“十三五”江苏省高等学校重点教材 编号2016-2-039

软件系统分析与体系结构设计

杨洋 刘全·编著

东南大学出版社
SOUTHEAST UNIVERSITY PRESS

软件系统分析与体系结构设计

杨 洋 刘 全 编著

东南大学出版社
SOUTHEAST UNIVERSITY PRESS
·南京·

内容提要

本书旨在从基础着手，系统地介绍软件系统分析和体系结构设计的原理、方法和实践。全书分为11章，主要内容包括软件工程基础概述(第1章)、结构化分析和设计方法(第2章)、面向对象的分析和设计方法(第3～7章)以及软件体系结构设计(第8～11章)。

第1章系统介绍了软件工程的相关背景知识。第2章简要描述了传统的结构化方法的要点和分析设计的步骤。第3～7章详细讨论了面向对象方法的要点和分析设计的步骤，包括用例建模、静态建模、动态建模和实现建模等，并以UML为建模语言，Rational Rose为工具，给出了较完整的示例。第8～11章讨论了软件体系结构的风格、设计和评估等内容。

本书可以作为各类计算机相关专业本科学生的教科书，或者供软件开发人员参考，也适合推荐给有兴趣系统学习软件开发的非计算机专业的学生自学使用。

图书在版编目(CIP)数据

软件系统分析与体系结构设计 / 杨洋，刘全编著.
南京 ：东南大学出版社，2017.10
ISBN 978-7-5641-7356-2

Ⅰ.①软… Ⅱ.①杨… ②刘… Ⅲ.①软件工程—系统分析 ②软件设计 Ⅳ.①TP311.5

中国版本图书馆CIP数据核字(2017)第192543号

软件系统分析与体系结构设计

出版发行	东南大学出版社
出 版 人	江建中
社　　址	南京市四牌楼2号
邮　　编	210096
经　　销	全国各地新华书店
印　　刷	兴化印刷有限责任公司
开　　本	787 mm×1092 mm　1/16
印　　张	13
字　　数	316千字
版　　次	2017年10月第1版
印　　次	2017年10月第1次印刷
书　　号	ISBN 978-7-5641-7356-2
定　　价	35.00元

(本社图书若有印装质量问题，请直接与营销部联系。电话：025-83791830)

前　言

软件系统分析与设计是软件工程的核心内容之一，也是软件工程等计算机相关专业本科生的一门重要必修课。分析和设计由于处在软件开发的前期，对软件产品的质量保障起着关键的基础作用。但在实际工程开发中，往往没有被足够重视，广泛存在需求定义不规范、分析不充分、模型和体系结构设计不合理等诸多问题，导致软件质量低劣且很难更正。本书旨在从基础着手，系统地介绍软件系统分析和体系结构设计的原理、方法和实践，可以作为各类计算机相关专业本科学生的教科书，或者供软件开发人员参考，也适合推荐给有兴趣系统学习软件开发的非计算机专业的学生自学使用。

本书首先复习了软件工程中的相关重要概念和背景知识，令没有软件工程基础的读者也能快速入门。这个部分特别强调了软件工程中的核心原则，即“系统化”、“规范化”和“可度量”。这也是本书的理论基础，在全书中贯穿始终。

其次，从软件工程中“模型＋方法＋工具”的多个角度，讨论了多种软件开发过程模型、结构化和面向对象方法、UML 工具和以 Rational Rose 为代表的 OOCASE，使读者全面理解这些概念之间的关系并灵活运用。

第三，本书特别将体系结构设计和系统分析有效结合讲解。编者在十余年的一线教学实践中发现，软件开发中如何选择、设计恰当的体系结构和开发框架与软件需求密不可分，而软件分析和设计中复用的思想也是当前体系结构设计中的核心要求。因而，将这两者结合讲解，更有利于学生对知识点的深入理解和实践运用。

第四，教材突出“主题＋案例”式教学方法和过程的展示，提供了一批实际案例，其内容不仅仅是分析和设计的结果展示，更包含了完整的项目制作实例的过程记录。这些实例来源于编者的教学和开发实践，具有一定的应用价值。

