国家示范性软件学院系列教材

本书为教师配有电子教案

软件工程案例教程

韩万江 编著

软件企业的实际案例贯穿始终

- 内容全面、系统
- 汇集大量教学、实践经验
- 案例实用、可操作性强

机械工业出版社
China Machine Press

国家示范性软件学院系列教材

软件工程案例教程

韩万江 编著

软件企业的实际案例
贯穿始终

- 内容全面、系统
- 汇集大量教学、实践经验
- 案例实用、可操作性强

机械工业出版社
China Machine Press

本书以案例的形式，讲述了软件开发的全过程，包括软件开发中需求分析、概要设计、详细设计、编码、测试、提交以及维护等各个过程中涉及的理论、方法、注意事项、提交的产品和文档等。本书注重实效，讲解系统、全面，通过对案例的学习，读者可以在短时间内掌握软件开发的基本知识、基本过程，并有效提高实践能力。

本书既适合作为高等院校计算机及相关专业软件工程、软件测试课程的教材，也适合作为广大软件技术人员的培训教程，同时可以供软件开发人员参考。

图书在版编目（CIP）数据

软件工程案例教程/韩万江编著. -北京：机械工业出版社，2007.5
（国家示范性软件学院系列教材）·
ISBN 978-7-111-20667-5

Ⅰ.软…　Ⅱ.韩…　Ⅲ.软件开发-高等学校-教材　Ⅳ.TP311.52

中国版本图书馆CIP数据核字（2007）第164221号

机械工业出版社（北京市西城区百万庄大街22号　邮政编码　100037）
责任编辑：李东震
北京诚信伟业印刷有限公司印刷 · 新华书店北京发行所发行
2007年5月第1版第1次印刷
184mm×260mm · 20.5印张
定价：29.00元

凡购本书，如有倒页、脱页、缺页，由本社发行部调换
本社购书热线：（010）68326294

前　言

计算机软件行业与传统行业（如机械行业、建筑行业等）相比，还是比较年轻的行业。那些传统行业在生产、管理和工艺上都相对成熟，而软件行业在各个环节上都还不是很成熟。甚至有人提出，现在的软件开发根本谈不上是“工程”，因为它太稚嫩了，目前还没有一套成熟完善的评估标准，因而软件项目极易出现失败或失误。目前的软件工程理论还不能完全解决软件危机中的所有问题，还有待于提高和改善。软件工程领域包括三个重要方面：软件开发、软件项目管理、过程改进。为此，作者根据多年的软件项目经验，以实际的案例为依据，编著软件工程案例系列教程，包括《软件工程案例教程》、《软件项目管理案例教程》和《软件过程改进案例教程》三本书。本书主要讲述的是软件开发方面包含的重要技术。

本书总结了作者多年软件项目实践与教学过程的经验。书中以一个软件开发项目的实例贯穿始终，以路线图为导引讲述软件项目的开发过程。本书既可作为高等院校软件工程或软件测试的教材，对软件开发和管理人员也有一定的帮助，可以帮助读者在短时间内，更加系统、完整地掌握软件开发的基本过程和应完成的任务，同时对于规范软件企业软件流程也很有帮助。

最后，衷心希望这本书对读者有一定的帮助。当然，由于作者水平有限，难免有疏漏之处，诚请各位读者批评指正，以便在下版中进行完善。如果存在什么问题，请发E-mail：casey_han@263.net。

韩万江

2006年11月

目　录

前言

第1章　软件工程的实质 ……1
1.1　软件工程引发的故事 ……1
1.2　软件工程知识体系 ……3
1.3　软件工程的三段论 ……6
1.3.1　软件开发过程 ……7
1.3.2　软件项目管理 ……8
1.3.3　软件过程改进 ……9
1.4　软件工程生存期模型 ……12
1.4.1　瀑布模型 ……12
1.4.2　V模型 ……13
1.4.3　原型模型 ……13
1.4.4　增量式模型 ……14
1.4.5　螺旋式模型 ……14
1.5　软件工程中的复用原则 ……15
1.6　小结 ……17
1.7　练习题 ……17
第2章　软件项目的需求分析 ……18
2.1　概述 ……18
2.1.1　需求定义 ……19
2.1.2　需求类型 ……20
2.2　需求工程 ……20
2.2.1　需求获取 ……21
2.2.2　需求分析 ……23
2.2.3　需求规格编写 ……24
2.2.4　需求验证 ……25
2.2.5　需求变更管理 ……26
2.3　需求建模的基本方法 ……26
2.3.1　关联模型 ……27
2.3.2　行为模型 ……27
2.3.3　数据模型 ……29
2.3.4　结构化方法 ……29
2.3.5　面向对象模型 ……30
2.3.6　其他方法 ……41
2.4　需求分析过程 ……43
2.5　需求规格文档 ……44
2.6　案例说明 ……47
2.7　小结 ……59
2.8　练习题 ……59
第3章　软件项目的概要设计 ……60
3.1　软件设计定义 ……60
3.2　概要设计方法概论 ……61
3.2.1　传统（结构化）的设计方法 ……61
3.2.2　面向对象的设计方法 ……63
3.3　设计模型 ……72
3.3.1　架构设计 ……72
3.3.2　数据设计 ……74
3.3.3　用户界面设计 ……76
3.3.4　模块设计 ……82
3.4　体系结构介绍 ……83
3.4.1　主机 ……83
3.4.2　客户机/服务器（C/S）……83
3.4.3　浏览器/服务器（B/S）……84
3.4.4　应用程序框架的概念 ……85
3.4.5　struct体系结构 ……86
3.5　设计原则 ……87
3.6　概要设计过程 ……88
3.7　概要设计文档标准 ……89
3.8　案例说明 ……91
3.9　小结 ……115
3.10　练习题 ……115

