选题编辑：于文良　赵树祎
责任编辑：赵树祎
封面设计：刘苏锐

定价：45.00元

中国科大出版社旗舰店
请用天猫／淘宝APP扫描

普通高校计算机类应用型本科
系列规划教材

软件工程

主　编　石　冰
副主编　卜华龙　浮盼盼　陈丽萍

Software Engineering

中国科学技术大学出版社

内 容 简 介

本书针对软件工程课程的特点，全面系统地讲述了软件工程的概念、原理和方法。本书正文共分为11章，第1章是软件工程的概述，第2章介绍了软件过程模型，第3～9章讲述了软件生命周期各个阶段的目标、任务、原则、过程、模式和结构化方法等重要主题，第10章介绍了用面向对象方法进行分析、设计和实现的过程，第11章介绍了软件项目的管理技术。每章的末尾包括了小结、案例分析、阅读材料、习题以及实验。附录给出了一个案例的框架，学习者可以在此基础上做进一步的开发。

本书包括了软件工程理论与实践的最新进展，适合作为本科院校相关专业的教材，也可作为软件开发人员、科研人员以及大专院校师生的参考书。

图书在版编目(CIP)数据

软件工程/石冰主编. —合肥：中国科学技术大学出版社，2017.8
ISBN 978-7-312-04235-5

Ⅰ.软… Ⅱ.石… Ⅲ.软件工程 Ⅳ.TP311.5

中国版本图书馆CIP数据核字(2017)第118827号

出版 中国科学技术大学出版社
安徽省合肥市金寨路96号，230026
http://press.ustc.edu.cn
https://zgkxjsdxcbs.tmall.com
印刷 合肥华苑印刷包装有限公司
发行 中国科学技术大学出版社
经销 全国新华书店
开本 787 mm×1092 mm 1/16
印张 18.75
字数 480千
版次 2017年8月第1版
印次 2017年8月第1次印刷
定价 45.00元

前　言

软件工程课程是为了培养软件开发的高级人才而开设的，它是指导软件生产和管理的一门综合性的应用科学。虽然目前国内软件工程教材品种繁多，但大部分不符合培养技能型人才的需求，与软件企业开发实际相脱节。

本书针对软件工程课程的特点，全面系统地讲述了软件工程的概念、原理和方法，以及软件项目的管理技术，揭示了各知识点之间的内在联系。注重基础性、系统性和实用性，结合大量的软件项目分析实例，深入浅出地阐述了软件工程的具体内容。在软件工程学科的研究与实践的关系方面以及艺术性与科学性的结合方面做了一些探索。

本书包括了软件工程理论与实践的最新进展，反映了最新的业界动态。将软件工程领域中实践者对构造高质量产品的关注和研究者寻找各种方法改进产品质量以及提高开发人员生产效率的努力进行了融合。全书内容丰富，结构严谨，原理与方法相结合，丰富的图表与实例应用相结合，讲解由浅入深，既体现知识点的连贯性、完整性，又注重知识在实践中的应用。

本书正文共分为 11 章。第 1 章是软件工程的概述。第 2 章介绍了软件过程模型。第 3 章介绍了可行性分析与项目计划，第 4 章是软件需求分析，第 5 章是总体设计，第 6 章是详细设计，第 7 章是编码，第 8 章是软件测试，第 9 章是软件维护；在第 3～9 章关于软件生命周期各个阶段的叙述中，分别讲解了其目标、任务、原则、过程、模式、结构化方法和技术等重要主题。第 10 章介绍了用面向对象方法进行面向对象分析、面向对象设计和面向对象实现的过程。第 11 章介绍了软件项目管理的度量、计划、质量和配置。

本书主要有以下特点：

1. 在保证体系完整性的基础上，不过度强调基础理论的深度和难度，坚持“实用为主、够用为度”的原则。

2. 按工程规范化的观点，在过程部分包含了目标、任务、原则和流程，在方法部分包含了模式、方法和技术。

3. 每章的末尾除小结和习题外，还有相应的案例分析和阅读材料以及实验。

4. 附录给出了一个案例（客户关系管理系统）的框架，相关的源码和执行代码可以在出版社云盘上下载到，具体地址为：http://pan.baidu.com/s/1dFzgpyh/，学习者可参考，并可在此基础上做进一步的开发。