最后，本教材从实践出发，特别加入了软件设计模式的内容，具体地讲述了如 MVC 等多个常用的设计模式，并给出了部分示范代码，提高了理论知识的可理解性和可操作性。

吴芝婧、王昇、陈鑫鑫等参加了本书的部分编撰工作。本书的编写得到了苏州大学计算机学院师生的支持，在此表示感谢。由于编者水平有限，时间仓促，难免有疏漏、谬误之处，恳请读者批评指正。

杨　洋

目　录

第1章 软件工程概述

自1968年“软件工程”的概念提出至今，已有近半个世纪。作为一门从实践中抽象出来的年轻学科，软件工程的理论并未完全成熟，仍在实践中不断向前发展。软件系统的分析和设计是软件工程中的核心内容，能够从源头提高软件质量，避免软件危机的产生。而近年来基于面向对象的思想的迅猛发展和广泛应用，逐渐影响分析设计方法的改进，促进传统的软件开发方法不断更新。

本章将简要回顾软件工程发展中的历史里程碑，从系统层面探讨软件工程的开发模型，介绍经典的结构化方法和面向对象方法的基本要点，从应用层面了解软件工程的工具CASE，从而帮助读者在具体应用领域最大限度地发挥思维能力和创造性，为开发高质量的软件系统提供支持。

学习目标

- 了解软件危机与软件工程的重要基本概念
- 理解软件开发的过程模型与基本方法
- 理解面向对象思想中的重要概念
- 了解软件工程工具CASE

1.1 软件危机与软件工程

从1946年第一台计算机发明以来，微电子学技术的不断进步使得计算机硬件的性能和质量有了巨大且持续稳定的提高，著名的摩尔定律即描述了硬件发展的情况。与此同时，软件业也进入了一个高速发展的时期，但早期自由的软件开发方式、急剧膨胀的软件规模、日趋复杂的需求以及不断上升的软件成本，导致软件质量不尽如人意，开发效率非常低下。软件开发的滞后发展日益成为限制计算机技术在工业发展和国民生活中更广泛应用的关键因素。

更严重的是，计算机系统发展早期所形成的一系列错误概念和做法，已相当程度地阻碍了计算机软件的开发和维护，造成大量时间、人力、物力的浪费，从而导致软件危机的产生。

1.1.1 软件危机

软件危机泛指软件开发和维护过程中遇到的一系列严重的问题，概括来说包含两个方面：

(1) 如何开发软件,以满足不断增长的、日趋复杂的要求;

(2) 如何维护规模不断庞大的软件产品。

软件和硬件的开发、运行、维护过程有着本质的不同。从软件本身的特点来看,只要应用环境和应用需求没有发生改变,其运行和使用期间不会出现硬件那样的机械磨损、老化问题。如图 1-1 所示的是硬件失效率随时间变化的曲线,可以看到在硬件使用初期和磨损一段相当长的时间后,失效率较高,而在中间时间段,硬件工作状态稳定,失效率较低。如图 1-2 所示的是软件的理想失效率曲线和实际失效率曲线。在绝对理想情况下,虽然软件在其生命周期初期的失效率较高(这往往是由于用户对软件不够熟悉导致),但随着时间的推移,软件故障不断排除,失效率逐渐趋近于零,并且可以持续使用,永不失效;而实际中,软件在其生命周期内是需要根据应用环境和需求变化不断维护和更新的,因此故障曲线呈现锯齿状,并依然呈现上升的态势。

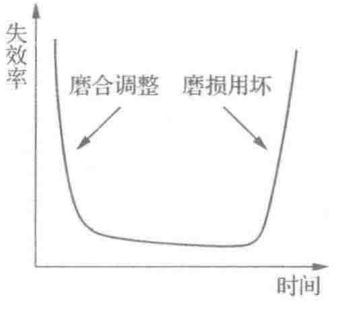

图 1-1 硬件失效率曲线

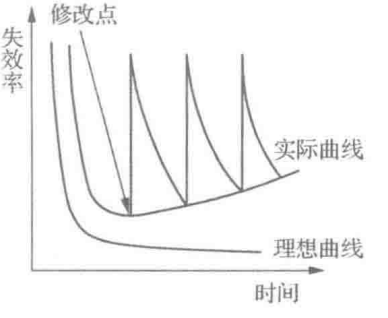

图 1-2 软件失效率曲线

具体地说,软件危机主要体现在如下方面:

(1) 软件开发成本日益增长;

(2) 软件开发进度难以控制;

(3) 软件质量难以保证;

(4) 软件维护困难。

案例学习

◎ 丹佛新国际机场行李传送控制系统:该机场是美国面积最大及全世界面积第二大的机场,拥有美国最长的跑道。该机场计划投资 1.93 亿美元建立一个地下行李传送系统,总长 21 英里,有 4 000 台遥控车,可按不同线路在 20 家不同的航空公司柜台、登机门和行李领取处之间发送和传递行李;支持该系统的是 5 000 个电子眼、400 台无线电接收机、56 台条形码扫描仪和 100 台计算机。按原定计划要在 1993 年万圣节前启用,但一直到 1994 年 6 月,机场的计划者还无法预测行李系统何时能达到可使机场开放的稳定程度。

◎ IBM 公司经典软件危机案例:该公司开发 OS/360 系统,共有 4 000 多个模块,约 100 万条指令,投入 5 000 人/年,耗资数亿美元,结果还是延期交付。在交付使用后的系统中仍发现大量(2 000 个以上)的错误。

造成软件危机的原因有很多，例如：软件本身的复杂性、软件产品的特殊性、人们认识的局限性等。其中软件本身的复杂性是核心原因，它主要包括以下两点：

(1) 开发结构的逐渐复杂性。例如，Windows 95 有 1 000 万行代码，Windows XP 达到了 4 000 万行代码，而 Windows 7 使用了 20 多个开发小组、近千名程序员，即使经过代码复用和优化，代码总行数也超过了 5 000 万行。甚至以内核简洁著称的 Linux 在 2.6.27 版之后，其源代码也超过了 1 000 万行。

(2) 软件技术的发展复杂性。如图 1-3 所示，软件技术发展 5 个阶段的典型技术比较显示了软件技术逐渐复杂化的发展过程。

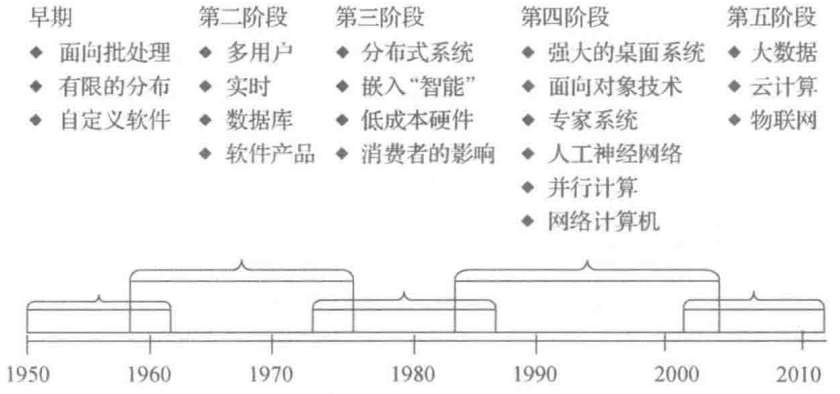

图 1-3　软件技术发展 5 个阶段的典型技术比较

要解决软件危机，技术(包括方法和工具)和必要的组织管理不可或缺。软件工程正是从技术和管理两方面，研究如何更好地开发和维护计算机软件的实践科学。

1.1.2　软件工程的定义

在 1968 年联邦德国召开的国际会议上正式提出并使用了软件工程这个术语，即运用工程学的基本原理和方法来组织和管理软件生产，从此人们对于软件开发是否已符合工程化思想这一核心问题进行了长达数十年的探索，并对软件工程这门学科有何自身特点等问题展开了广泛的讨论与研究，不同时代形成了对软件工程的不同定义，列举部分如下：

1. Fritz Bauer(1968) in NATO

"建立并使用完善的工程化原则，以较经济的手段获得能在实际机器上有效运行的可靠软件的一系列方法。"

2. IEEE(1983)

"软件工程是开发、运行、维护和修复软件的系统方法。"

