HZ BOOKS
华章教育

高等院校计算机课程案例教程系列

窦万峰 主编
杨坤 许敏 缪静娴 钱辰 等参编

软件工程
方法与实践

第2版

Software Engineering
Theory and Practice (Second Edition)

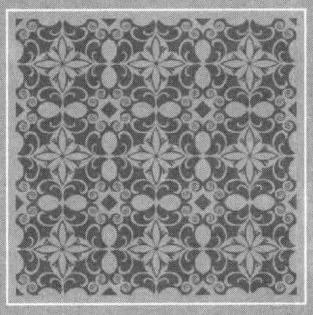
机械工业出版社
China Machine Press

高等院校计算机课程案例教程系列

窦万峰 主编
杨坤 许敏 缪静娴 钱辰 等参编

软件工程方法与实践

第2版



Software Engineering
Theory and Practice (Second Edition)

机械工业出版社
China Machine Press

图书在版编目（CIP）数据

软件工程方法与实践／窦万峰主编. —2 版. —北京：机械工业出版社，2013.1
（高等院校计算机课程案例教程系列）

ISBN 978-7-111-40696-9

Ⅰ. 软… Ⅱ. 窦… Ⅲ. 软件工程—高等学校—教材 Ⅳ. TP311.5

中国版本图书馆 CIP 数据核字（2012）第 305185 号

本书从传统的结构化软件开发范型和面向对象软件开发范型两个方面，把软件工程的理论与方法融入开发实践当中，通过丰富的案例分析与设计，深入地介绍软件开发中各个阶段的技术、方法和管理过程。本书介绍了软件工程的基本概念、软件过程和软件过程模型，结构化分析与设计和面向对象分析与设计的基本概念、分析与设计过程、分析与设计模型、相关技术与方法，以及软件测试原理与技术、维护策略与方法、软件项目管理等内容，是一本全面介绍软件开发的“工程化”思想的理想教材。

本书适合作为高等院校软件工程课程的教材，既适用于计算机专业的学生，也适用于其他非计算机专业的学生以及从事软件开发、应用和管理的技术人员，同时也适合专业软件开发人员参考。

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：刘立卿
北京市荣盛彩色印刷有限公司印刷
2013 年 7 月第 2 版第 1 次印刷
185mm×260mm · 20 印张
标准书号：ISBN 978-7-111-40696-9
定 价：39.00 元

凡购本书，如有缺页、倒页、脱页，由本社发行部调换
客服热线：（010）88378991 88361066 投稿热线：（010）88379604
购书热线：（010）68326294 88379649 68995259 读者信箱：hzjsj@hzbook.com

前　言

软件工程包含了一系列原理、方法和实践，它将计算机科学理论与现代工程方法论相结合，着重研究软件过程及其模型、分析与设计方法、软件工程开发与管理技术，是指导人们进行正确的软件生产和管理的一门新兴的、综合性的应用科学。软件开发强调从工程化的原理出发，按照标准化规程和软件开发实践来引导软件开发人员进行软件开发，并进行过程改进，促进软件企业向标准化和成熟化发展。软件工程是一门理论与实践相结合的学科，注重通过实践来理解理论、原理与方法。为此，本书结合多年的软件工程教学和项目开发经验，通过项目实例，从不同的角度和范型循序渐进地介绍软件开发过程中所涉及的原理、方法与技术。

全书分为四个部分：

第一部分：软件工程基础。这一部分共安排了3章内容，初步介绍软件工程的基本概念、软件过程和软件过程模型等。

第二部分：结构化分析与设计。这一部分共安排了5章内容，介绍结构化分析与设计的基本概念、分析与设计过程、分析与设计模型，以及相关技术与方法，包括需求分析、结构化分析方法、结构化设计概念、结构化概要设计和结构化详细设计的内容。本部分用案例进一步深化结构化分析与设计的原理、方法及过程。

第三部分：面向对象分析与设计。这一部分安排了3章内容，介绍面向对象分析与设计的基本概念、用例分析模型与设计过程、面向对象分析与设计模型，以及相关技术与方法，包括用例需求描述、面向对象分析方法和面向对象设计等内容。

