的WWW。为了适合超企业规模、资源共享、群组协同工作的需要，企业需要开发大量的分布式处理系统。这一时期软件工程的目的在于不仅提高个人生产率，而且通过支持跨地区、跨部门、跨时空的群组共享信息，协同工作来提高群组、集团的整体生产效率。因整体性软件系统难以更改、难以适应变化，所以提倡基于构件的开发方法——即部件互连及集成。同时人们认识到计算机软件开发领域的特殊性，不仅要重视软件开发方法和技术的研究，更要重视总结和发展包

括软件体系结构、软件设计模式、互操作性、标准化、协议等领域的复用经验。软件复用和软件构件技术正逐步成为主流软件技术，软件工程也由此进入了新的发展阶段——构件软件工程时代。

1.2.2　软件工程的基本概念

软件工程这一概念已提出 40 多年，人们对软件工程的理解是不断深入的。作为一门新兴的交叉性学科，它所研究的对象、适用范围和所包含的内容都在不断发展和变化。

1. 软件工程的定义

在 NATO 会议上，软件工程被定义为："为了经济地获得可靠的和能在实际机器上高效运行的软件，而建立和使用的健全的工程原则"。这个定义虽然没有提到软件质量的技术层面，也没有直接谈到用户满意程度或要求按时交付产品等问题，但人们已经认识到借鉴和吸收人类对各种工程项目开发的经验无疑对软件的开发是有益的。

软件工程是指导计算机软件开发和维护的工程学科。它强调按照软件产品的生产特性，采用工程的概念、原理、技术和方法来开发与维护软件，把经过时间考验而证明正确的管理技术和当前最好的技术结合起来，以便经济地开发出高质量的软件并有效地维护它。

由于引入了软件工程的思想，其他工程技术研究和开发领域中行之有效的知识和方法被运用到软件开发工作中来，人们又提出了按工程化的原则和方法组织软件开发工作的解决思路和具体方法，这在一定程度上缓解了"软件危机"。

2. 软件工程的目标

软件工程的目标是基于软件项目目标的成功实现而提出的，主要体现在以下几方面。

① 软件开发成本较低。

② 软件功能能够满足用户的需求。

③ 软件性能较好。

④ 软件可靠性高。

⑤ 软件易于使用、维护和移植。

⑥ 能按时完成开发任务，并及时交付使用。

在实际开发中，试图让以上几个质量目标同时达到理想的程度往往是不现实的。软件工程目标之间存在的相互关系如图 1.1 所示。从图 1.1 中可以看出，有些目标之间是相互补充的，如易于维护和高可靠性之间、功能强与可用性之间；有些目标是彼此相互冲突的，如若只考虑降低开发成本，很可能同时也降低了软件的可靠性，如果一味追求提高软件的性能，可能造成开发出的软件对硬件的依赖性较强，从而影响到软件的可移植性；不同的应用对软件质量的要求不同，如对实时系统来说，其可靠性和效率比较重要，对生命周期较长的软件来说，其可移植性、可维护性比较重要。

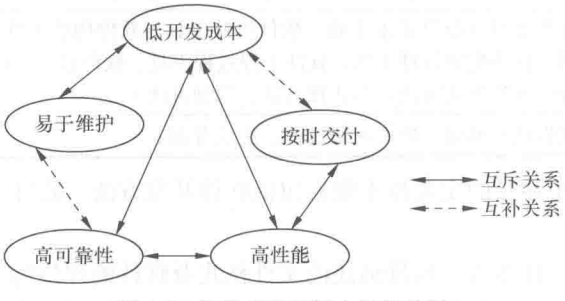

图 1.1　软件工程目标之间的关系

软件工程的首要问题是软件质量。软件工程的目的就是在以上目标的冲突之间取得一定程度的平衡。因此，在涉及平衡软件工程目标这个问题的时候，软件的质量应该摆在最重要的位置加以考虑。软件质量可用功能性、可靠性、可用性、效率、可维护性和可移植性等特性来评价。功能性是指软件所实现的功能能够达到它的设计规范和满足用户需求的程度；可靠性是指在规定的时间和条件下，软件能够正常维持其工作的能力；可用性是指为了使用该软件所需要的能力；效率是指在规定的条件下用软件实现某种功能所需要的计算机资源的有效性；可维护性是指当环境改变或软件运行发生故障时，为了使其恢复正常运行所做努力的程度；可移植性是指软件从某一环境转移到另一环境时所做努力的程度。在不同类型的应用系统中对软件的质量要求是不同的。