第4章 软件项目的详细设计 ……116
4.1 详细设计的概念 ……116
4.2 详细设计方法 ……117
4.2.1 传统（结构化）的详细设计方法 …117
4.2.2 面向对象的详细设计 ……121
4.3 详细设计过程 ……123
4.4 详细设计规格文档 ……124
4.5 案例说明 ……125
4.6 小结 ……150
4.7 练习题 ……150
第5章 软件项目的编码 ……151
5.1 编码概述 ……151
5.2 编码方法 ……151
5.2.1 传统（结构化）编程方法 ……151
5.2.2 面向对象编程（OOP）……155
5.2.3 编码指南 ……156
5.3 编码标准和规范 ……156
5.4 重用原则 ……162
5.5 关于重构理念 ……163
5.6 编码过程 ……163
5.7 编码文档 ……164
5.8 案例说明 ……165
5.9 小结 ……190
5.10 练习题 ……190
第6章 软件项目的测试 ……191
6.1 软件测试概述 ……191
6.2 软件测试方法概论 ……192
6.3 静态测试 ……193
6.4 动态测试 ……193
6.4.1 白盒测试方法 ……193
6.4.2 黑盒测试方法 ……199
6.4.3 灰盒测试方法 ……206
6.5 软件测试级别 ……206
6.5.1 单元测试 ……208
6.5.2 集成测试 ……210
6.5.3 系统测试 ……213
6.5.4 接收测试 ……215
6.5.5 回归测试 ……216
6.6 面向对象的测试 ……216
6.6.1 面向对象中的单元测试策略 ……216
6.6.2 面向对象中的集成测试策略 ……217
6.7 测试管理过程 ……218
6.7.1 软件测试计划 ……218
6.7.2 软件测试设计 ……219
6.7.3 软件测试开发 ……220
6.7.4 软件测试执行 ……226
6.7.5 软件测试跟踪 ……227
6.7.6 软件测试评估与总结 ……227
6.8 自动化测试 ……227
6.8.1 测试自动化的程度 ……228
6.8.2 测试工具的使用现状及分析 ……228
6.9 软件测试过程 ……230
6.9.1 单元测试过程 ……230
6.9.2 集成测试过程 ……230
6.9.3 系统测试过程 ……231
6.9.4 接收测试过程 ……232
6.10 软件测试过程的文档 ……233
6.10.1 测试计划文档 ……233
6.10.2 测试设计 ……234
6.10.3 测试开发 ……235
6.10.4 测试执行 ……235
6.10.5 测试跟踪 ……235
6.10.6 测试总结 ……236
6.11 案例说明 ……237
6.12 小结 ……281
6.13 练习题 ……282
第7章 软件项目的提交 ……283
7.1 软件项目提交 ……283
7.2 培训 ……283
7.2.1 培训对象 ……283
7.2.2 培训方式 ……284
7.2.3 培训指南 ……285
7.3 用户文档 ……285
7.3.1 用户手册 ……285
7.3.2 系统管理员手册 ……285
7.3.3 其他文档 ……285
7.4 软件项目的提交过程 ……286

7.5 软件项目提交文档 ……286
7.5.1 用户手册文档标准 ……287
7.5.2 系统管理员手册文档标准 ……288
7.5.3 产品提交文档标准 ……289
7.6 案例说明 ……291
7.7 小结 ……310
7.8 练习题 ……310
第8章 软件项目的维护 ……311
8.1 软件项目维护概述 ……311
8.2 软件项目维护的任务 ……311
8.2.1 纠错性维护 ……312
8.2.2 适应性维护 ……312
8.2.3 完善性维护 ……312
8.2.4 预防性维护 ……312
8.3 软件项目维护过程 ……312
8.4 软件维护过程文档 ……314
8.5 案例说明 ……315
8.6 小结 ……315
8.7 练习题 ……315
附录 软件工程知识体系的10个知识领域简介 ……316
参考文献 ……322

第 1 章

■ 软件工程的实质

软件（software）是计算机系统中与硬件（hardware）相互依存的另一部分，它包括程序（program）、相关数据（data）及其说明文档（document）。其中程序是按照事先设计的功能和性能要求执行的指令序列；数据是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有关的各种图文资料。

软件工程（Software Engineering，简称为SE）是针对软件这一具有特殊性质的产品的工程化方法。软件工程涵盖了软件生存周期的所有阶段，并提供了一整套工程化的方法，来指导软件人员的工作。

1.1 软件工程引发的故事

本世纪中期，软件产业从零开始起步，在短短的50年的时间里迅速发展成为推动人类社会发展的龙头产业。随着信息产业的发展，软件对人类社会性越来越重要。软件发展的50年历史中，人们对软件的认识经历了一个由浅到深的过程。