第四部分：软件测试、维护与管理。这一部分安排了3章内容，主要介绍软件测试原理与技术、维护策略与方法、软件项目管理概念与原理、软件成本估算、项目计划与管理，包括软件测试策略、测试基本技术、测试用例设计、软件维护类型、结构化维护过程、项目管理过程和任务、项目管理的范围、软件项目成本估算方法、项目计划安排、软件配置管理、风险管理等内容。

本书的编写思想

- 对比介绍传统的结构化开发方法和面向对象开发方法，有利于学生理解不同的软件开发范型的特点和适用情况，适合作为教材使用。
- 注重软件分析与设计的思想介绍，通过案例分析让学生理解这些思想和原理。
- 加强了研究性教学方法。本书以敏捷开发实践“结对编程”为内容，让学生学习结对编程的原理，并通过上机结对实验活动来理解结对的过程，进而提出开发结对编程系统的需求。学期项目也可以完成结对编程的设计与开发及测试。

本书的结构体系和内容特点

- 分为软件工程原理、结构化分析与设计、面向对象分析与设计、软件测试与维护和管理四个方面。
- 针对每一个方面分别介绍软件分析与设计的过程、原理、方法和案例。
- 案例分析注重解析与思考原理和实现过程。

内容特点

- 将结构化软件分析与设计、面向对象分析与设计分开来讲，并选择较为适合的案例来帮助理解两个范型的特点和方法，避免学生产生概念混淆和方法上的混乱。
- 结构化软件分析与设计部分，介绍结构化开发的过程、原理、思想，这些都可以推广到面向对象的分析与设计过程中。通过一个需求稳定的案例介绍传统结构化开发的基本方法和做法。
- 针对面向对象软件分析与设计，注重面向对象分析模型和设计模型的构建，强调它们的直接关系，抓住面向对象模型开发的要点，通过 UML 来描述分析和设计过程，通过复杂的系统案例分析来理解面向对象分析与设计的思想。
- 注重本科生研究性教学实践，在内容上安排了敏捷实践的结对编程的内容，让学生理解结对编程思想的同时，分析存在的问题和解决方法，结合结对编程系统地分析该软件系统的需求，并进行设计、实现及测试。从研究性的角度提出问题和解决问题，从而达到研究性教学的目的。
- 全书先结构化后面向对象的讲解路线，便于学生逐步接受软件工程的开发思想，尤其适合没有任何开发概念的学生（未接触过软件工程概念的学生）。

本书的案例

本书介绍传统的结构化范型和面向对象范型两大体系，因此会选择适合不同范型中具有代表性的案例进行讲解，以便读者能够深入理解这两大范型各自的优势。本书的案例既有简单常见的应用系统，如面对面结对编程系统和 ATM 系统；也有比较实用的系统，如 POS 机系统；还有一些稍微复杂的系统，如分布式结对编程系统。本书之所以选择这些案例进行讲解，是因为许多人都熟悉这些问题，并且其中蕴涵的分析和设计问题具有普遍性。这些系统由简单到复杂，循序渐进，引导读者逐步理解系统的开发过程和关键问题。

面对面结对编程系统是一个辅助实现学生在实验室或公司进行结对编程和学习的系统，它克服了面对面结对编程存在的一些不便，如交换位置、相互的干扰等，支持角色交换、信息统计和相容性分析等功能。同时，该系统采用一台主机支持结对，还具有节约实验室建设费用等优点。

POS 机系统是电子收款机系统的简称，它通过计算机处理销售和支付，记录销售信息。该系统包括计算机终端、条码扫描仪、现金抽屉、票据打印机等硬件，使系统运转的软件，以及为不同服务的应用程序提供的接口。收银员通过条码扫描仪读取（或键盘输入）商品的条码号来记录商品信息，系统自动计算销售的总价。收银员能够通过系统处理支付，包括现金支付、信用卡支付和支票支付。经理通过系统能处理顾客退货。