本书是作者在近三十年从事该领域的教学实践和软件开发经验的基础上，参考国内外众多最新教材和内容，参阅大量的论著和期刊文献，也引用了开源软件社区的分享经验，在与同行和业界软件开发人员的交流中编写而成的。改变了现有教材侧重理论、对工程的规范化注重不够、可操作性不强的现状。

本书由安庆师范大学石冰任主编，巢湖学院陈丽萍编写了第 1 章、第 8 章的大部分内容以及第 10 章的前 3 节；巢湖学院卜华龙编写了第 2、3、6、7 章和第 8 章的 8.7 节；宿州学院

浮盼盼编写了第4、5、9、11章；石冰编写了本书的其余部分，并对全书进行了审订。客户关系管理系统的开发由石冰和石琦共同完成。

石琦对软件项目计划和管理提出了自己的观点，沈玉玲对优化和代码审查等问题提出了有益的建议，就本书的框架作者与他们进行了多次反复的讨论，他们为作者提供了业界的视角，谨在此向他们表示感谢。另外，还有很多开源社区开发人员分享的经验没有一一注明出处，我们引用的部分图和表也未逐一注明，在此一并致谢。

本书的编写还得到王一宾同志的支持，更是得到了中国科学技术大学出版社编辑的大力支持，在此向他们表示衷心的感谢！

由于作者水平有限，加上时间仓促，书中难免有错误和不妥之处，恳请读者批评指正。如有信息反馈请发至邮箱：shibing@aqnu.edu.cn，以便再版时修订。

编　者

2017年3月

目　录

前言 …… (i)

第 1 章　软件工程概述 …… (1)
1.1　软件与软件危机 …… (1)
1.2　软件工程 …… (3)
1.3　软件生命周期 …… (8)
1.4　软件开发工具 …… (11)
1.5　本章小结 …… (12)

第 2 章　软件过程 …… (15)
2.1　软件过程概述 …… (15)
2.2　软件过程模型 …… (16)
2.3　RUP 统一过程 …… (22)
2.4　敏捷过程与极限编程 …… (26)
2.5　软件过程标准 …… (29)
2.6　本章小结 …… (30)

第 3 章　可行性分析与项目计划 …… (33)
3.1　可行性分析概述 …… (33)
3.2　技术可行性 …… (35)
3.3　经济可行性 …… (37)
3.4　项目计划 …… (39)
3.5　本章小结 …… (41)

第 4 章　软件需求分析 …… (45)
4.1　需求分析概述 …… (45)
4.2　需求获取 …… (50)
4.3　需求建模 …… (54)
4.4　需求规格说明与需求验证 …… (68)
4.5　本章小结 …… (72)

第 5 章　总体设计 …… (81)
5.1　软件设计概述 …… (81)
5.2　设计原则 …… (87)
5.3　设计方法 …… (90)
5.4　软件架构基础 …… (92)

5.5 架构风格 …… (100)
5.6 面向数据流的设计 …… (107)
5.7 数据设计 …… (113)
5.8 本章小结 …… (115)

第 6 章 详细设计 …… (121)
6.1 详细设计概述 …… (121)
6.2 设计准则 …… (122)
6.3 设计模式 …… (124)
6.4 过程设计 …… (127)
6.5 人机界面设计 …… (130)
6.6 设计评审 …… (135)
6.7 本章小结 …… (136)

第 7 章 编码 …… (140)
7.1 程序设计语言 …… (140)
7.2 编码规范 …… (141)
7.3 代码审查 …… (143)
7.4 重构 …… (144)
7.5 程序的复杂性度量 …… (147)
7.6 本章小结 …… (148)

第 8 章 软件测试 …… (151)
8.1 软件测试概述 …… (151)
8.2 软件测试技术 …… (154)
8.3 测试过程 …… (164)
8.4 调试 …… (170)
8.5 测试管理 …… (172)
8.6 软件可靠性 …… (174)
8.7 优化 …… (179)
8.8 本章小结 …… (182)

第 9 章 软件维护 …… (187)
9.1 软件维护概述 …… (187)
9.2 软件维护的特点 …… (188)
9.3 软件维护过程 …… (192)
9.4 软件的可维护性 …… (195)
9.5 软件再工程 …… (200)
9.6 本章小结 …… (202)

第 10 章 面向对象方法学 …… (207)
10.1 面向对象方法学概述 …… (207)
10.2 UML …… (212)

