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
宋效东 史玉梅 李东振 赵菁 等参编

系统分析与设计方法及实践

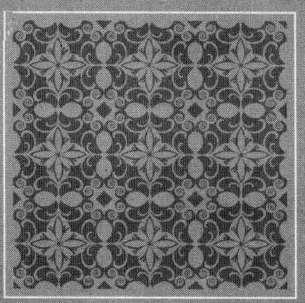
System Analysis and Design
Technique and Practice

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
宋效东 史玉梅 李东振 赵菁 等参编

系统分析与设计方法及实践



System Analysis and Design

Technique and Practice

机械工业出版社
China Machine Press

图书在版编目（CIP）数据

系统分析与设计方法及实践／窦万峰主编. —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 978-7-111-40217-6

Ⅰ. 系… Ⅱ. 窦… Ⅲ. ①信息系统－系统分析－高等学校－教材 ②信息系统－系统设计－高等学校－教材 Ⅳ. G202

中国版本图书馆 CIP 数据核字（2012）第 256385 号

本书分别从传统的结构化开发范型和面向对象开发范型两个方面，把软件分析与设计的概念和理论知识融入实践当中，通过丰富的案例分析与设计，深入地介绍系统分析与设计中各个阶段的技术、方法与典型工具的使用。本书前三部分为软件分析与设计基础、结构化分析与设计、面向对象分析与设计，最后一部分介绍了软件绘图工具 Visio、面向对象建模工具 Rose、数据建模工具 PowerDesigner 3 个流行的软件工程工具。

本书适合作为高等院校软件工程和软件开发课程的教材，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、应用及管理的技术人员，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：朱秀英
北京诚信伟业印刷有限公司印刷
2013 年 1 月第 1 版第 1 次印刷
185mm×260mm · 18.75 印张
标准书号：ISBN 978-7-111-40217-6
定 价：35.00 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

软件分析与设计是软件系统开发的重要组成部分，它包含了一系列原理、方法和实践，指导人们进行正确的软件开发。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。软件分析与设计是一门理论与实践相结合的课程，注重通过实践来理解理论、原理与方法。为此，本书结合作者多年的软件工程教学和项目开发经验，通过6个案例和3个工具软件，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4部分：

第一部分：软件分析与设计基础。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、软件过程生命周期及其模型，以及本教材的案例与要求。

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。这一部分共安排了5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，包括用例需求分析、面向对象分析方法、面向对象设计、统一开发过程和高要求的系统分析与设计等内容。

第四部分：软件分析与设计工具。这一部分共安排了3章内容，介绍软件分析与设计中常见的建模工具的使用和案例开发模型，包括结构化分析工具 Visio、面向对象分析与设计工具 Rose 和面向数据分析与设计工具 PowerDesigner。

编写思想

- 将传统结构化分析与设计和面向对象分析与设计进行对比介绍，有利于学生理解软件开发的不同范型的特点和适用情况，适合作为教材使用。
- 注重介绍软件分析与设计的思想，通过案例分析让学生理解这些思想和原理。
- 通过工具的介绍进一步理解软件开发的过程和实施技术，同时理解企业的主流开发工具与做法。

内容特点

- 分为软件分析与设计基础、结构化分析与设计、面向对象分析与设计和软件分析与设计工具四个方面。对于每一个方面分别介绍软件分析和设计的过程、原理、方法和案例，以及工具支持。
- 将结构化软件分析与设计和面向对象软件分析与设计分开来讲述，并通过不同的案例来帮助读者理解两种范型的特点和适用对象。
- 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。此外，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。