ATM 系统即自动柜员机系统，能够自动处理银行储户的各种业务，如取款、存款、转账、查询、修改密码等。开发一个 ATM 软件系统，使客户能够直接访问银行计算机，完成交易而无需银行工作人员的介入。

分布式结对编程支持跨地域进行结对编程或学习。为了支持异地结对者能够像本地面对面结对那样自然地工作，系统通过文本、音频和视频进行交流。系统与集成开发环境进行集成，与源代码编辑器集成，包括 VC++、Eclipse 等。系统支持角色交换。结对编程的程序员通常不

严格遵循“驾驭者”和“导航员”的角色，所以分布式结对编程的工具应该允许合作者很容易地访问控制键盘。

本书内容翔实，并提供多个较为完整的案例，便于读者学习和深入体会软件分析与设计的原理与方法。不同的案例充分体现了不同的技术，因此有不同的案例介绍和分析层次，主要是突出方法的实用性。本书第 1～3 章由窦万峰和李东振编写，第 4～6 章由窦万峰和史玉梅编写，第 7～8 章由窦万峰和钱辰编写，第 9～10 章由窦万峰和许敏编写，第 11 章由杨坤编写，第 12～14 章由窦万峰和缪静娴编写。全书由窦万峰统稿、校对。

本书适合作为高等院校软件工程和软件分析与设计课程的教材，既适用于计算机专业的学生，也适用于其他非计算机专业的学生和从事软件开发与应用的技术人员。

由于作者水平有限，书中难免有疏漏之处，恳请各位读者指正。尤其是案例的详细程度和方案的多样性，请读者给予意见，以便以后改进和完善。

目　录

第一部分

软件工程基础

本部分介绍软件工程的基本概念和软件过程及其模型，分为软件工程概述、软件过程和软件过程模型等三章进行讲述。

本部分将回答以下问题：

- 什么是软件工程？
- 什么是工程化思想？
- 软件工程的基本原理和原则是什么？
- 什么是软件过程？
- 什么是软件过程模型？
- 常见的软件过程模型有哪些？

学习过本部分内容后，请思考下列问题：

- 如何选择软件过程模型？
- 为什么统一过程模型得到广泛流传？
- 敏捷过程有哪些优势？其模型对现代软件开发产生什么样的影响？
- 软件工程有哪些实践活动？
- 如何实施结对编程？

第 1 章

软件工程概述

软件工程（Software Engineering，SE）概念是在 20 世纪 60 年代末期提出的。这一概念的提出，目的是倡导以工程化的思想、原则和方法开发软件，并用来解决软件开发和维护过程中出现的诸多问题。

1.1 关于软件

既然软件工程中的主角是软件开发，那么在现代社会中，软件究竟担任一种什么样的角色呢？我们使用的大部分软件同时担任着两个角色，既是软件产品，又是软件工具。软件产品是指为最终用户使用并带来益处的具有商业价值的软件系统。软件工具是指开发其他软件的软件系统。我们可以利用这些软件系统存储信息，进行信息的变换等等。

什么是软件？软件是计算机系统中与硬件相对应的另一部分，是一系列程序、数据及其相关文档的集合。在这里，程序是按照特定顺序组织的计算机数据和指令的集合；数据是使程序能正常执行的数据结构；文档是与程序的开发、维护和使用有关的资料。计算机软件的核心是程序，而文档则是软件不可分割的组成部分。

1.1.1 软件的定义与特性

要理解软件的真正含义，需要先了解软件有哪些特征。在计算机体系结构中，硬件是有物理形态的；而软件是逻辑的、没有固有形态的，所以，计算机软件和硬件有着截然不同的特征。

1．复杂性

软件是一个庞大的逻辑系统，比人类构造的其他产品更复杂，甚至硬件的复杂性和软件比起来也是微不足道的。此外，软件主要依靠人脑的“智力”构造出来，多种人为因素使得软件难以统一化，因而更增加了其复杂性。软件的复杂性使得软件产品难以理解、难以生产、难以维护，更难以对生产过程进行管理。