10.3 面向对象分析 …… (221)
10.4 软件重用 …… (228)
10.5 面向对象设计 …… (230)
10.6 框架 …… (233)
10.7 面向对象实现 …… (236)
10.8 本章小结 …… (238)
第 11 章 软件项目管理 …… (244)
11.1 项目估算 …… (244)
11.2 项目进度安排 …… (252)
11.3 项目管理 …… (260)
11.4 质量保证 …… (269)
11.5 软件评审 …… (273)
11.6 配置管理 …… (278)
11.7 本章小结 …… (280)
附录 客户关系管理系统 …… (287)
参考文献 …… (289)

第 1 章　软件工程概述

软件工程项目的最终目标是要开发出符合用户需求的软件产品。

1.1　软件与软件危机

1.1.1　软件

1. 软件的概念

很多人想当然地说:“软件就是程序吧。”答案当然是错误的。这里一定要纠正这个错误的认识:开发软件并不仅仅是指编写程序。那么软件究竟是什么呢?

软件是与计算机硬件相互依存,构成完整计算机系统的另一部分,它是能够实现预定功能和性能的计算机程序、支持程序运行的数据及其相关文档的完整集合。其中,程序是按特定功能和性能要求而设计的能够执行的指令序列;数据是程序运行的基础和操作的对象;文档是与程序开发、过程管理、维护以及使用有关的图文资料。

2. 软件的特点

软件作为一种特殊的产品,具有如下一些特点:

(1) 软件是一种逻辑产品

软件是一种逻辑产品,而不是物理实体,人们无法看到其具体形态,因而具有抽象性。这时必须通过观察、分析、测试、思考和判断去了解它的功能和性能。

(2) 软件不存在磨损、用坏和老化的问题

软件在运行和使用过程中,没有硬件那样的机械磨损、用坏和老化问题,但却存在退化问题,为了适应硬件、系统环境及需求的变化,必须进行升级或者淘汰。

(3) 软件的诞生过程主要是研制,生产只是简单的拷贝

软件是通过人们的智力活动,把知识和信息转化为信息的一种产品,一旦软件开发研制完成,只需复制就可以产生大量的软件产品,所以,在软件开发过程中必须对质量进行良好控制。

(4) 软件成本昂贵

软件的研制尚未摆脱手工方式,它需要投入大量的、复杂的、高强度的脑力活动,它的成本是相当高的。

(5) 软件受硬件和环境的影响

受计算机硬件和支撑环境不断发展的影响,软件必须不断地维护。在软件维护过程中,易产生新的问题,导致软件维护工作量大且成本高于开发成本。

3. 软件的分类

(1) 按照软件功能划分

① 系统软件：是与计算机硬件紧密配合，使计算机系统各部件、相关软件和数据协调、高效地工作的软件。如：操作系统、设备驱动程序等。

② 支撑软件：是协助用户开发软件的工具性软件。如：软件开发环境等。

③ 应用软件：是在特定领域内开发，为特定目的服务的一类软件。如：工程与科学计算软件等。

(2) 按照软件规模划分

① 微型软件：一个人在几天之内完成，程序不超过 500 行。这类软件不必有严格的设计和测试文档。

② 小型软件：不超过 2 个人在半年内完成的 1 000～2 000 行的程序。它通常没有与其他程序的接口。

③ 中型软件：不超过 5 个人在 1～2 年内完成的 5 000～50 000 行的程序。这类软件通常需要有严格的文档和设计规范。

④ 大型软件：由 5～20 个人在 2～3 年内完成的 50 000～100 000 行的程序。这类软件需要按照软件工程方法进行管理。

⑤ 超大型软件：由 100～1 000 个人在 4～5 年内完成的具有 100 万行程序的软件项目。这类软件必须按照软件工程方法开发，有严格的质量管理措施。

⑥ 巨型软件：由 2 000～5 000 个人在 5～10 年内完成的具有 100 万～1 000 万行程序的软件项目。这类软件必须按照软件工程方法开发，有严格的质量管理措施。