- 面向对象软件分析与设计部分注重面向对象分析模型和设计模型的构建，强调它们之间的关系，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。书中还介绍了高要求系统的分析与设计过程以及建模要点，进一步加深读者对面向对象模型开发本质及其适用情况的理解。统一开发过程的描述帮助读者深入理解当今流行的软件开发过程和具体做法，并通过复杂的系统案例分析理解面向对象分析与设计的思想。
- 案例研究注重分析与思考实现过程，通过三个软件工具 Visio、Rose 和 PowerDesigner 介绍如何应用这些工具快速、准确地开发系统。
- 采用由结构化到面向对象再到工具的路线，便于学生逐步接受软件开发的思想和本质，尤其适合没有任何开发概念（未接触软件工程概念）的学生阅读。

本书内容翔实，有典型案例支持，便于读者学习和深入体会软件分析与设计的原理和方法。

本书第 1～8 章由窦万峰编写，第 9～10 章由窦万峰和彭涛编写，第 11～12 章由窦万峰和洪奎编写，第 13 章由窦万峰和赵菁编写，第 14～15 章由宋效东编写，第 16 章由史玉梅和李东振编写。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、应用及管理的技术人员。

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。

教学建议

本书围绕软件工程两个开发范型（结构化分析与设计和面向对象分析与设计）及其案例与工具的讲解，分为四个部分，教师可根据教学课时和侧重点进行灵活教学。

第一部分着重介绍软件工程基本原理和基本过程，可用 1.5～2 课时介绍，讲授内容主要包括软件工程的基本原理和思想、生命周期、过程模型、软件过程活动等。如果课时足够的话，也可以介绍一下敏捷开发思想和结对编程的概念。对于研究性教学，建议让学生阅读结对编程的有关材料，展开介绍面对面结对编程和分布式结对编程的优缺点，并讨论如何改进等。在这一部分最后，教师可以安排学生的学期课程项目作业，建议安排两种类型的项目，一种是结构化分析与设计的项目（如图书馆系统等），另一种是面向对象分析与设计的项目（如 POS 机系统等）。

第二部分着重介绍结构化开发方法学，包括非形式化需求分析、结构化分析建模、结构化概要设计和结构化详细设计等。建议课时为 8～10 课时，围绕项目作业介绍结构化分析过程和主要技术，重点是讲授结构化的分析建模技术（包括实体-关系模型、数据流模型和状态机模型）、结构化概要设计技术（包括面向数据流的设计方法、面向数据的设计方法）和结构化详细设计技术（包括结构化设计描述工具、人机交互界面设计、数据库设计等）。在这一部分，学生要完成结构化的项目作业，包括软件需求规格说明书、软件概要设计说明书和软件详细设计说明书的编写，其中含各种模型的表示图等。如果时间充裕的话，可安排学生汇报并进行讲评。

第三部分着重介绍面向对象开发方法学，包括面向对象分析、面向对象设计、统一过程模型、形式化技术与方法（选讲）等。建议课时为 8～10 课时，围绕项目作业介绍面向对象分析与设计建模技术和 UML 描述方法，主要内容包括用例模型、类模型、交互模型、部署模型，以及设计模式和实现方面的讨论等。另外，根据课时安排，还可介绍形式化的设计与分析方法。在这一部分，学生要完成面向对象的项目作业，包括基于面向对象的软件需求分析报告、软件设计报告和代码框架报告的编写，其中含各种模型图等。如果时间充裕的话，可安排学生汇报并进行讲评。

第四部分是软件工具部分，主要安排在实验教学中完成，学生在实验中运用这些工具完成项目作业的各种模型的构建和表示。实验教学建议安排 8～10 个实验，其中 Visio 工具适合于结构化的分析与设计建模，包括业务建模、领域建模、数据流建模、数据结构建模、数据库建模、软件结构建模和模块设计等，可用 6～8 课时完成；PowerDesigner 侧重于数据建模，用于数据关系建模、业务流程建模、面向对象建模、概念建模、物理建模等，可用 4～6 课时完成；Rose 工具侧重于面向对象建模，用于用例建模、领域建模、行为建模、部署建模等，可安排 6～8 课时完成。