2．一致性

软件必须和运行它的硬件保持一致，这是由软件对硬件的依赖所决定的，一般都采用软件顺应硬件接口而不是硬件顺应软件的方案。如果硬件系统是“现有”的，软件必须顺应现有的硬件系统接口。此外，由于计算机的软件和硬件是具有功能互换性的，所以也可能出现用软件来替代硬件接口的功能。

3．退化性

器械设备在其运行周期中的失效率大致遵循图 1-1 所示的 U 形曲线（浴盆曲线）。 而软件的情况与此截然不同，因为它不存在磨损和老化的问题。事实上，软件不会磨损，但它却会退化，因此，软件在其生命周期中需要进行多次维护，图 1-2 中的理想曲线是软件的实际故障模型的粗略简化。

4．易变性

软件在生产过程中，甚至在投入运行之后，都可以再改变。软件必须能够经历变化并容易改变，这也是软件产品的特有属性。软件易变性的好处是：改变软件往往可以收到改变或者完善系统功能的效果；软件具有易维护、易移植、易复用的特点，修改软件比更换硬件容易。修改软件的压力总是存在的，所以软件产品始终在“变”。这种动态的变化不仅难以预测、难以控制，而且可能对软件的质量产生负面影响。

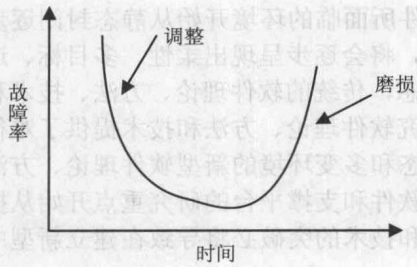


图 1-1　硬件的故障曲线

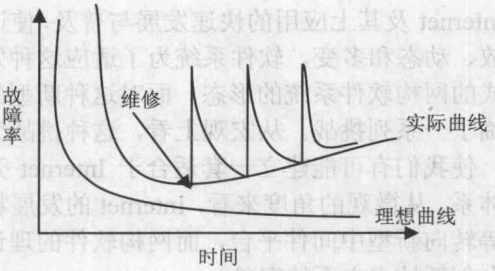


图 1-2　软件的理想故障曲线和实际故障曲线

5．移植性

软件的运行受计算机系统的影响，不同的计算机系统平台可能会导致软件无法正常运行，这里就牵扯到软件的可移植性了。好的软件在设计时就考虑到如何应用到不同的系统平台。

6．高成本性

软件开发是一个复杂的过程，涉及大量的人力成本和管理成本。所以，软件的成本非常高昂。

1.1.2　软件技术演化

按照系统论的观点，演化是事物从一种多样性统一形式转变成另一种多样性统一形式的具体过程。软件技术的发展也经历了一个演化的过程，自从 20 世纪 40 年代产生了世界上第一台计算机后，伴随而生的就是程序或软件。纵观前后的几十年间，软件技术的演化大致经历了四个阶段。

1．第一阶段

20 世纪 40 年代到 60 年代初，这是计算机软件发展的初期，一般称为程序设计阶段，其主要特征是程序生产方式为个体手工方式。

2．第二阶段

20 世纪 60 年代初到 70 年代中期，这是计算机软件发展的第二个阶段，我们称这个时期为程序系统阶段。在这个阶段，软件工程学科诞生了。程序的规模已经发展得很大了，需要多人分工协作，软件的开发方式由个体生产发展到了小组生产。但是，小组生产的开发方式基本上沿用了软件发展早期所形成的个体开发方式，开发与维护费用以惊人的速度增加。因此，许多软件产品后来根本不能维护，最终导致软件危机的出现。

3．第三阶段

20 世纪 70 年代中期至 80 年代中期，这是计算机软件发展的第三个阶段，一般称为软件工程阶段。在这个阶段，软件工程师把工程化的思想加入到软件的开发过程中，用工程化的原则、方法和标准来开发和维护软件。

4．第四阶段

20 世纪 80 年代中期至今，面向对象的方法学受到了人们的重视，促进了软件业的飞速发展，软件产业在世界经济中已经占有举足轻重的地位，这个阶段一般称为面向对象阶段。