第一个写软件的人是Ada（Augusta Ada Lovelace），在19世纪60年代他尝试为Babbage（Charles Babbage）的机械式计算机写软件。尽管失败了，但他将永远载入了计算机发展的史册。20世纪50年代，软件伴随着第一台电子计算机的问世诞生了。以写软件为职业的人也开始出现，他们多是经过训练的数学家和电子工程师。20世纪60年代美国大学里开始出现授予计算机专业的学位，设置相关课程教人们写软件。

软件发展的历史可以大致分为如下的3个阶段：

第一个阶段是20世纪50年代到20世纪60年代，是程序设计阶段，基本是个体手工劳动的生产方式。这个时期，一个程序是为一个特定的目的而编制的，软件的通用性很有限。软件往往带有强烈的个人色彩。早期的软件开发也没有什么系统的方法可以遵循，软件设计是在某个人的头脑中完成的一个隐藏的过程。而且，除了源代码往往没有软件说明书等文档，因此这个时期尚无软件的概念，基本上只有程序、程序设计概念，不重视程序设计方法，设计的程序主要是用于科学计算，规模很小、采用简单的工具、基本上采用低级语言，硬件的存储容量小、运行可靠性差。

第二阶段是20世纪60年代到20世纪70年代，是软件设计阶段，小组合作生产方式。在这

一时期软件开始作为一种产品被广泛使用，出现了“软件作坊”专职应别人的需求写软件。这个阶段，基本采用高级语言开发工具，开始提出结构化方法。硬件的速度、容量、工作可靠性有明显提高，而且硬件的价格降低。人们开始使用产品软件（可购买），从而建立了软件的概念。程序员数量猛增，但是开发技术没有新的突破，软件开发的方法基本上仍然沿用早期的个体化软件开发方式，软件的数量急剧膨胀，软件需求日趋复杂，维护的难度越来越大，开发成本令人吃惊得高，开发人员的开发技术不适应规模大、结构复杂的软件开发，失败的项目越来越多。

第三个阶段是从20世纪70年代至今，为软件工程时代，采用的是工程化的生产方式。这个阶段的硬件向超高速、大容量、微型化以及网络化方向发展，第三、四代程序设计语言出现。数据库、开发工具、开发环境、网络、分布式、面向对象技术等工具方法都得到应用。软件开发技术有很大进步，但未能获得突破性进展，软件开发技术的进步一直未能满足发展的要求。一些复杂的、大型的软件开发项目提出来了，在那个时代，很多的软件最后都得到了一个悲惨的结局。很多软件项目开发时间大大超出了规划的时间表。同时软件开发人员也发现软件开发的难度越来越大，在软件开发中遇到的问题找不到解决的办法，使问题积累起来，形成了尖锐的矛盾，失败的软件开发项目屡见不鲜，因而导致了软件危机。

软件危机指的是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。概括来说，软件危机包含两方面问题：一、如何开发软件，以满足不断增长，日趋复杂的需求；二、如何维护数量不断膨胀的软件产品。落后的软件生产方式无法满足迅速增长的计算机软件需求，从而导致软件开发与维护过程中出现一系列严重问题的现象。

最为突出的例子是美国IBM公司于1963～1966年开发的IBM 360系列机的操作系统。该项目的负责人Fred Brooks（F. D. 希罗克斯）在总结该项目时无比沉痛地说：“……正像一只逃亡的野兽落到泥潭中做垂死挣扎，越是挣扎，陷得越深，最后无法逃脱灭顶的灾难，……程序设计工作正像这样一个泥潭……一批批程序员被迫在泥潭中拼命挣扎，……，谁也没有料到问题竟会陷入这样的困境……”IBM 360操作系统的历史教训已成为软件开发项目中的典型事例被载入历史史册。

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。开发成本超出预算，项目经常延期，无法按时完成任务。

2）开发的软件不能满足用户要求。

3）软件产品的质量低。

4）开发的软件可维护性差。

5）软件通常没有适当的文档资料。

6）软件的成本不断提高。

7）软件开发生产率的提高赶不上硬件的发展和人们需求的增长。

软件危机的原因，一方面是与软件本身的特点有关；另一方面是由软件开发和维护的方法不正确引起的。由于软件危机的产生，迫使人们不得不研究、改变软件开发的技术手段和管理方法。从此软件生产进入软件工程时代。

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第一次提出了“软件危机”（software crisis）这个名词。同时，讨论和制定摆脱“软件危机”的对策。在那次会议上第一次提出了软件工程（software engineering）这个概念，从此一门新兴的工程学

科——软件工程学——为研究和克服软件危机应运而生。

“软件工程”的概念是为了有效地控制软件危机的发生而被提出来的，它的中心目标就是把软件作为一种物理的工业产品来开发，要求“采用工程化的原理与方法对软件进行计划、开发和维护”。软件工程是一门旨在开发满足用户需求、及时交付、不超过预算和无故障的软件的学科。软件工程的主要对象是大型软件。它的最终目的是摆脱手工生产软件的状况，逐步实现软件开发和维护的自动化。

我们要求工程目标能在一定的时间、一定的预算之内完成。软件工程是针对软件危机提出来的。从微观上看，软件危机的特征正是表现在完工日期一再拖后、经费一再超支，甚至工程最终宣告失败等方面。而从宏观上看，软件危机的实质是软件产品的供应赶不上需求的增长。