3. IEEE(1993)

"(1) 将系统化的、规范的、可度量的方法应用于软件的开发、运行和维护的过程，即将工程化应用于软件中。

(2) 在(1)中所述方法的研究。"

可以看到，和其他工程学一样(建筑、电子、机械等)，软件工程的核心在于采用工程化的概念、原理、技术和方法，把经过时间考验而证明正确的技术和管理方法结合起来，对软件进

行系统化、规范化和可度量的开发。

我们把研究软件工程的科学称为软件工程学，其主体知识大致分为 10 个领域（《SWEBOK 指南》，Guide to the Software Engineering Body of Knowledge），包括：

- 软件需求
- 软件设计
- 软件构造
- 软件测试
- 软件维护
- 软件配置管理
- 软件工程管理
- 软件工程过程
- 软件工程工具和方法
- 软件质量

对软件工程的研究常常从三个层面进行，分别是软件的开发模型、开发方法和开发工具。随着软件工程的发展，涌现了很多的模型、方法和工具，而且不断有新的模型、方法、工具被提出来。但本质上，无论使用何种模型、方法、工具以及它们的组合，运用软件工程的唯一目的是提高软件本身的质量。

1.2 软件开发过程模型

经典的软件工程思想将软件开发分为 5 个基本阶段，即需求分析、系统设计、系统实现、测试和维护，如图 1-4 所示。采用软件生命周期来划分软件的工程化开发，使得软件开发能够分阶段依次进行。

图 1-4 软件生命周期的 5 个阶段

一直以来软件开发的过程都是软件工程领域研究的重点，这是基于一个普遍认可的原理："好的过程决定好的结果"，因此人们设计出各种开发工具以及过程模型来反映软件生命周期内各种各个阶段的衔接和组织管理。软件开发过程模型将抽象的软件工程思想具体化，它是在不断地软件开发实践中总结出来的实施于过程模型的软件开发工具、方法和步骤。总的来说，软件开发过程模型是跨越整个软件生命周期的系统开发、运作、维护所实施的全部过程、活动和任务的结构框架。软件开发过程模型有很多，以下分别介绍三种基本的软件开发过程模型：线性模型、增量模型和螺旋模型。

1.2.1 线性模型

线性模型即瀑布模型，又称生存周期模型，由温斯顿·罗伊斯（Winston Royce）在 1970 年提出。其核心思想是采用结构化的分析与设计方法将软件过程工序化，将功能的设计和实现分开，便于分工协作。线性模型将软件生命周期划分为软件计划、需求分析和定义、软件设计、软件实现、软件测试、软件运行和维护这 6 个阶段，规定了它们自上而下、相互衔接的固定次序，如同瀑布流水逐级下落。采用线性模型的软件开发过程如图 1-5 所示。

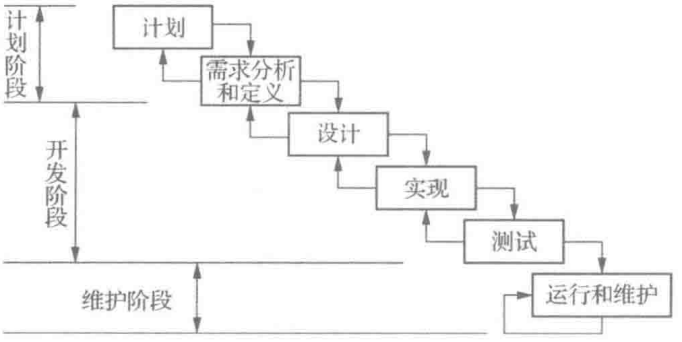

图 1-5　线性模型

线性模型是最早出现并唯一被广泛采用的软件开发过程模型，它提供了软件开发的基本框架，在软件工程中占有重要的地位。线性模型的本质是一次通过所有活动工序，最后得到软件产品。其特点是：

(1) 前一项活动的工作成果作为后一项活动的输入。

(2) 以这个输入为工作依据实施该项活动应完成的内容。

(3) 给出该项活动的工作成果，并作为输出传给下一项活动。

(4) 评审该项活动的实施，若评审通过，则继续下一项活动，否则返回之前的活动。

线性模型有利于软件开发过程中人员的组织及管理，有利于软件开发方法和工具的研究与使用，从而提高了软件项目开发的质量和效率。然而软件开发的实践表明，这种无法回溯的传统线性模型过于理想化，存在如下一些缺陷：