1.1.3　软件发展趋势

20 世纪末开始流行的 Internet 给人们提供了一种全球范围的信息基础设施，形成了一个资源丰富的计算平台，未来如何在 Internet 平台上进一步整合资源，形成巨型的、高效的、可信的虚拟环境，使所有资源能够高效、可信地为所有用户服务，成为软件技术的研究热点。

Internet 平台具有一些传统软件平台不具备的特征：分布性、节点的高度自治性、开放性、异构性、不可预测性、连接环境的多样性等。这对软件工程的发展提出了新的问题。软件工程需要新的理论、方法、技术和平台来应对这个问题。目前投入很大精力研究的中间件技术就是这方面的典型代表。

Internet 及其上应用的快速发展与普及，使计算机软件所面临的环境开始从静态封闭逐步走向开放、动态和多变。软件系统为了适应这种发展趋势，将会逐步呈现出柔性、多目标、连续反应式的网构软件系统的形态。面对这种新型的软件形态，传统的软件理论、方法、技术和平台面临了一系列挑战。从宏观上看，这种挑战为我们研究软件理论、方法和技术提供了难得的机遇，使我们有可能建立一套适合于 Internet 开放、动态和多变环境的新型软件理论、方法和技术体系。从微观的角度来看，Internet 的发展将使系统软件和支撑平台的研究重点开始从操作系统等转向新型中间件平台，而网构软件的理论、方法和技术的突破必将导致在建立新型中间件平台创新技术方面的突破。

1.2　关于软件工程

早在软件开发的早期阶段，人们过高地估计了计算机软件的功能，认为软件能承担计算机的全部责任，甚至有些人曾误解为软件可以做任何事情。如今，绝大多数专业人士已经认识到软件神化思想的错误，但是，旧的观念和习惯并非轻易能丢弃的，仍有部分人认为软件可以神化。而软件危机的出现迫使人们开始思考软件并非是万能的，它不能满足人们越来越多的需要，进而提出需要好的开发与维护方法来指导人们高效率地开发软件。

1.2.1　软件危机

在软件技术发展的第二阶段，随着计算机硬件技术的不断进步，要求软件能与之相适应。然而软件技术的进步一直未能满足形势发展提出的要求，致使问题积累起来，形成了日益尖锐的矛盾，最终导致了软件危机。软件危机的主要表现如下：

1）软件的规模越来越大，复杂度不断增加，软件的需求量也日益增大，且价格昂贵，供需矛盾日益增大。

2）软件开发常常受挫，质量差，很难按照指定的进度表来完成指定的任务，软件的研制过程很难管理，即软件的研制往往失去控制。

3）软件开发的模式及技术已经不能适应软件发展的需要，因此导致大量低质量的软件涌向市场，部分软件花费了大量的人力财力，有的软件甚至在开发过程中就夭折了。例如，伦敦股票交易系统当初预算 4.5 亿英镑，后来追加到 7.5 亿，历时 5 年，但最终还是失败，导致伦敦股票市场声誉下跌。

下面通过伦敦救护服务系统的例子来分析软件危机的表现和问题。

伦敦救护服务系统覆盖伦敦市区 600 平方公里的地域和大约 680 万的救护人口，成为世界上最大的救护服务中心。该服务中心拥有 318 辆事故与应急救护车、445 辆病人运输救护车、一个摩托车接应团队和一架直升机。中心的工作人员达 2746 人，他们分布在伦敦市区 70 个救护站，每个救护站又分成 4 个运营部门。

伦敦救护服务系统的目的是提供自动化救护呼叫请求和处理紧急救护需要，通过计算机系统处理人工系统的所有任务。呼叫 999 和请求救护服务将呼叫者和派遣者连接起来，派遣者记录呼叫细节和分派合适的车辆，并选择救护车和转发救护信息给车载系统。