自从软件工程概念提出以来，经过几十年的研究与实践，虽然“软件危机”没得到彻底解决，但在软件开发方法和技术方面已经有了很大的进步。尤其应该指出的是，自20世纪80年代中期，美国工业界和政府部门开始认识到，在软件开发中，最关键的问题是软件开发组织不能很好地定义和管理其软件过程，从而使一些好的开发方法和技术都起不到所期望的作用。也就是说，在没有很好定义和管理软件过程的软件开发中，开发组织不可能在好的软件方法和工具中获益。

1.2 软件工程知识体系

“工程”是科学和数学的某种应用，通过这一应用，使自然界的物质和能源的特性能够通过各种结构、机器、产品、系统和过程，成为对人类有用的东西。因而，“软件工程”就是科学和数学的某种应用，通过这一应用，使计算机设备的能力借助于计算机程序、过程和有关文档成为对人类有用的东西。

软件工程的成果是为软件设计和开发人员提供思想方法和工具，而软件开发是一项需要良好组织，严密管理且各方面人员配合协作的复杂工作。软件工程正是指导这项工程的一门科学。软件工程在过去一段时间内已经取得了长足的进展，可以说在软件的开发和应用中起到了其应有的作用。

随着软件开发的深入、各种技术的不断创新以及软件产业的形成，人们越来越意识到软件过程管理的重要性。经过多年的开发实践和技术交流，不断在对传统的软件工程理论进行反思。

高质量的软件工程可以保证软件工业中生产的软件是高质量的产品、用户满意的产品。但是，对软件工程的界定，总是存在一定的差异。这里我们引用 IEEE在软件工程知识体系指南（SWEBOK：Guide to the Software Engineering Body of Knowledge 2004 Version）中的定义，它是这样定义软件工程的：1）软件开发、实施、维护的系统化、规范化、质量化的方法的应用，也就是软件的应用工程；2）对上述方法的研究。

IEEE的软件工程知识体系指南（SWEBOK）中界定了软件工程的11个知识领域（KAs：Knowledge Areas）即软件需求（Software requirements）、软件设计（Software design）、软件构建（Software construction）、软件测试（Software testing）、软件维护（Software maintenance）、软件配置管理（Software configuration management）、软件工程管理（Software engineering management）、软件工程过程（Software engineering process）、软件工程工具和方法（Software engineering tools and methods）、软件质量（Software quality）和相

关学科知识领域（Knowledge Areas of the Related Disciplines）。这10个知识领域的每个知识领域还包括很多子领域，如图1-1和图1-2所示。其中：

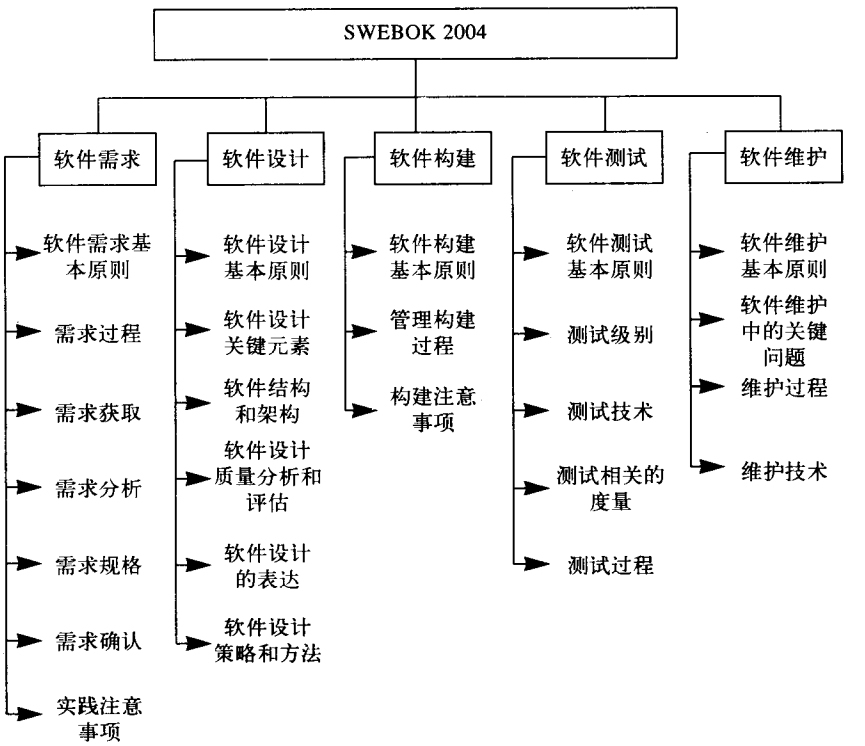

图1-1 SWEBOK的前5个知识域

软件需求知识域包括7个子领域：软件需求的基本原则（Software Requirements Fundamentals）、需求过程（Requirements Process）、需求获取（Requirements Elicitation）、需求分析（Requirements Analysis）、需求规格（Requirements Specification）、需求确认（Requirements Validation）、实践注意事项（Practical Considerations）。

软件设计知识域包括6个子域，即软件设计基本原则（Software Design Fundamentals）、软件设计关键元素（Key Issues in Software Design）、软件结构和架构（Software Structure and Architecture）、软件设计质量分析和评估（software Design Quality Analysis and Evaluation）、软件设计的表达（Software Design Notations）、软件设计策略和方法（Software Design Strategies and Methods）。