(3) 按照软件工作方式划分

① 实时处理软件：指在事件或数据产生时需要立即处理并及时反馈信号，以控制需要监测的部分和控制过程的软件。

② 分时软件：指允许多个联机用户同时使用计算机的软件。

③ 交互式软件：指能实现人机通信的软件。

④ 批处理软件：指把一组输入作业或一批数据以批处理的方式一次运行，并按顺序逐个处理的软件。

1.1.2 软件危机

软件危机是指在计算机软件的开发和维护过程中所遇到的一系列的严重问题，导致了软件生产与市场需求出现极其不适应的现象。

软件危机的表现如下：

1. 软件实现的功能不符合用户的实际需求

软件开发人员对用户的需求缺乏全面、准确和深入的理解，往往急于编程，导致最后实现的软件系统与用户的实际需求相距甚远。

2. 软件生产供不应求

软件开发生产率提高的速度远远低于计算机硬件的发展速度，软件应用需求的增长得不到满足，出现了供不应求的局面，从而使人类不能充分利用计算机硬件提供的巨大潜力。

3. 对软件开发进度和成本估计不准确

软件实际开发成本比估算成本高出几倍，实际进度比预期进度推迟几个月甚至几年。软件开发者为了追赶进度、降低成本和减少损失，采取一些策略，致使软件质量降低。这种现象损害了开发者的信誉，同时又引起了用户的不满。

4. 软件产品质量不可靠

软件质量保证技术没有严密有效地贯穿到软件开发的全过程，导致交付给用户的软件质量差，在运行过程中频繁产生问题，甚至带来极其严重的后果。

5. 软件可维护性差

程序中出现的很多错误非常难改正，要使软件适应新的硬件环境又几乎不可能，也不能根据用户提出的新需求在原有程序上添加一些新的功能，造成软件维护困难和不可重用，使得开发人员重复开发基本类似的软件。

6. 软件文档资料不完整

计算机软件不仅包含程序，还包括软件开发过程中各阶段的文档资料，如：各阶段的说明书、软件测试用例等。这些文档作为软件开发人员交流信息的工具，对于维护人员更是不可缺少。缺少必要的文档资料或文档资料不严密、不正确，必然为软件开发和维护工作带来许多困难。

软件危机的产生主要有两方面原因：一是与软件本身的特点有关；二是软件开发与维护的方法不正确。

软件是计算机系统中的逻辑部件，缺乏“可见性”，程序在计算机上试运行之前，软件开发工作进展情况及软件的质量也较难评价，因此给软件开发和维护带来了困难。此外，规模庞大是软件的一个显著特点，并且程序复杂度将随着程序规模的增加而变得越来越高。为了在预期的时间内完成规模庞大的软件，必须由多人分工合作完成，在这个合作开发的过程中不仅涉及技术问题，而且还必须有严格而科学的管理。

软件本身具有的特点确实给开发和维护工作带来了一些客观困难，但是如果在开发过程中坚持使用已被证明是正确的方法和成功的经验，许多困难是完全可以克服的。但是，目前很多软件开发人员仍然对软件开发和维护工作有错误的认识，并且采用错误的技术、方法和落后的工具。这可能是使软件错误放大并最终导致软件危机的主要原因。错误的认识和方法主要表现为缺乏正确的理论指导、忽视软件需求分析的重要性、轻视软件维护等。

1.2 软件工程

要消除软件危机，主要的途径有以下几种：

(1) 对计算机软件有一个正确的认识。

(2) 软件开发工作不是个体工作，而是由各类人员协同配合、共同完成的工程项目，因此必须加强开发过程的管理，构建良好的组织、严密的管理和协调工作机制。

(3) 推广和使用在软件开发实践中总结出来的成功的技术和方法，探索更好、更有效的技术和方法，尽快纠正在计算机系统早期发展阶段形成的错误概念和做法。

(4) 开发和使用更好的软件工具，软件工具为实现软件技术提供了自动的或半自动的

软件支撑环境。

总之，消除软件危机既要有技术措施又要有管理措施。软件工程正是从这两个方面研究如何更好地开发和维护计算机软件的一门新兴学科。

1.2.1 软件工程的概念

为了消除软件危机，“软件工程”一词是在1968年北大西洋公约组织的会议上首次被提出的。关于软件工程，人们给出了很多的定义，以下是几个典型的定义：

(1) Barry Boehm对软件工程的定义：运用现代科学技术知识来设计并构造计算机程序以及开发、运行和维护这些程序所必需的相关文件资料。

(2) IEEE对软件工程的定义。① 1983年IEEE对软件工程的定义：软件工程是开发、运行、维护和修复软件的系统方法；② 1993年IEEE对软件工程给出了更为全面的定义：把工程化思想应用到软件开发中，把系统化的、规范的、可度量的途径应用于软件开发、运行和维护过程。