伦敦救护服务系统包括三个组成部分：计算机辅助派遣系统，包括软硬件基础设施、事故记录保存系统、无线电通信系统和无线电系统接口；计算机地图显示系统，包括复杂地域地形分析软件；自动化车辆定位系统，包括无线电系统和移动数据终端，它具有车辆自动定位能力，以便以最短的时间到达指定位置，并跟踪分析系统的性能。

伦敦救护服务系统项目于 1987 年 4 月启动，前期投资 250 万英镑开发一个有限功能的派遣系统。1989 年设计规格被重新修改，增加了移动数据终端和声讯转换系统。1990 年 10 月项目经过两次峰值负载性能测试失败而被迫终止。截至项目被取消，项目已经花费了 750 万英镑，超过预算 300%。

1991 年 8 月项目重新启动。为了保证项目的顺利进行，合作方定期举行会议来协调项目进度和解决存在的问题。但是到项目截止日期 1992 年 1 月，项目还是被延期。派遣系统没有完全实现、测试，无线电接口系统未能交付，救护车数据终端设计和定位系统需求还需进一步完善，车载定位跟踪系统没有完成安装调试。

1992 年 10 月 26 日，整个新系统全部运转。但是过载问题仍然没有很好地解决，存在呼叫丢失和响应不及时问题。1992 年 10 月 27 日，系统不得不改为半自动化方式。1992 年 11 月，系统运行性能开始全面下降，并最终导致系统锁死。由于没有及时响应和系统存在的故障，导致病人死亡事件发生。工作人员试图切换和重启系统，但均告失败。系统没有备份系统，操作人员被迫恢复到完全人工过程。

伦敦救护服务系统的失败是由于一系列软件工程中的错误，特别是项目管理中的缺陷，从而导致了 1992 年秋天出现的两次故障。伦敦救护服务系统失败的例子告诉我们，系统的复杂性和庞大，系统需求的不准确和经常变更，以及管理不到位等因素经常会导致系统的失败。

我们称软件开发和维护过程中所遇到的这种严重问题为软件危机。软件危机主要有两个方面的问题：

1）如何开发软件，以满足对软件功能的日益增长的客户需求。

2）如何维护数量不断膨胀的现有软件。

1.2.2　软件危机的解决途径

在软件危机日益严重的背景下，软件工程产生了。在引入工程化的思想后，人们总结了导致软件危机的原因：在软件开发的初期阶段，需求不够明确，或是未能得到确切的表达，开发工作开始后，软件开发人员和用户又未能及时交换意见，造成开发后期矛盾的集中暴露以至无法解决。由此看来，前期工作很重要。如果认为软件开发仅仅是编写程序，就可能对软件开发前期的需求分析不重视，从而使需求分析不到位，造成后期开发的软件达不到客户要求，导致软件的二次开发。

为此人们提出了以下解决对策：

1）需求分析可以进一步地了解客户的需求，对软件的开发很有帮助。

2）需求分析后，要做好软件定义时期的工作，这样可以在一定程度上降低软件开发的成本，同时又无形中提高了软件的质量。软件是一种商品，提高其质量是软件开发过程中的重中之重。

3）开发过程要有统一的、公认的方法论和规范指导，必须按照规定的方法论进行开发。由于软件是逻辑部件，开发阶段的质量难以衡量与评价，开发过程的管理和控制较难，所以要有统一的软件工程理论来指导开发工作。

4）必须在测试阶段充分做好检测工作，给客户提交高质量的软件。要借鉴软件开发的经验和利用积累的有关软件开发数据，确保开发工作按时完成，在期限内完成软件的开发。

1.2.3　软件工程的概念

软件工程化思想的核心是把软件看做是一个需要需求分析、设计、实现、测试、管理和维护的工程产品，用完善的工程化原理研究软件生产的规范方法，不仅保证软件开发在指定的期限内完成，还要节约成本，保证软件的质量。