软件构建包括3个子领域，即软件构建基本原则（Software Construction Fundamentals）、管理构建过程（Managing Construction）、构建注意事项（Practical Considerations）。

软件测试包括5个子域即软件测试基本原则（Software Testing Fundamentals）、测试级别（Test Levels）、测试技术（Test Techniques）、测试相关的度量（Test-Related Measures）和测试过程（Test Process）。

软件维护（Software maintenance）是在产品提交之后就开始了，而且维护活动可能会更早些，一旦在操作中有不规则的事情发生、操作环境发生变化、用户界面发生新变化等都有

可能导致维护活动。它包括4个子域，即软件维护基本原则（Software Maintenance Fundamentals）、软件维护中的关键问题（Key Issues in Software Maintenance）、维护过程（Maintenance Process）和维护技术（Techniques for Maintenance）。

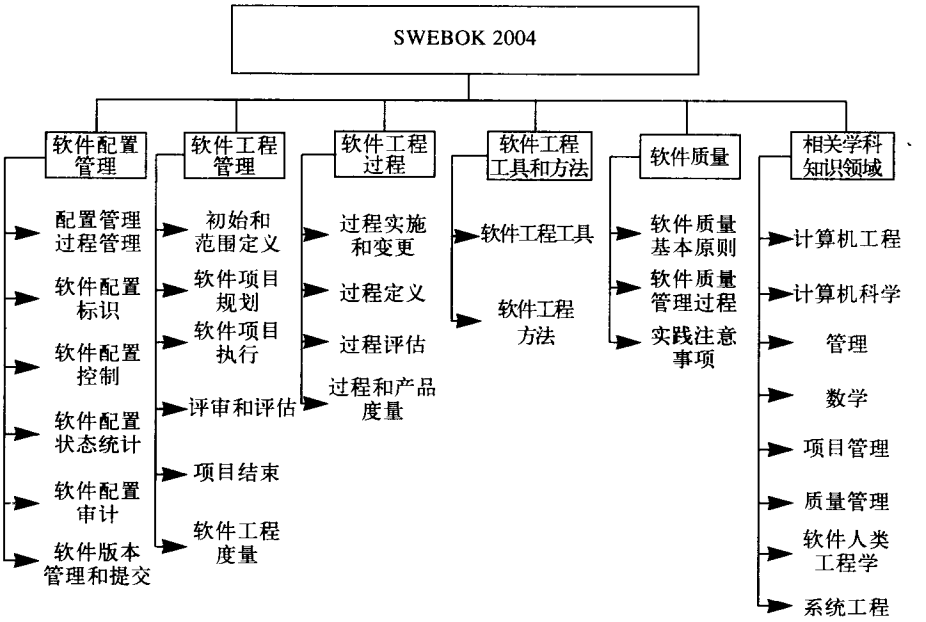

图1-2 SWEBOK后6个知识域

软件配置管理包括6个子域，即配置管理过程管理（Management of the SCM Process）、软件配置标识（Software Configuration Identification）、软件配置控制（Software Configuration Control）、软件配置状态统计（Software Configuration Status Accounting）、软件配置审计（Software Configuration Auditing）和软件版本管理和提交（Software Release Management and Delivery）。

软件工程管理包括6个子域，即初始和范围定义（Initiation and Scope Definition）、软件项目规划（Software Project Planning）、软件项目执行（Software Project Enactment）、评审和评估（Review and Evaluation）、项目结束（Closure）和软件工程度量（Software Engineering Measurement）。

软件工程过程包括4个子域，即过程实施和变更（Process Implementation and Change）、过程定义（Process Definition）、过程评估（Process Assessment）和过程和产品度量（Process and Product Measurements）。

软件工程工具和方法包括2个子域，即软件工程工具（Software Engineering Tools）和软件工程方法（Software Engineering Methods）。

软件质量包括3个子域，即软件质量基本规则（Software Quality Fundamentals）、软件质量管理过程（Software Quality Management Processes）和实践注意事项（Practical Considerations）。

相关学科知识领域包括8个相关学科，即计算机工程（Computer Engineering）、计算机科学（Computer Science）、管理（Management）、数学（Mathematics）、项目管理（Project

Management)、质量管理（Quality Management)、软件人类工程学（Software Ergonomics)、系统工程（Systems Engineering)。

1.3 软件工程的三段论

软件工程是为克服软件危机而提出的一种概念，并在实践中不断地探索它的原理，技术和方法。在此过程中，人们研究和借鉴了工程学的某些原理和方法，并形成了软件工程学。软件工程的目标是提高软件的质量与生产率，最终实现软件的工业化生产。既然软件工程是“工程”，那么我们从工程的角度看一下软件项目的实施过程，如图1-3所示。