(3) Fritz Bauer对软件工程的定义：建立并使用完善的工程化原则，以较经济的手段获得可靠的且能在实际机器上有效运行的软件的一系列方法。

虽然对软件工程定义的各种描述采用了不同的词句，但其基本思想是要用工程化的思想来开发软件。总之，软件工程是应用计算机科学、数学及管理科学等原理指导计算机软件开发和维护的一门工程学科。

软件工程的目标就是运用最先进的技术和经过时间检验证明是正确的管理方法来提高软件的质量和生产率，也就是在给定成本、进度的前提下，开发出满足用户需求的高质量软件产品。从短期效益看，追求高质量和生产率是一对矛盾；但从长远利益看，高质量可保证软件开发的全过程更加规范、流畅，很大程度降低了软件维护代价，实际上是提高了生产率，同时又获得了很好的信誉。因此，好的软件工程方法能同时提高软件质量和生产率。

软件工程包括3个要素：方法、工具和过程。软件工程的层次结构如图1.1所示。

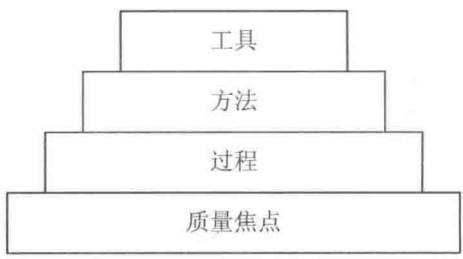

图1.1 软件工程的层次结构

工具层为软件过程和方法提供了自动或半自动的软件支撑环境。目前已有许多不错的软件工具，如：Rational Rose、Power Designer、Eclipse等。如果把各阶段使用的工具有机地结合起来，支持软件开发的全过程，可以有效地改善软件开发过程，提高软件开发的效率。

方法层在技术上说明了如何去开发软件。它包括各种方法：如何进行软件需求分析、如何实现总体结构设计、如何进行测试等。

过程层提供了一个框架，在这个框架下可以建立一个软件开发的综合计划。目的是把软件方法和工具综合起来，并使其被有效利用，能够保证软件及时、合理地开发出来。

质量焦点层以有组织的质量保证为基础，支持软件工程的根基就在于对质量的关注。

1.2.2 软件工程的特性

软件工程具有下述本质特性：

1. 软件工程关注于中大型软件系统的构建

中大型软件系统通常由多人合作完成，涉及技术、人员通信、工具等；传统的程序设计技术和工具是支持小型程序设计的，因此不能简单地把这些技术和工具用于开发中大型系统。

2. 软件工程的中心课题是控制复杂性

通常软件所要解决的问题十分复杂，以致不能把一个问题作为整体直接去考虑，必须把问题分解，分解成的每个部分是可以被理解的，而且各部分之间保持简单的通信关系。用这种方法并不能降低问题的整体复杂性，但却可使它变成可以管理的。

3. 软件经常变化

绝大多数软件都模拟了现实世界的某一部分。为了顺应现实世界的不断变化，保证软件不会很快被淘汰，软件必须也要变化。因此，在软件投入使用后，还需耗费成本。

4. 开发软件的效率非常重要

目前，软件供不应求的现象日益严重。因此，软件工程的一个重要课题是寻求开发软件和维护软件的更有效的方法和工具。

5. 和谐的合作是开发软件的关键

通常软件处理的问题比较大，必须由多人协同工作才能解决这类问题。为了有效地合作，必须明确规定每个人的责任和相互通信的方法，并使每个人严格地按规定行事。

6. 软件必须有效地支持它的用户

开发一个软件的目的是为了支持用户的工作，也就是软件提供的功能可以协助用户有效地完成他们的工作。要做到有效地支持它的用户，就必须仔细地分析、研究用户的相关情况，以确定合适的功能及其他质量要求等；要做到有效地支持它的用户，不但要提交软件产品，还必须提供用户手册和培训材料等。