B. W. Boehm 将软件工程定义为“运用现代科学技术知识来设计、构造计算机程序，以及为开发、运行和维护这些程序所必需的相关文件资料”。

Fritz Bauer 对软件工程的定义为：为经济地获得能够在实际机器上有效运行的可靠软件而建立和使用的一系列完善的工程化原则。

IEEE 软件工程标准术语对软件工程的定义为：软件工程是开发、运行、维护和修复软件的系统方法，其中“软件”指的是计算机程序、方法、规则、相关的文档资料以及程序在计算机上运行时所必需的数据。

尽管软件工程的具体定义不尽相同，一些学者又提出了更完善的定义，但其主要思想都是在强调在软件开发过程中需要应用工程化思想。

软件工程是一门研究如何用系统化、规范化、数量化等工程化思想和方法进行软件开发、维护和管理的学科。因此，软件工程学涉及的范围很广，它涉及计算机科学、管理学、系统工程学和经济学等多个学科领域。

软件工程学分成软件开发技术和软件工程管理两个方面，重点是对软件开发方法和工程性技术进行研究。软件开发技术和软件工程管理的复杂程度，均与软件的规模密切相关。规模越大的软件产品，越要严格遵守软件工程的开发原则和方法。

软件开发不同于一般的产品生产，因为软件是一种没有具体形体和尺寸的特殊产品，它提供的产品或服务是逻辑的，具有独特性、临时性和周期性等特点。不同于其他产品的制造，软件过程更多的是设计过程（没有制造过程）。另外，软件开发不需要使用大量的物质资源，而主要是人力资源。并且，软件开发的产品只是程序代码和技术文件，并没有其他的物质结果。基于上述特点，软件项目管理与其他项目管理相比，有很大的独特性。

软件开发中工程化的思想主要体现在软件项目管理。软件项目管理的作用一方面是提高质量，降低成本；另一方面则是为软件的工程化开发提供保障。与其他项目相比，软件项目还比较新，在软件行业的迅猛发展中，一些问题和危机逐渐暴露出来，如项目时间总是推迟、项目结果不能令客户满意、项目预算成倍超出、项目人员不断流动等等都是软件开发商不断面临的一些问题。

软件工程学家分析认为，导致上述情况的主要原因是缺乏软件过程控制能力，开发过程随心所欲，时间计划和费用估算缺乏现实的基础，管理者主要在应付突发事件，对产品质量缺乏客观基础，软件开发的成败建立在个人能力基础上等。

为了解决软件工程中各种各样的问题，美国软件工程研究所 SEI 自 1986 年开始研究软件过程成熟度框架，并于 1991 年提交了能力成熟度模型（Capability Maturity Model）CMM V1.0。历经多方软件专家的评审，美国软件工程研究所又发布了 V1.1 版，并更名为 SW-CMM。1999 年底发布了 V2.0 版。该模型强调企业软件开发能力取决于企业的过程能力而不是个人能力，强调持续的过程能力的改善是衡量软件企业的软件开发、管理水平的重要参考。该模型既可以作为软件开发组织改善软件开发过程的参考模型，也可以作为用户评估软件项目承包商的依据。

如今，软件开发的工程化管理思想已经得到了认可，软件的开发管理已经不同于以往过分依赖软件技术精英的情况。运用项目管理的经验和方法是软件项目成功的前提和保证，这已是今天软件业内人士的共识。随着信息技术的飞速发展，软件产品的规模也越来越庞大，个人单打独斗的作坊式开发方式已经越来越不适应发展的需要。尤其是近几年，随着网络技术的快速发展，项目管理也随之快速发展，各软件企业都在积极将软件项目管理引入开发活动中，对开发实行有效的管理，并取得了不同的成绩。

1.3　软件工程原理与原则

人们根据软件开发的特点和软件工程概念，提出了一些软件工程的基本原理和基本原则。

1.3.1　基本原理

本节介绍几个软件工程活动的通用原理。

1. 推迟实现原理

推迟实现是软件方法学的一条基本指导思想。软件开发过程应该理性地“推迟实现”，即把逻辑设计与物理设计清楚地划分开来，尽可能推迟软件的物理实现。这样在遇到大、中型软件项目时，就可以避免因过早而仓促地考虑程序的具体实现，造成考虑不周而导致大量返工的问题。