对于学期项目，建议学生 4～5 人为一组，设组长 1 名，培养学生的团队合作精神和组织协调能力。如果项目较大的话，每个组可完成部分子系统，几个组共同完成整个项目。

目　录

第一部分

软件分析与设计基础

第 1 章

软件分析与设计概述

软件系统分析与设计是软件工程（Software Engineering，SE）的重要组成部分，其目的是倡导以工程化的原理、原则和方法进行软件系统开发，是解决当时出现的“软件危机”的根本途径。

1.1 什么是软件

软件分析与设计的主旨是以工程化的思想进行软件开发，以便生产出高质量和高效率的软件系统，即软件分析与设计研究的基础就是软件。那么，软件是怎么定义的呢？它有哪些特性呢？

1.1.1 软件定义与特性

软件是计算机系统中与硬件系统相对应的部分，包括一系列程序、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；文档是与程序开发、维护和使用有关的图文资料。软件系统的核心是程序，而文档则是软件系统不可分割的组成部分。

要理解软件的真正含义，首先需要了解软件有哪些特性。人们利用结构化的思想创造出的软件是逻辑的，而不是有固有形态的实体，所以，软件具有以下特性：

1）复杂性：软件是一个庞大的逻辑系统，比任何人类构造的其他产品都复杂，甚至硬件系统的复杂性和软件系统比起来也是微不足道。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，增加了其复杂性。软件的复杂性使得软件难以理解、生产、维护，更难以对生产过程进行管理。

2）一致性：软件系统必须和运行软件的硬件系统保持一致，这是由软件系统对硬件系统的依赖所决定的。如果硬件系统是“现有”的，那么软件系统必须与现有硬件系统对接。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系统来替代硬件系统部分功能的情况。

3）抗磨损性：软件系统的运行周期与一般的器械设备系统截然不同，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一般都需要进行多次维护。图 1-1 给出了软件系统的理想故障曲线和实际故障曲线。

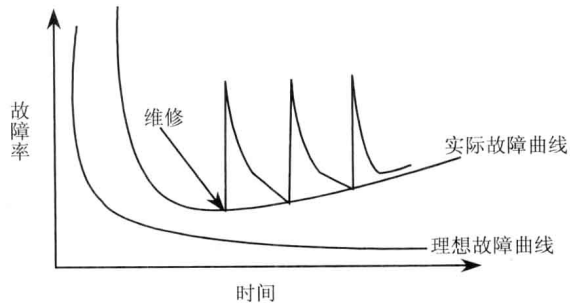


图 1-1　软件系统的理想故障曲线和实际故障曲线

4）易变性：软件在生产过程中，甚至在投入运行之后，也可以再改变。软件必然需要变化，这是软件的特有属性。改变软件往往可以收到改变或者完善系统功能的效果，且比更换硬件系统容易，使得软件系统易维护、易移植、易复用。但是，修改软件导致软件始终在“变”，这种动态的变化不仅难以预测、难以控制，还可能对软件的质量产生负面影响。

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，这里就涉及软件的可移植性了。好的软件在设计时就考虑到软件如何应用到不同

的系统平台。

6）高成本性：软件系统的开发是一个复杂的过程，显然，软件系统的成本非常高昂。

1.1.2　软件的演化

软件的发展经历了一个演化的过程，自从 20 世纪 40 年代生产出世界上第一台计算机后，伴随而生的就是程序。纵观前后的几十年，软件的演化大致经历了 4 个阶段：

1）程序设计阶段（第一阶段）：从 1946 年到 20 世纪 60 年代初，是计算机软件系统发展的初期，其主要特征是程序生产方式为个体手工方式。

2）程序系统阶段（第二阶段）：从 20 世纪 60 年代初到 70 年代初，软件工程学科诞生。当时，程序的规模已经很大，需要多人分工协作，程序的开发方式由个体生产发展到了小组生产，其开发与维护费用以惊人的速度增加。因此许多程序系统后来根本不能维护，最终导致了严重的软件危机。