3. 软件工程知识体系及知识域

软件工程作为一门学科，在取得对其核心的知识体系的共识方面已经达到了一个重要的里程碑。2005年9月，ISO/IEC JTC1/SC7正式发布为国际标准，即ISO/IEC 19759—2005软件工程知识体系指南（SWEBOK）。SWEBOK将软件工程知识体系划分为10个知识域，包含在两类过程中。一类过程是开发与维护过程，包括软件需求、软件设计、软件构造、软件测试和软件维护；另一类过程是支持过程，包括软件配置管理、软件工程管理、软件工程过程、软件工程工具与方法、软件质量。每个知识域还可进一步分解为若干个论题，在论题描述中引用有关知识的参考文献，形成一个多级层次结构，以此确定软件工程知识体系的内容和边界。每个知识域的具体内容如表1.1所示。有关知识的参考文献涉及相关学科，包括计算机工程、计算机科学、管理学、数学、项目管理、质量管理、软件人类工程学、系统工程等。

表1.1　　软件工程知识体系指南的内容

知 识 域	子 知 识 域
软件需求	软件需求基础、需求过程、需求获取、需求分析、需求规格说明、需求确认、实践考虑
软件设计	软件设计基础、软件设计关键问题、软件结构与体系结构、软件设计质量的分析与评价、软件设计记法、软件设计的策略与方法
软件构造	软件构造基础、管理构造、实际考虑
软件测试	软件测试基础、测试级别、测试技术、与测试相关的度量、测试过程
软件维护	软件维护基础、软件维护关键问题、维护过程、维护技术
软件配置管理	软件配置过程管理、软件配置标识、软件配置控制、软件配置状态报告、软件配置审计、软件发行管理和交付
软件工程管理	项目启动和范围定义、软件项目计划、软件项目实施、评审与评价、项目收尾、软件工程度量
软件工程过程	过程定义、过程实践与变更、过程评估、过程和产品度量
软件工程工具与方法	软件工具（软件需求工具、软件设计工具、软件构造工具、软件测试工具、软件维护工具、软件配置管理工具、软件工程过程工具、软件质量工具和其他工具问题）、软件工程方法（启发式方法、形式化方法、原型方法）
软件质量	软件质量基础、软件质量过程、实践考虑

软件工程知识体系中涉及的主要技术要素包括软件开发方法、软件开发工具和软件过程。

（1）软件开发方法

软件开发方法是在工作步骤、软件描述的文件格式及软件的评价标准等方面做出规定。它主要解决什么时候做什么以及怎样做的问题，是软件工程最核心的研究内容。实践表明，在开发的

早期阶段多做努力，就会使后来的测试和维护阶段缩短，从而大大缩减费用。因此，针对分析和设计阶段的软件开发方法特别受到重视。目前，人们提出了结构化方法、面向数据结构方法、原型化方法、面向对象的方法、形式化方法等多种实用有效的软件开发方法，利用这些方法确实也开发出不少成功的系统，但各种开发方法具有一定的适用范围，所以选择正确的开发方法是非常重要的。

针对软件开发方法的评价一般通过以下 4 个方面来进行。

① 技术特征：支持各种技术概念的方法特征，如层次性、抽象性（包括数据抽象和过程抽象）、并行性、安全性、正确性等。

② 使用特征：具体开发时的有关特征，如易理解性、易转移性、易复用性、工具的支持、使用的广度、活动过渡的可行性、易修改性、对正确性的支持等。

③ 管理特征：对软件开发活动管理的能力方面的特征，如易管理性、支持协同工作的程度、中间阶段的确定、工作产物、配置管理、阶段结束准则和代价等。

④ 经济特征：对软件机构产生的质量和生产力方面的可见效益，如分析活动的局部效益、整个生命周期效益、获得该开发方法的代价、使用它和管理它的代价等。

下面重点介绍一些常用的软件开发方法。

① 结构化方法。结构化方法是传统的基于软件生命周期的软件工程方法，自 20 世纪 70 年代产生以来，获得了极有成效的软件项目应用。结构化方法是以软件功能为目标来进行软件构建的，包括结构化分析、结构化设计、结构化实现、结构化维护等内容。这种方法主要通过数据流模型来描述软件的数据加工过程，并可以通过数据流模型，由对软件的分析过渡到对软件的结构设计。

② JSD 方法。JSD 方法主要用在软件设计上，于 1983 年由法国学者 Jackson 提出。它以软件中的数据结构为基本依据来进行软件结构与程序算法设计，是对结构化软件设计方法的有效补充。在以数据处理为主要内容的软件系统开发中，JSD 方法具有比较突出的设计建模优势。