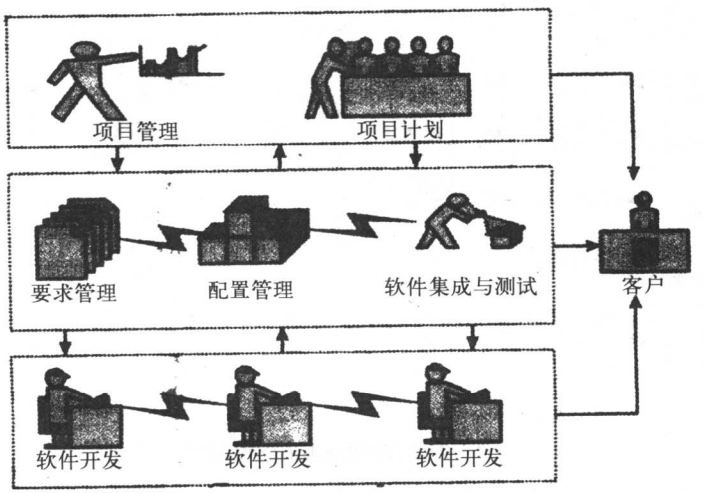

图1-3 工程化软件开发

客户的需求启动了一个软件项目，为此我们需要先规划这个项目，即完成项目计划，然后根据这个项目计划实施项目。项目实施的依据是需求，这个需求类似工程项目的图纸，开发人员按照这个图纸生产软件即设计、编码，在开发生产线上，将开发过程的半成品，通过配置管理存储和管理，然后进行必要的集成和测试，直到最后提交给客户。在整个开发过程中需要进行项目跟踪管理。软件工程活动是“生产一个最终满足需求且达到工程目标的软件产品所需要的步骤”。这些活动主要包括开发类活动、管理类活动和过程类活动，这里将它定义为“软件工程的三段论”，或者“软件工程的三线索”。一段论是“软件项目管理论”，二段论是“软件项目开发论”，三段论是“软件过程改进论”。这个三段论可以用一个三角形表示，如图1-4所示，它们类似于相互支撑的三角形的三个边。我们知道三角形是最稳定的，要保证三角形的稳定性，三角形的三个边必不可少，而且要保持一定的相互关系。

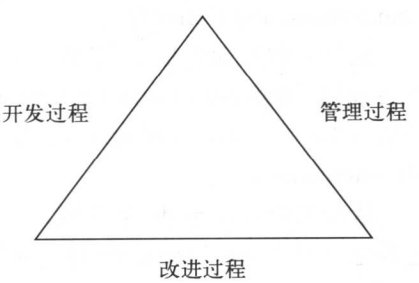

图1-4 软件工程的三个线索

其中：开发过程是软件人员生产软件的过程，例如需求分析、设计、编码、测试等，相当于生产线上的生产过程。

管理过程是项目管理者规划软件开发、控制软件开发的过程，相当于生产线上的管理过程，管理过程是伴随开发过程进行的过程。

过程改进相当于对软件开发过程和软件管理过程的“工艺流程”进行管理和改进，如果没有好的工艺生产不出好的产品，它包括对开发过程和管理过程的定义和改进。

为了保证软件管理、软件开发过程的有效性，应该保证这些过程的高质量和过程的持续改进。

传统工业中知名的生产方式可算是“手工作坊式”生产方式。过去的一段时间里，甚至到现在，中国软件业中还有一部分公司的开发方式与之类似。公正地说，以此方式还是成就了许多成功的应用开发项目，甚至可以说此法支撑起了软件开发的初期事业。但是，在我们的“作坊”里还有太多的项目失败，例如开发周期的不可控制、项目结果为用户所不认可、项目最终的严重亏损，这种失败的惨痛出乎我们的意料，以至于我们无所适从。再加上技术人员、资金严重匮乏的困扰，软件开发管理举步维艰。

作坊式的软件企业中，很多方法、规则都是装在开发人员的脑子里面的，往往会因为一两个开发骨干走了，就造成整个公司的瘫痪。赌注完全押在这一两个人的身上，资本投入风险很大，如果研发骨干另谋高就，公司投资将付之流水，作坊式的运作模式严重阻障了软件企业的成长。

让软件工程成为真正的工程，就需要软件项目的开发、管理、过程等方面规范化、工程化、工艺化、机械化。

软件开发过程中脑力活动的“不可见性”大大增加了过程管理上的困难。因此软件工程管理中的一项指导思想就是千方百计地使这些过程变为“可见的”以及事后可以检查的记录。只有从一开始就在开发过程中严格贯彻质量管理，软件产品的质量才有保证。否则，开发工作一旦进行到后期，无论怎样测试和补漏洞，都会无济于事。

1.3.1 软件开发过程

开发过程是软件工程的核心过程，通过这个生产线可以生产出用户满意的产品，软件工程提供了一整套工程化的方法，来指导软件人员的工作。

如同传统工程的生产线上有很多工序，每道工序都有明确的规程，软件生产线上的工序主要包括：需求、概要设计、详细设计、编码、测试、提交、维护等。采用一定的流程将各个环节连接起来，并可用规范的方式操作全过程，如同工厂的生产线。这样就形成了生存期模型（见第1.4节）。

软件开发过程是随着开发技术的演化而随之改进的。从早期的瀑布式（Waterfall）的开发模型到后来出现的螺旋式的迭代（Spiral）开发，以致最近开始兴起的敏捷开发方法（Agile），它们展示出了在不同的时代软件产业对于开发过程的不同的认识以及对于不同类型项目的理解方法。

图1-5是软件开发的路线图，这个路线图展示了软件开发的基本工艺流程，从需求开始，需求是项目开发的基础，概要设计为软件需求提供了实施方案，详细设计是对概要设计的细化，是为编码提供依据，编码是软件的具体实现，测试是验证这个软件的正确性，提交（发布）是将软件提交给使用者，在软件的使用过程中有问题需要维护。