(1) 由于开发模型呈线性，所以当开发成果尚未经过测试时，用户无法得到直观结果。软件与用户接触时间间隔较长，增加了一定的风险。

(2) 软件开发前期未发现的错误可能扩散到开发后期的活动之中，因此软件项目开发可能会产生各种隐患。

(3) 复杂系统的软件需求分析阶段常常不能全面确定用户的所有需求，线性模型无法解决这一问题。

1.2.2　增量模型

增量模型是一种演化模型，其融合了线性模型的基本成分(重复应用)和原型实现模型的迭代特征，采用随着日程时间的进展而交错的线性序列，每一个线性序列产生软件的一个可发布的“增量”。当使用增量模型时，第一个增量往往是核心的产品，实现基本的需求，但很多补充的特征还没有发布。客户对每一个增量的使用和评估都作为下一个增量发布的新特征和功能，这个过程在每一个增量发布后不断重复，直到产生最终的完善产品。采用增量模型的软件开发过程如图 1-6 所示。

增量模型与原型实现模型和其他演化方法一样，本质上是迭代的，但与原型实现模型不一样的是，其强调每一个增量均发布一个可操作产品。例如，某一个采用增量模型开发的图形处理软件，在第一个增量中提供基本的图形编辑、管理和文档生成等功能，在第二个增量中提供复杂的图形编辑和管理功能，在第三个增量中提供扩展工具功能，在第四个增量中提

供高级的图层设计与排版功能，而任何增量的过程都可能使用原型实现模型。

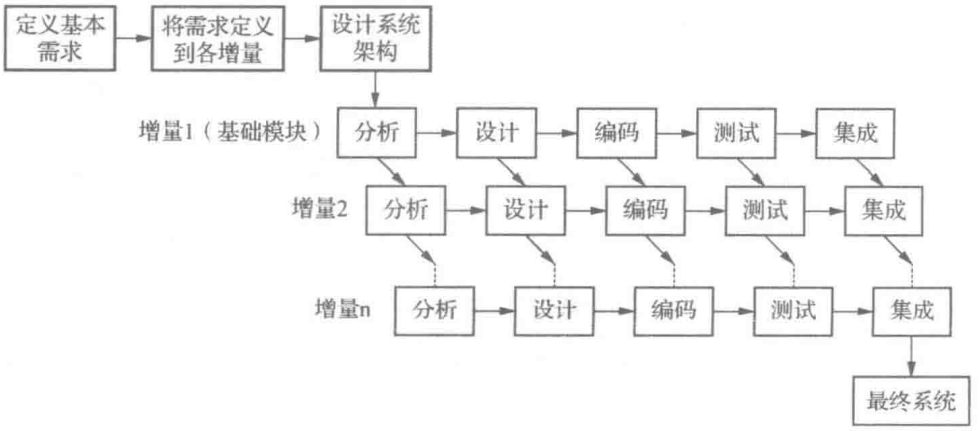

图 1-6　增量模型

增量模型的优点是灵活性高，如果项目既定期限前人力资源不足，增量模型就会特别有用。它提供了一种先推出核心产品的途径，如果核心产品口碑很好，则可增加人力实现下一个增量。这样即可先发布部分功能给客户，不至于造成项目过分延期，也能够有计划地避免和管理技术风险。但是，增量模型也存在一定的缺陷：

(1) 如果增量包之间存在相交的情况，则需要具备开放式的体系结构，做全盘系统分析，必须不破坏已构造好的系统部分。

(2) 这种模型将功能细化后适用于需求经常改变的软件开发过程，但容易退化成边做边改模型，从而失去整体性。

1.2.3　螺旋模型

螺旋模型由 Barry Boehm 在 1988 年提出，是一种演进式软件开发过程模型。整个模型紧密围绕开发中的风险分析，强调持续的判断、确定和修改用户的任务目标，并按成本、效益来分析候选的软件产品对任务目标的贡献。其将软件开发过程组成为一个逐步细化的螺旋周期，每经历一个周期，系统就得到进一步的细化和完善，推动着软件设计向深层扩展和求精。