3）软件工程阶段（第三阶段）：从 20 世纪 70 年代中期至 80 年代中期，软件工程师把工程化的思想加入软件系统的开发过程中，用工程化的原则、方法和标准来开发与维护软件。

4）面向对象阶段（第四阶段）：从 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。

随着计算机的普及，程序的稳健性和易读性受到了广泛的关注，于是，程序从个人按自己意图创造的“艺术品”转变成了能被广大用户接受的“工程化”产品。由于外部环境和用户需求的不断变化及软件开发技术的不断发展，注定了软件系统只有不断的演化才能适应用户的新需求。

从整个系统的角度看，开发软件系统的目的是为了满足用户的需求，提高生产率，因此软件系统的需求仍是软件发展的动力。早期的程序开发者只是为了满足自己的需要，这种自给自足的生产方式仍然是其低级阶段的表现。进入软件工程阶段以后，软件系统的开发具有社会属性，它要在市场中流通以满足更多用户的需要。

软件演化过程的各个阶段也有不同的特征，在这些阶段中，软件系统开发的范围从只考虑程序的编写扩展到涉及整个软件生命周期。

1.1.3　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，人们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提出的要求，导致问题不断积累，形成了日益尖锐的矛盾。这就导致了软件危机。

这场软件危机主要表现在：软件系统的规模越来越大，复杂度不断增加，软件系统的需求量也日益增大，且价格昂贵，供需差日益增大。而软件系统的开发过程是一种高密集度的脑力劳动，软件系统开发常常受挫，质量差，很难按照指定的进度表来完成指定的任务，软件系统的研制过程很难管理，往往失去控制。软件系统开发的模式及技术已经不能适应软件系统发展的需要，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，有的软件系统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 亿英镑，历时五年，但最终还是失败，导致伦敦股票市场声誉下跌。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。

1.1.4　软件危机的解决途径

在软件危机相当严重的背景下，软件工程产生了。在引入工程化的思想后，人们总结了出现软件危机的原因并提出了相应的解决对策。

在软件开发的初期阶段，需求提得不够明确，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。如果前期的需求分

析不到位，认为软件的开发仅仅是编写程序，很有可能导致后期开发的软件达不到客户的要求，并进一步导致软件的二次开发。

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，同时又在无形中提高了软件的质量，毕竟软件是一种商品，提高质量是软件开发过程中的重中之重。

开发过程要有统一的、公认的方法论和规范指导，参加的人员必须按照规定的方法论进行开发。重视设计和实现过程的资料，不要忽视每个人的工作与其他人的接口，以便后期能够较好地进行软件的维护工作。由于软件是逻辑部件，开发阶段的质量较难衡量，开发过程管理和控制同样不易实现，这就需要开发人员必须有统一的软件开发理论来指导。

软件工程师必须在测试阶段做好充分的检测工作，提交给客户高质量的软件。要借鉴软件开发的经验，积累与软件开发有关的数据，确保开发工作按时完成。

1.2　什么是软件分析与设计

软件分析与设计是软件工程的重要组成部分，其定义目前还没有统一的标准。早期，软件工程专家 B.W. Boehm 将软件工程定义为：设计并构造计算机程序，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 如下定义软件工程：为了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 软件工程标准定义软件工程为：开发、运行、维护和修复软件的系统方法。

尽管软件工程的具体定义不尽相同，且又有一些学者提出了更完善的定义，但都是在强调在软件开发的过程中需要应用工程化思想的重要性。我们综合考虑上述的定义，给出软件系统分析与设计的定义为：为开发满足用户需要的软件系统，而采用工程化思想进行分析和设计的原则和方法以及实践。