7. 在软件工程领域中通常由具有一种文化背景的人代替具有另一种文化背景的人创造产品

软件工程师通常是诸如 C++ 程序设计、数据库设计、测试等方面的专家，他们往往并不是教学管理、银行事务等领域的专家，但是他们要为这些领域开发软件系统。缺乏应用领域的相关知识，常常会使软件项目出现问题，所以要求软件工程师要通过访谈等各种方法了解用户组织的工作流程，然后用软件实现这个工作流程，当然用户组织要能真正遵守这个工作流程。

1.2.3 软件工程的基本原理

自 1968 年“软件工程”这个术语被提出并使用以来，研究软件工程的专家学者们陆续提出了 100 多条关于软件工程的准则或“信条”。著名的软件工程专家 B. W. Boehm 综合了这些专家的意见并总结了 TRW 多年来开发软件的经验，于 1983 年提出了软件工程的 7 条基本原理。B. W. Boehm 认为这 7 条原理是确保软件产品质量和开发效率的原理的最小集

合。这7条原理是互相独立的，是缺一不可的最小集合。

下面简要介绍软件工程的7条基本原理。

1. 用分阶段的生命周期计划严格管理

统计发现，在失败的软件项目中有一半以上是由于计划不全面而造成的。在软件开发与维护的漫长生命周期中，需要完成很多性质各异的工作。这意味着应该把软件生命周期划分为若干个阶段，并制订出切实可行的计划，然后严格按照计划对软件的开发和维护过程进行管理。

2. 坚持进行阶段评审

统计发现，大约63%的软件错误是在编码之前造成的，并且错误发现越晚，修改它付出的代价越大。因此，软件的质量保证工作不能等到编码以后再进行，必须贯穿到软件开发的各个阶段。在每个阶段进行严格的评审，尽早地发现错误，以确保软件在每个阶段都能具有较高的质量。

3. 实行严格的产品控制

在软件开发过程中需求的变动会给项目带来巨大的风险，会导致项目的成本增加、延长项目交付周期、影响软件的质量。然而在软件开发中完全避免需求的变动又是不可能的，这就要求我们采用科学的产品控制技术来顺应这种要求。

4. 采用现代程序设计技术

从提出软件工程概念开始，各种新的程序设计技术、先进的软件开发和维护技术就不断被人们研究出来，从20世纪60年代末的结构化软件开发技术到最新的面向对象的技术，实践表明：采用先进的技术不仅可以提高软件开发的效率，而且可以提高软件产品的质量。

5. 结果应能清楚地审查

软件产品不同于一般的物理产品，它是看不见摸不着的逻辑产品，因此软件开发小组的工作进展情况可见性差，难以准确度量、评价和管理。为了更好地管理，必须提高软件开发过程的可见性，应根据软件开发的总目标及完成期限，尽可能明确开发小组的责任和产品标准，从而使得到的结果能够清楚地被审查。

6. 开发小组的人员应该少而精

开发人员的素质和数量是影响软件质量和开发效率的重要因素。开发小组组成人员的素质应该较好，而且人员数量要少。

7. 承认不断改进软件工程实践的必要性

遵循上述6条基本原理，就能按照当代软件工程基本原理较好地实现软件的工程化生产，但是这6条原理并不能保证软件开发和维护的过程赶上技术不断向前发展的步伐。因此应该注意承认不断改进软件工程实践的必要性。这就要求在实际应用过程中，不仅要积极主动地采纳新的软件技术，还要注意不断总结经验。

1.2.4 软件工程方法学

通常把在软件生命周期全过程中使用的一整套技术的集合称为方法学(Methodology)，也称为范型(Paradigm)。在软件工程范畴中，这两个词的含义基本相同。

软件工程方法学包括3个要素：方法、工具和过程。其中，方法是完成软件开发的各项任务的技术方法，回答“如何做”的问题；工具是为方法的运用提供自动的或半自动的软件支

撑环境；过程是为了获得高质量的软件所需要完成的一系列任务的框架，它规定了完成各项任务的工作步骤。

目前使用得最广泛的软件工程方法学有两种，分别是传统方法学和面向对象方法学。

1. 传统方法学

1977 年出现的传统方法学也称为生命周期的方法学，是一种面向过程的方法学，它采用结构化技术(结构化分析(Structured Analysis，SA)、结构化设计(Structured Design，SD)和结构化实现)来完成软件开发的各项任务。这种方法学把软件生命周期的全过程依次划分为若干个阶段，然后顺序地完成每个阶段的任务。包含可行性分析(经济、技术、社会)、需求分析、概要设计、详细设计、编码、测试几个阶段。