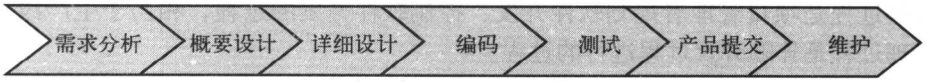

图1-5 软件开发过程路线图

没有规则的软件开发过程带来的只可能是无法预料的结果，这是我们在经历了一次次的项目失败之后，逐渐领悟到的道理。随着软件项目的规模不断加大，参与人员的增多，对规范性的要求愈加严格，基于软件项目管理的、工程化的软件开发时代已经来临。

1.3.2 软件项目管理

在我们的生活和社会中，项目是无处不在的，小到日常生活的商品采购，大到航天飞行计划等都是项目。美国项目管理专家James P.lewis说：项目是一次性、多任务的工作，具有明确规定的开始和结束日期、特定的工作范围、预算和要达到的特定性能水平。

项目经理首先必须弄明白什么是项目，项目涉及4个要素：预期的绩效、费用（成本），时间进度，工作范围。这4个要素相互关联、相互影响。

例如你去进行一次商品采购，原来想好要采购很多东西，回来却发现忘记买了很多东西，所以，你决定再出去采购的时候，你会在一张纸上记录所有需要购买的东西，即采购清单，这个采购清单帮助你不会遗漏采购项，你可以采用“完成一个采购项，在采购清单上打一个勾”的方法协助你完成采购，如果清单每项都打勾了，就表示所有的采购任务完成了。这里的“采购清单”便是计划在商品采购过程中通过“打勾”的方式控制这个计划的执行。

类似地，软件项目管理也是这样，软件项目管理就是如何管理好软件项目的范围、时间、成本，也就是管理好项目的内容、花费的时间（进度）以及花费的代价（规模成本），其他相关的事情都是围绕这三件事情进行的。为此需要制定一个好的项目计划，然后控制好这个计划，即软件项目管理的实质是软件项目计划的编制和软件项目计划的跟踪控制如图1-6所示。它们是相辅相成的关系：计划是项目成功实施的指南和跟踪控制的依据；而跟踪控制又保证项目计划的成功执行。

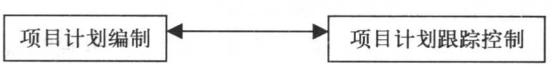

图1-6 软件项目管理的实质

实际上，要做到项目计划的切合实际是一个非常高的要求，需要对项目的需求进行详细的分析，根据项目的实际规模制定合理的计划，计划的内容包括进度安排、资源调配、经费使用等，为了降低风险，要进行必要的风险分析与制定风险管理计划。同时要对自己的开发能力有非常准确的了解，制定切实可行的质量计划和配置管理计划等。这来源于项目经理的职业技能和实践经验的持续积累。

制定了合适的项目计划之后，才能进行有效的跟踪与监督，项目计划是根本的依据。当发现项目计划的实际执行情况与计划不符的时候要进行适当的、及时的调整，确保项目按期、按预算、高质量地完成。项目成功与否的关键是能不能成功地实施项目管理。图1-7便是项目管理的路线图。

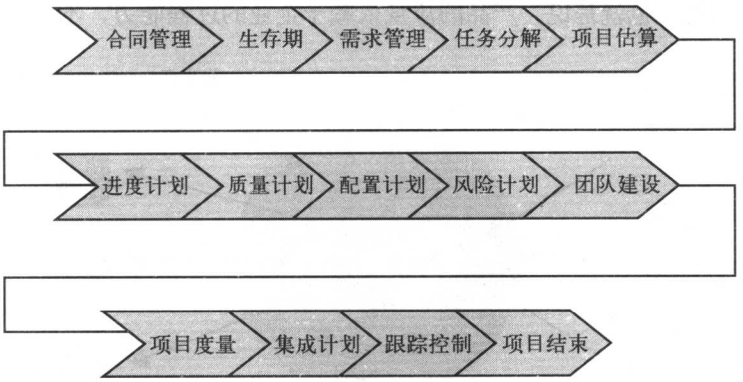

图1-7 软件项目管理路线图

1.3.3 软件过程改进

自从20世纪70年代软件危机以来，人们不断地展开新方法和新技术的研究与应用，但仍未取得突破性的进展。直到20世纪80年代末，人们得出这样一个总结：一个软件组织的软件能力取决于该组织的过程能力。如果一个软件组织的过程能力越成熟，那么该组织的软件生产能力就越有保证。

所谓过程，简单来说就是我们做事情的一种固有的方式，我们做任何事情都有过程存在，小到日常生活中的琐事，大到工程项目。对于做一件事，有过经验的人对完成这件事的过程会很了解，他会知道完成这件事需要经历几个步骤，每个步骤都完成什么事，需要什么样的资源、什么样的技术等等，因而可以顺利地完成工作；没有经验的人对过程不了解，就会有无从着手的感觉。下面两个图可以形象地说明过程在软件开发中的地位。如果项目人员将关注点只放在最终的产品上，如图1-8所示，不关注期间的开发过程，那么不同的开发队伍或者个人可能就会采用不同的开发过程，结果导致开发的产品质量是不同的，有的质量高，有的质量差，这完全依赖个人的素质和能力。