1.2.1　基本原理

本节介绍软件开发活动的一些基本思想和原理。

1．工程化原理

工程化思想的核心是，把软件看作一个工程产品，这种产品需要经过需求分析、设计、实现、测试、管理和维护一系列过程。用完善的工程化原理研究软件系统生产的规范方法，这样，软件的开发不仅会在指定的期限内完成，还会节约成本，保证软件的质量。

通过工程化的思想，一方面可以提高软件的质量，降低成本；另一方面可以为软件的工程化开发提供保障。随着软件行业的迅猛发展，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、项目预算成倍超过、项目人员不断流动等都是软件开发商不断面临的问题。其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；管理者忙于应付突发事件，对产品质量缺乏客观认识；软件开发的成败建立在个人能力的基础上。

2．推迟实现原理

推迟实现是软件开发方法学的基本指导思想。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。这样就避免了在软件开发过程中遇到大中型的软件项目时，由于过早而仓促地考虑程序的具体实现，但考虑不周而导致大量返工。

3．逐步求精原理

逐步求精也称逐步细化，它承认人类思维能力的局限性（认为人类思维能力仅有 7±2 信息量子的局限性），将一个复杂问题有条理地从抽象到具体，逐步分解、细化和解决。这是人类把复杂问题趋于简单化控制和管理的有效策略。逐步求精解决复杂问题的策略是软件工程方法学中的一项通用技术，且与分解、抽象和信息隐蔽等概念紧密相关。

4．系统分解原理

系统分解是把复杂问题趋于简单化处理的有效策略。根据人类解决一般问题的经验，可得出如下规律：如果一个问题由两个问题组合而成，那么它的复杂程度大于分别考虑每个问题的复杂程度之和；同理，如果一个复杂问题分解成若干容易解决的小问题，那么就能够减少解决问题所需要的总工作量。当然，系统分解必须是科学而合理的，否则可能会增加解决问题的难度和工作量。

5．系统抽象原理

抽象是把一些事物（状态或过程）中存在的相似的方面（忽略它们的差异）概括成“共性”。抽象的主要思想是抽取出事物的本质特性，而暂不考虑它们的细节，即抓“大”放“小”。这是一种分层次的渐进过程。软件系统开发广泛采用分层次的从抽象到具体的逐步求精技术。建立模型是软件系统开发最常用的方法和技术之一。模型是一个方法或过程的合理而规范的框架。

6．信息隐蔽原理

科学而合理的分解，还表现在得到的是一个个最简单、最清晰的“独立”部分，即这些部分的交互接口简单而清晰。不仅便于维护，而且利于复用。信息隐蔽是指“局部化”的信息（关系密切的软件元素，如实现过程、数据等），对于不需要了解这些信息的其他“局部”来说是不可访问的（隐蔽的）。信息隐蔽意味着把一些关系密切的软件元素物理地放得彼此靠近，使信息最大限度地局部化。软件模块中使用局部数据元素就是局部化的一个例子。

1.2.2　基本原则

美国著名的软件工程专家 B.W. Boehm 综合不同专家关于软件开发的意见，并总结了多家公司开发软件系统的经验，于 1983 年提出了软件分析与设计的 7 条基本原则。Boehm 认为，这些原则是确保软件产品质量和开发效率的原则的最小集合。它们是相互独立的，是缺一不可的最小集合。

1．分阶段的开发原则

根据这条基本原则，可以把软件开发划分为若干个阶段，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开发与维护进行管理。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。Boehm 认为，在整个软件生命周期中应制定并严格执行 6 类计划：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、运行维护计划。

2．阶段评审原则

统计结果显示：在软件系统开发的各阶段中，编码阶段之前的错误约占 63%，而编码错误仅占 37%。并且，错误发现得越晚，更正它付出的代价就会越大，要差 2～3 个数量级甚至更高。因此，软件系统的质量保证工作不能等到编码结束以后再进行，必须坚持进行严格的阶段评审，以便尽早地发现错误。