螺旋模型通常用来指导大型软件项目的开发。图 1-7 显示了螺旋模型的原理，沿着螺旋线旋转，笛卡儿坐标系的四个象限分别表达以下四类活动：

(1) 制订计划：决定软件目标，选定实施方案并明确项目开发的限制条件。

(2) 风险分析：分析评估方案，识别和消除风险。

(3) 实施开发：实施软件开发和验证。

(4) 客户评估：评价软件功能，提出修正建议并制定下一步计划。

从图 1-7 中可以看到，沿着螺旋线每转一圈，表示开发出一个更完善的新版本的软件。多数情况下沿着螺旋线继续下去并向外逐步延伸，最终会得到满意的软件产品。

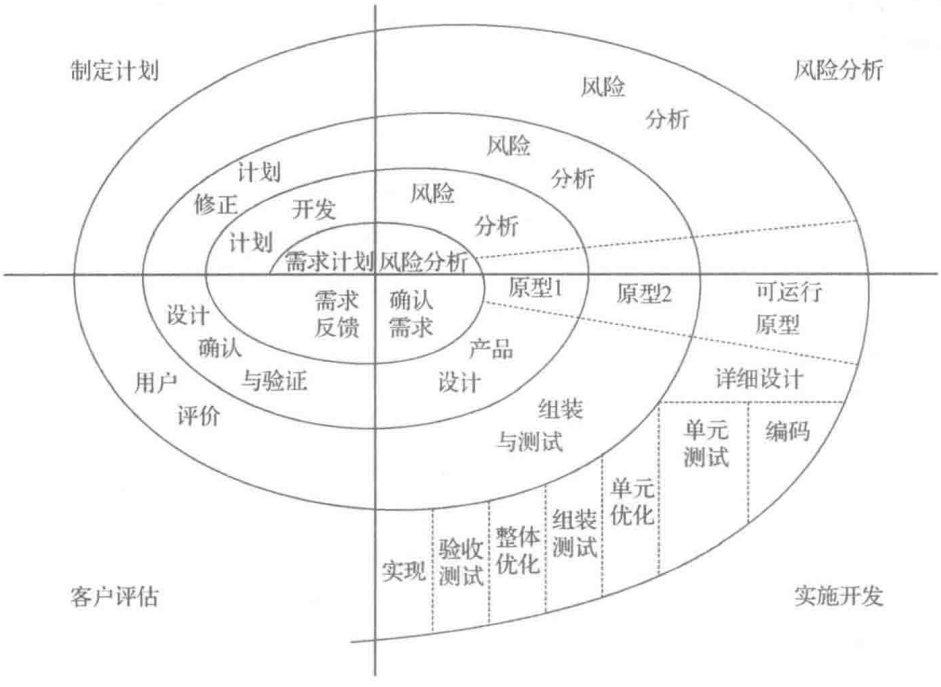

图 1-7　螺旋模型

与线性模型相比，螺旋模型支持用户需求的动态变化，方便用户参与软件开发的所有关键决策，有助于提高目标软件的适应能力；方便项目管理人员及时调整管理决策，从而降低了软件开发风险。

但是，我们不能说螺旋模型绝对比其他模型优越，事实上，这种模型也有如下问题需要解决：

(1) 采用螺旋模型需要具备相当丰富的风险评估经验和专门知识，在风险较大的项目开发中，如果未能够及时识别风险，势必造成重大损失。

(2) 螺旋模型不适用于合同项目的开发模型，因为必须在签订合同前考虑清楚开发项目的风险，因风险分析导致中途停止开发就会造成一定的经济损失。

(3) 过多的迭代次数会增加开发成本，延迟提交时间。

在开发过程中，项目经理需要根据项目实际情况和开发人员情况，灵活选择、组合不同的开发过程模型，甚至可以发明新的开发模型，以提高开发效率，保证软件质量，切忌教条主义。

1.3　软件系统分析与设计的重要性

需求分析阶段中分析人员根据计划阶段产生的可行性研究报告以及项目开发计划等，从系统的角度理解软件并确定目标系统的综合需求，提出这些需求的实现条件以及需求应达到的标准，即确定软件系统“做什么”。设计阶段中设计人员根据需求分析的结果，对整个软件系统进行整体和详细设计，具体包括算法设计、系统框架设计、数据库设计等，即确定软件系统“怎么做”。