③ 面向对象方法。面向对象方法是从现实世界中客观存在的事物出发来构造软件，包括面向对象分析、面向对象设计、面向对象实现、面向对象维护等内容。一个软件是为了解决某些问题，这些问题所涉及的业务范围被称作该软件的问题域。面向对象强调以问题域中的事物为中心来思考问题、认识问题，并根据这些事物的本质特征，把它抽象地表示为系统中的对象，作为系统的基本构成单位。确定问题域中的对象成分及其关系，建立软件系统对象模型，是面向对象分析与设计过程中的核心内容。自 20 世纪 80 年代以来，人们提出了许多有关面向对象的方法，其中，由 Booch、Rumbaugh、Jacobson 等人提出的一系列面向对象方法成为了主流方法，并被结合为统一建模语言（UML），成为了面向对象方法中的公认标准。

（2）软件开发工具

软件开发工具是指用来辅助软件开发、维护和管理的软件。现代软件工程方法得以实施的重要保证是软件开发工具和环境。软件开发工具使软件在开发效率、工程质量，以及减少软件开发对人的依赖性等多方面得到改善。软件开发工具与软件开发方法有着密切的关系，软件开发工具是软件方法在计算机上的具体实现。

软件开发环境是方法与工具的结合以及配套软件的有机组合。该环境旨在通过环境信息库和消息通信机制实现工具的集成，从而为软件生命周期中某些过程的自动化提供更有效的支持。集成机制主要实现工具的集成，使之能够系统、有效地支持软件开发。

（3）软件过程

尽管有软件开发工具与工程化方法，但这并不能使软件产品生产完全自动化，它们还需要合

适的软件过程才能真正发挥作用。软件过程是指生产满足需求且达到工程目标的软件产品所涉及的一系列相关活动，它覆盖了需求分析、系统设计、实施以及支持维护等各个阶段。这一系列活动就是软件开发中开发机构需要制订的工作步骤。

软件过程有各种分类方法。按性质划分软件过程可概括为基本过程、支持过程类和组织过程。按特征划分有管理过程、开发过程与综合过程。按人员的工作内容来分类有获取过程、供应过程、开发过程、运作过程、维护过程、管理过程与支持过程。软件过程研究的对象涉及从事软件活动的所有人。提高软件的生产率和质量，其关键在于管理和支持能力。所以，软件过程特别重视管理活动和支持活动。

1.3 软件工程的基本原则

为了保证在软件项目中能够有效地贯彻与正确地使用软件工程规程，需要有一定的原则来对软件项目加以约束。经过长期的实践，著名软件工程专家 B.W.Boehm 提出了以下 7 条软件工程的基本原则。

1. 采用分阶段的生命周期计划，以实现对项目的严格管理

软件项目的开展，需要计划在先，实施在后。统计资料表明，有 50%以上的失败项目是由于计划不周而造成的。在软件开发与维护的漫长生命周期中，需要完成许多性质各异的工作，这意味着，应该把软件生命周期划分为若干个阶段，并相应地制订出切实可行的计划，然后严格按照计划对软件的开发与维护进行管理。

2. 坚持进行阶段评审，以确保软件产品质量

软件的质量保证工作贯穿软件开发的各个阶段。实践表明，软件的大部分错误是编程之前造成的。根据 B.W.Boehm 的统计，设计错误占软件错误的 63%，编码错误仅占 37%。软件中的错误发现与纠正得越晚，所需要付出的代价也越高。因此，在每个阶段都要进行严格的评审，尽早地发现软件中的错误，通过对软件质量实施过程监控，以确保软件在每个阶段都能够具有较高的质量。

3. 实行严格的产品控制，以适应软件规格的变更

在软件开发过程中不应随意改变需求，因为改变一项需求需要付出较高的代价。但在软件开发过程中改变需求又是难免的，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，必须实现严格的产品控制，其中主要是实行基准配置管理。所谓基准配置是指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，是指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，都必须按照严格的规程进行评审，获得批准以后，才能实施修改，绝对不能随意修改软件。实行严格的产品控制，能够对软件的规格进行跟踪记录，使软件产品的各项配置成分保持一致性，由此来适应软件的需求变更。

4. 采用现代程序设计技术

采用先进的软件开发和维护技术，不仅能够提高软件开发和维护效率，而且可以提高软件产品的质量，降低开发成本，缩短开发时间，增加软件的使用寿命。例如，构件架构系统的特点是通过创建比“类”更加抽象、更具有通用性的基本构件，使软件开发如同可插入的零件一样装配，这样的软件不仅开发容易，维护方便，而且可以根据用户的特定需求方便地进行改装。