3．严格的控制原则

改动需求是让开发人员很头痛的一件事。但是，实践告诉我们，需求的改动往往是不可避免的。这就要求我们采用科学的产品控制技术来顺应这种要求。其中主要是实行基准配置管理（又称为变动控制），即凡是修改软件的建议，尤其是涉及对基准配置的修改，都必须按规定进行严格的评审，评审通过后才能实施。这里的“基准配置”指的是经过阶段评审后的软件配置成分及各阶段产生的文档或程序代码等。当需求变动时，其他各个阶段的文档或代码都要随之相应变动，以保证软件系统的一致性。

4．采用有效的开发技术原则

采用先进的软件分析与设计技术既可以提高软件系统开发与维护的效率，又可以提高软件系统的质量并减少维护的成本。

5．明确责任原则

软件系统是一种逻辑产品，软件开发小组的工作进展情况可见性差，难以评价和管理。为更好地进行管理，应根据软件系统开发的总目标及完成期限，尽量明确地规定开发小组的责任和软件产品标准，从而使所得到的标准能清楚地审查。

6．人员应少而精原则

开发人员的素质和数量是影响软件系统质量和开发效率的重要因素，应该少而精。事实上，高素质开发人员的工作效率比低素质开发人员的工作效率要高几倍到几十倍，开发工作中犯的错误也要少得多；当开发人员增加时，可能的通信信道会随着人数的增加而增大，而通信开销也将急剧增大。

7．不断改进开发过程原则

在软件系统的工程化生产中，我们不仅要积极采用新的软件开发技术，还要注意不断总结经验，收集进度和消耗等数据，进行出错类型和问题报告统计。这些数据不仅可以用来评估新的软件技术的效果，还可以用来指明必须着重注意的问题和应该优先进行研究的工具和技术。

1.3　软件系统开发范型

范型的意思是指模型或模式；而在软件工程学科中，范型用来表示一套涵盖整个软件生产过程的技术的集合。目前使用得最广泛的软件工程方法学，分别是结构化开发范型和面向对象开发范型。

1.3.1　结构化开发范型

结构化开发范型自 1968 年被提出，经过了多年的发展，形成了一套完整的体系。构成结构化开发范型的技术包括结构化分析、结构化设计、结构化编程和结构化测试，这些技术在以数据为主的系统或小型系统方面得到广泛应用。采用结构化技术来完成软件系统开发的各项任务，并使用适当的软件工具或软件系统开发环境来支持结构化技术的运用。这种范型把软件的生命周期依次划分为若干个阶段，然后顺序地完成每个阶段的任务。采用这种范型开发软件的时候，从对问题的抽象逻辑分析开始，一个阶段接一个阶段地进行开发，从而降低了整个软件开发工程的困难程度。引入结构化开发范型所带来的主要改进体现在软件工业上。结构化开发范型在工业领域的运用是软件工程被大规模采纳的主要原因。

结构化开发范型获得成功的原因是，结构化技术考虑问题要么面向行为，要么面向数据。软件的基本组成部分包括软件的行为和这些行为操作的数据。有些结构化技术，如数据流分析是面向行为的，这些技术集中处理产品的行为，数据则是次要的。反之，有些结构化技术，如 JSP 系统开发技术是面向数据的，这些技术以数据为中心，在数据上操作的行为则是次要的。

1.3.2　面向对象开发范型

随着软件系统的规模不断扩大，结构化开发范型越来越难以应付。也就是说，结构化开发范型处理 5000 行或 50 000 行代码是有效的，然而，当今的软件系统，50 000 行或 5 000 000 行甚至更多行代码的产品非常普遍。

面向对象开发范型把数据和行为看成同等重要，即将对象视作一个融合了数据及在其上操作的行为的、统一的软件组件。对象的概念符合业务或领域的客观实际，反映了实际存在的事物，也符合人们分析业务本质的习惯。