传统方法学又称生命周期方法学或结构化范型。一个软件从开始计划起，到废弃不用止，称为软件的生命周期。在传统的软件工程方法中，软件的生存周期分为需求分析、总体设计、详细设计、编程和测试几个阶段。

软件工程学中的需求分析具有两方面的意义。在认识事物方面，它具有一整套分析、认识问题域的方法、原则和策略。这些方法、原则和策略使开发人员对问题域的理解比不遵循软件工程方法时更为全面、深刻和有效。在描述事物方面，它具有一套表示体系和文档的规范。但是，传统的软件工程方法学中的需求分析在上述两方面都存在不足。它在全局范围内以功能、数据或数据流为中心来进行分析。这些方法的分析结果不能直接地映射问题域，而是经过了不同程度的转化和重新组合。因此，传统的分析方法容易隐蔽一些对问题域的理解偏差，与后续开发阶段的衔接也比较困难。

在总体设计阶段，以需求分析的结果作为出发点构造出一个具体的系统设计方案，主要是决定系统的模块结构，以及模块的划分，模块间的数据传送及调用关系。详细设计是在总体设计的基础上考虑每个模块的内部结构及算法，最终将产生每个模块的程序流程图。但是传统的软件工程方法中设计文档很难与分析文档对应，原因是两者的表示体系不一致，所谓从分析到设计的转换，实际上并不存在可靠的转换规则，而是带有人为的随意性，从而很容易因理解上的错误而留下隐患。

编程阶段是利用一种编程语言产生一个能够被机器理解和执行的系统，测试是发现和排除程序中的错误，最终产生一个正确的系统。但是由于分析方法的缺陷很容易产生对问题的错误理解，而分析与设计的差距很容易造成设计人员对分析结果的错误转换，以致在编程时程序员往往需要对分析员和设计人员已经认识过的事物进行重新认识，并产生不同的理解。因此为了使两个阶段之间能够更好地衔接，测试就变得尤为重要。

软件维护阶段的工作，一是对使用中发生的错误进行修改，二是因需求发生了变化而进行修改。前一种情况需要从程序逆向追溯到发生错误的开发阶段。由于程序不能映射问题以及各个阶段的文档不能对应，每一步追溯都存在许多理解障碍。第二种情况是一个从需求到程序的顺向过程，它也存在初次开发时的那些困难，并且又增加了理解每个阶段原有文档的困难。

传统软件工程方法面向的是过程，它按照数据变换的过程寻找问题的结点，对问题进行分解。由于不同人对过程的理解不同，故面向过程的功能分割出的模块会因人而异。对于问题世界的抽象结论，结构化方法可以用数据流图、系统结构图、数据字典、状态转移图、实体关系图来进行系统逻辑模型的描述，得到生产一个最终能满足需求且达到工程目标的软件产品所需要的步骤。

传统软件工程方法学强调以模块为中心，采用模块化、自顶向下、逐步求精的设计过程，系统是实现模块功能的函数和过程的集合，结构清晰、可读性好，是提高软件开发质量的一种有效手段。结构化设计从系统的功能入手，按照工程标准，严格规范地将系统分解为若干功能模块，因此系统是实现模块功能的函数和过程的集合。然而，由于用户的需要和软硬件技术的不断发展变化，作为系统基本组成部分的功能模块很容易受到影响，局部修改甚至会引起系统的根本性变化。开发过程前期入手快而后期频繁改动的现象比较常见。

当然，传统的软件工程方法学也存在很多的缺点，主要表现在生产效率非常低，从而导致不能满足用户的需要，复用程度低，软件很难维护等。尽管如此，传统方法学仍然是人们在软件开发过程中使用得十分广泛的软件工程方法学，在开发某些类型的软件时也比较有效。因此传统软件工程方法学的价值并不会因面向对象方法学的出现而减少，并且它还是学习面向对象方法学的基础。