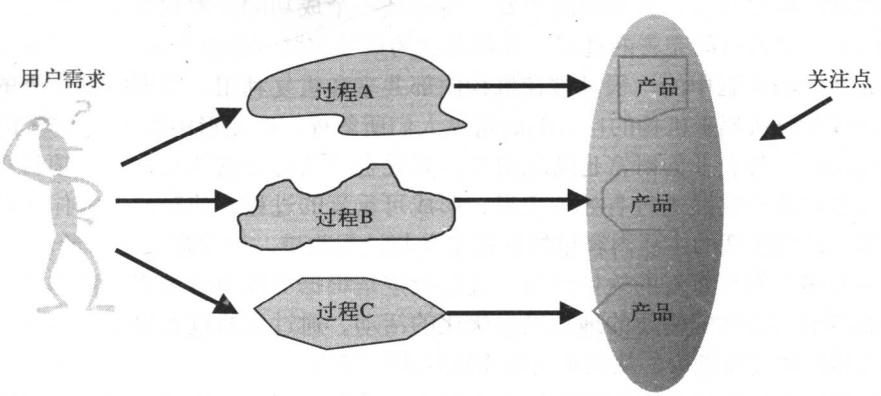

图1-8 关注开发的结果

反之，如果将项目的关注点放在项目的开发过程，见图1-9，不管谁来做，都采用统一的开发过程，也就是说，企业的关注点在过程。经过同一企业过程开发的软件，产品的质量是一样的。可以通过不断提高过程的质量，来提高产品的质量，这个过程是公司能力的体现，它是

不依赖于个人能力的。也就是说，产品的质量依赖于企业的过程能力，不依赖于个人能力。

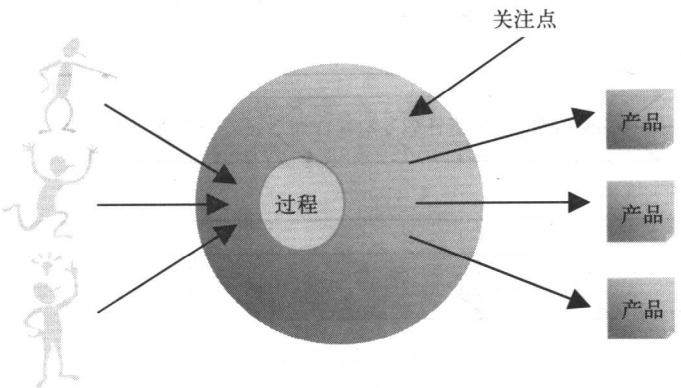

图1-9 关注开发的过程

对于软件过程的理解，绝对不能简单地理解为软件产品的开发流程，因为我们要管理的并不只是软件产品开发的活动序列，而是软件开发的最佳实践。它包括：流程、技术、产品、活动间关系、角色、工具等，是软件开发过程中的各个方面的因素的有机结合。因此，在软件过程管理中，首先要进行过程定义，将过程以一种合理的方式描述出来，并建立起企业内部的过程库，使过程成为企业内部可以被重用的共享资源。对于过程，要不断地进行改进，以不断地改善和规范过程，帮助提高企业的生产力。

软件过程是极其复杂的过程。我们知道，软件是由需求驱动的，有了用户应用的实际需求才会引发开发一个软件产品的活动。软件产品从需求的出现直到最终的产品出现，要经历一个复杂的开发过程，软件产品在使用时要根据需求的变更进行不断的修改，这称为软件维护。我们把用于从事软件开发及维护的全部技术、方法、活动、工具以及它们之间的相互变换统称为软件过程。由此可见，软件过程的外延非常之大，包含的内容非常之多。对于一个软件开发机构来说，做过一个软件项目，无论成功与否，都能够或多或少地从中总结出一些经验。做过的项目越多，其经验越丰富，特别是一个成功的开发项目是很值得总结的，从中可以总结出一些做事的完善的过程，我们称之为最佳实践（Best Practices）。最佳实践开始是存放在成功者的头脑中的，很难被在机构内部共享和重复利用，以发挥其应有的效能。长期以来，这些本应从属于机构的巨大的财富被人们所忽视，这无形中给机构带来了巨大的损失，当人员流动时这种企业的财富也随之流失，并且也使这种财富无法被其他项目再利用。过程管理，就是对最佳实践进行有效的积累，形成可重复的过程，使我们的最佳实践可以在机构内部共享。过程管理的主要内容包括过程定义与过程改进。过程定义是对最佳实践加以总结，以形成一套稳定的可重复的软件过程。过程改进是根据实践中对过程的使用情况，对过程中有偏差或不够切合实际需要的地方进行优化的活动。通过实施过程管理，软件开发机构可以逐步提高其软件过程能力，从根本上提高软件生产能力。

有效的软件过程可以提高组织的生产能力，理解软件开发的基本原则，可以帮我们作出明智的决定；可以标准化你的工作，提高软件的可重用性和团队间的协作；有效的软件过程可以改善我们对软件的维护。

美国卡内基－梅隆大学软件工程研究所（CMU/SEI）主持研究与开发的CMM/PSP/TSP 技术，为软件工程管理开辟了一条新的途经。PSP、TSP和CMM为软件产业提供了一个集成化