分析和设计在软件工程中的重要性体现在其发生在软件系统的定义阶段。其中需求分析阶段为整个软件开发项目的顺利进行和成功打下良好的基础，而设计阶段则为软件程序的编写打下良好的基础。这两个阶段若做好，可以从根本上减少整个软件开发过程中耗费的时间及相应的开发成本；反之，如果对系统的需求分析阶段不重视，所开发的软件无法准确反映用户的需求甚至发生错误，所带来的损失将是不可估量的，而设计阶段的失误通常需要开发人员付出大量时间和成本进行弥补，甚至会导致项目进入“死亡行军”，最终不得不完全推翻重新进行。

1.4 软件开发方法

从20世纪60年代的手编程序到70年代的结构化分析和设计方法，到80年代CASE工具和环境的研制，再到20世纪90年代至今的软件复用和构件框架技术的广泛应用，越来越多的实际开发方法被世人提出。方法是一种把人的思维和行动结构化的明确方式，软件工程的方法解决了开发软件在技术上需要“如何做”的问题。目前主流的软件开发方法有两类：结构化方法和面向对象方法。

1.4.1 结构化方法

结构是指系统内各组成要素之间相互联系、相互作用的框架。结构化方法也称面向过程的方法或传统软件工程开发方法，由E. Yourdon和L. L. Constantine于1978年提出，其特点是自顶向下地分析与设计，逐步求精，在获取完整的需求之后实施开发，建立系统并测试部署。结构化方法强调系统结构的合理性以及所开发的软件的结构合理性，因此提出了一组提高软件结构合理性的准则，如分解和抽象、模块独立性、信息隐蔽等。针对不同的开发活动，有结构化分析、结构化设计、结构化编程和结构化测试等方法。面向数据流的方法是结构化方法家族中的一员，它具有明显的结构化特征。

1）结构化分析

结构化分析的基本步骤如下：

(1) 分析用户当前需求，创建实体-关系图并据此做出反映当前物理模型的数据流图。

(2) 推导出等价的逻辑模型的数据流图。

(3) 设计新的逻辑系统，生成数据字典和基元描述。

(4) 建立人机接口界面，提出可供选择的目标系统的物理模型数据流图。

(5) 确定各种方案的成本和风险等级，据此对各种方案进行分析。

(6) 选择一种方案。

(7) 建立完整的需求规约。

2）结构化设计

结构化设计给出一组帮助设计人员在模块层次上区分设计质量的原理与技术，通常和结构化分析衔接起来使用，以数据流图为基础得到软件模块结构。结构化设计方法尤其适用于变换型结构和事务型结构的目标系统。在设计过程中，它从整个程序的结构出发，利用

模块结构图表述程序模块之间的关系。

结构化设计的步骤如下：

(1) 评审和细化数据流图。

(2) 确定数据流图的类型。

(3) 把数据流图映射到软件模块结构，设计出模块结构的上层。

(4) 基于数据流图逐步分解高层模块，设计中下层模块。

(5) 对模块结构进行优化，得到更为合理的软件结构。

(6) 描述模块接口。

1.4.2　面向对象方法

面向对象(OO)方法是当前计算机界关心的重点，已经深入到软件领域几乎所有分支，是软件开发方法的一次飞跃。面向对象方法认为：现实客观世界是由对象组成的，任何客观的事物和实体都是对象，复杂对象可以由简单对象组成；对象可以被归类、描述、组织、组合、创建和操纵；类可以派生出子类，继承能避免共同行为的重复；对象间通过消息传递进行联系。

面向对象方法包括面向对象分析(OOA)、面向对象设计(OOD)、面向对象编程(OOP)等。它是一种自底向上的归纳和自顶向下的分解相结合的方法，通过对象模型的建立，能够真正基于用户的需求进行软件开发，而且系统的可维护性大大改善。