2. 面向对象方法学

面向对象是围绕现实世界的概念来组织模块，采用对象描述问题空间的实体，用程序代码模拟现实世界中的对象，使程序设计过程更自然、更直观。

面向过程是以功能为中心来描述系统，而面向对象是以数据为中心来描述系统。相对于功能而言，数据具有更强的稳定性。

面向对象方法模拟了对象之间的通信。就像人们之间互通信息一样，对象之间也可以通过消息进行通信。这样，我们不必知道一个对象是怎样实现其行为的，只需通过对象提供的接口进行通信并使用对象所具有的行为功能。而面向过程方法则通过函数参数和全局变量达到各过程模块联系的目的。

面向对象方法把一个复杂的问题分解成多个能够完成独立功能的对象(类)，然后把这些对象组合起来去完成这个复杂的问题。采用面向对象模式就像在流水线上工作，我们最终只需将多个零部件(已设计好的对象)按照一定关系组合成一个完整的系统。这样使得软件开发更有效率。

1.3 软件生命周期

软件的生命周期是指一个软件从提出开发要求开始直到该软件废弃不用的整个时期。从时间的角度可把软件的生命周期依次划分为若干个阶段，即问题定义、可行性研究、需求获取与分析、概要设计、详细设计、编码(包括单元测试)、综合测试(集成测试、确认测试、系统测试、验收测试)、运行与维护。

把软件的生命周期划分为若干阶段，每个阶段都有明确的任务，然后逐步完成每个阶段的任务，使规模较大和管理复杂的软件开发变得容易管理和控制。这几个阶段可归纳为3个时期，即软件定义时期、软件开发时期和软件维护时期。下面将逐一简要介绍软件生命周期各个阶段的目标和主要任务。

1.3.1 软件定义时期

软件定义时期的主要任务包括：确定所要开发软件的总体目标；确定工程的可行性；确定系统必须完成的功能；估算项目所需的资源和成本；制订开发进度表等。软件定义时期通常进一步划分为3个阶段：问题定义、可行性研究和需求获取与分析。

1. 问题定义

问题定义阶段必须要回答的问题是“要解决的问题是什么”，如果不知道待解决的问题是什么就盲目地去做，只能是白白浪费时间和金钱，而且最终得出的结果也很可能是毫无意义的。

2. 可行性研究

在上一阶段清楚了问题的性质、目标后，本阶段就是确定在预定的时间、费用之内该问题是否有行得通的解决方法。如果这些问题没有有效的解决方法，就贸然开发这些项目则会造成时间、人力、资源和资金的浪费。因此，软件可行性研究的目的，就是用最小的代价在尽可能短的时间内确定该软件项目是否能够开发，是否值得去开发。必须记住，可行性研究的目的不是去开发某个软件项目，而是确定某个软件项目是否值得去开发。可行性研究实质上是系统相关开发人员在较为抽象的高层次上进行分析和设计，探索这个问题是否值得去解决，是否有可行的方法，并最终提交可行性研究报告。

可行性研究的结果是客户决定是否继续开发这项软件的重要依据。一般来说，投资可能取得较大效益的软件项目才值得继续下去。

3. 需求获取与分析

这个阶段的任务仍然不是具体地解决问题，而是准确地确定“为了解决这个问题，软件系统必须做什么”，主要确定软件系统必须具备哪些功能。

用户了解他们所面对的问题，知道必须要做什么，但是对他们自身的需求又不能准确而完整地表达出来，也不知道如何使用计算机来解决他们的问题；开发人员知道如何用软件来实现人们的各种要求，但是对特定用户的具体业务和需求不完全清楚。因此需要系统分析员和用户密切联系和配合，充分交流各自的信息，全面地收集、分析用户业务中的信息和处理，导出用户确认的系统逻辑模型，通常用数据流图、数据字典和简要的算法描述来表示。

需求分析阶段确定的系统逻辑模型是以后设计和实现系统的基础，所以该逻辑模型必须准确、全面地体现用户的功能要求和性能要求，并把这些要求以正式文档记录下来，这个文档通常称为软件需求说明书。

1.3.2 软件开发时期

软件开发时期主要是设计和实现在软件定义阶段定义的软件。它通常由4个阶段组成：概要设计、详细设计、编码和单元测试、综合测试。

1. 概要设计

概要设计又称总体设计，该阶段的主要任务是确定应该怎样实现目标软件系统。开发人员要把已确定的软件功能需求转换成需要的软件模块结构，即确定明确的模块、确定模块间的调用关系、确定每个模块的功能是什么。同时还要设计该软件系统要存储哪些数据，这