面向对象技术自 20 世纪 90 年代提出以来得到快速发展，并被应用于各种各样的软件开发中。面向对象技术将数据和数据上的操作封装在一起，对外封闭，实现了信息隐藏的目的。使用这个对象的用户只需知道其暴露的方法，通过这些方法来完成各种各样的任务，完全不需要知道对象内部的细节，保证了相对独立性。

面向对象技术的优势主要体现在维护阶段。相对于结构化技术，面向对象开发范型无论对象的内部细节如何变化，只要对象提供的方法（即接口）保持不变，则整个软件产品的其他部分就不会受到影响，不需要了解对象内部的变化。因此，面向对象开发范型使维护更快、更容易，同时产生回归错误的机会也大大降低。

面向对象开发范型使开发变得相对容易。大多数情况下，一个对象对应物理世界中的一个事物。软件产品中的对象和现实世界的同等对应物之间的密切对应关系，促进了更优化的软件开发。对象是独立的实体，因此面向对象开发范型促进了复用，降低了开发维护的时间和费用。

目前，传统的结构化开发范型仍然是人们在开发软件时使用得十分广泛的软件工程方法学。广大软件工程师对这种范型比较熟悉，而且在开发某些类型的软件时也比较有效，因此，在相当长一段时期内这种方法学还会有生命力。此外，如果没有完全理解传统范型，也就不能深入理解这种范型与面向对象开发范型的差别以及面向对象开发范型为何优于传统范型。在使用结构化开发范型时，分析阶段和设计阶段过渡太快，而面向对象开发范型是一种迭代的从一个阶段向另一个阶段过渡的范型，比结构化开发范型平滑得多，从而降低了开发过程中的故障数。

1.4　软件生产活动

在软件工程概念被提出来之前，开发人员错误地认为，软件就是编码，至于分析和设计等都是次要的。随着软件规模的不断增大，软件生产过程中暴露出很多问题。软件工程是为克服这些问题（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。软件开发的工程化思想让开发人员看到，软件生产活动不仅是开发活动，还有重要的维护活动、管理活动，进而发展了过程改进活动。

1．开发活动

开发活动是软件人员生产软件的活动。开发活动是软件工程的核心过程活动，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。开发活动有一系列的阶段，如需求、设计、编码、测试、提交、维护等。这些阶段需要采用一定的控制流程连接起来，并需要规范的操作方式，这样就形成了软件生命周期模型。

软件开发活动是随着开发技术的演化而随之改进的，例如从早期的瀑布模型、螺旋模型，到当今的敏捷开发方法和统一过程。它们展示出了在不同的时代，软件产业对于开发活动的不同认识以及对于不同类型项目的理解方法。

2．维护活动

软件开发完成交付用户使用后，就进入软件的运行和维护阶段。软件维护就是在软件交付运行后，保证软件正常运行、适应新变化等需要而进行的一系列修改活动。软件维护的主要工作就是在软件运行和维护阶段对软件产品进行必要的调整和修改。

软件维护是软件生命周期的最后一个阶段，也是持续时间最长、工作量最大的一个不可避免的过程。软件维护的基本目标和任务是改正错误、增加功能、提高质量、优化软件、延长软件寿命以及提高软件产品价值。

3．管理活动

当今的软件开发活动是一个非常复杂的过程。项目涉及几十、几百甚至几千的人员，项目周期少则几个月，多则几年，项目费用越来越高，因此，这样的项目就需要很好的管理活动。著名的项目管理专家 James P. Lewis 指出，项目是一次性的、多任务的工作，具有明确的开始和结束日期、特定的工作范围、预算和要达到的特定性能水平。因而，项目涉及预期的目标、费用、进度和工作范围 4 个要素。

软件项目管理活动就是如何管理好项目的范围、进度、成本等。为此需要制定一个好的项目计划，然后跟踪与控制好这个计划。实际上，要做到项目计划切合实际是一个非常高的要求，需要对项目进行详细的需求分析，制定合理的计划，安排好进度、资源调配、经费使用等，并