自 20 世纪 80 年代开始，5 年之内面向对象方法迅速从 5 种发展到 50 种以上。比较著名的面向对象方法包括：

(1) Booch 方法：这是由 Booch 提出的面向对象开发方法。Booch 最先描述了面向对象的软件开发方法的基础问题，指出面向对象开发是一种根本不同于传统的功能分解的设计方法。面向对象的软件分解更接近人对客观事务的理解，而功能分解只能通过问题空间的转换来获得。布什(Booch)方法现今居于领导地位。

(2) OMT 方法：这是 1991 年由 James Rumbaugh 等人提出来的面向对象建模技术。该方法对真实世界的对象建模，然后围绕这些对象使用分析模型来进行独立于语言面向对象的建模和设计。该方法为大多数应用领域的软件开发提供了一种实际的、高效的保证，努力寻求一种问题求解的实际方法。

(3) Coad 方法：这是 1989 年由 Coad 和 Yourdon 提出的面向对象开发方法。该方法的主要优点是将多年来大系统开发的经验与面向对象概念有机结合，在对象、结构、属性和操作的认定方面提出了一套系统的原则。该方法完成了从需求角度进一步进行类和类层次结构的认定。

此外还有 OOSE 方法(由 Ivar Jacobson 提出)、雪莉-米勒方法(由 Shlaer 和 Mellor 提出)等。

面向对象方法是一种模型化设计的抽象方法，结构上具有良好的高内聚低耦合特性。采用面向对象技术设计和开发的软件系统更易于维护，在对系统进行修改时，能够产生较少的副作用。同时，面向对象技术提出了类、继承、封装、接口等概念，为对象的复用提供了良好的支持机制，因此采用面向对象技术对软件产品进行设计和开发，能有效地提高软件组织的开发效益。另外，面向对象方法与技术在需求分析、可维护性和可靠性这 3 个软件开发的

关键环节和质量指标上有了实质性的突破，基本解决了软件开发在这方面存在的严重问题，成为当前分析、设计和开发软件产品的首选范型。业界关于面向对象可视化建模的标准是适用统一建模语言(UML)。

统一建模语言(UML)独立于特定开发语言和开发过程，提供了了解建模语言的一个基本手段，它推动了 OO 工具市场的成长，并支持更高层的开发概念，如协同、框架、模式和构件，是最佳实践经验的集成。

运用面向对象技术进行软件系统的分析设计是本书的核心内容，其开发方法在后面章节会做详细介绍。

1.5 软件工程工具

这里所讨论的软件工程工具泛指软件开发全过程中使用的各种程序系统，具体来讲就是开发人员在系统分析、设计、编程、测试等过程中应用的一系列辅助软件工具。例如，在设计、分析阶段有 PSL/PSA、AIDES、SDL/PAD 等；在编程阶段有 BASIC 编译器、PASCAL 编译器等。

软件工具通常由工具、工具接口和工具用户接口 3 部分组成。工具通过工具接口与其他工具、操作系统或网络操作系统及通信接口、环境信息库接口等进行交互作用，当工具需要与用户进行交互作用时，则通过工具的用户接口来进行。软件工具能在软件开发各个阶段帮助开发者控制开发的复杂性，提高工作质量和效率。随着软件工程思想的日益深入，计算机辅助软件开发工具和开发环境得到越来越广泛的应用。

1.5.1 计算机辅助软件工程

计算机辅助软件工程(Computer-Aided Software Engineering，CASE)，原指支持管理信息系统开发的、由各种计算机辅助软件和工具组成的大型综合性软件开发环境，但随着各种工具和软件技术的产生、发展、完善和集成，其逐步由单纯的辅助开发工具环境转化为一种相对独立的方法论。注意，CASE 用作方法支持，其有效性需要通过集成达到。一般的 CASE 环境需要网络的支持，允许若干软件工程师在某个环境中同时使用相同或不同的软件工具相互通信协同工作。CASE 技术是软件技术发展的产物，它既起源于软件工具的发展，又起源于软件开发方法学的发展，同时还受到实际应用发展的驱动。CASE 环境的核心是软件工程信息库。CASE 工具及其环境的开发和使用是软件工业化的产物，已引起工业界的普遍重视，也成为软件工程学科的一个重要研究领域。

1.5.2 CASE 工具的分类

支持软件工程活动的软件工具品种多、数量大，都是在软件工程信息库的支持下工作的。可以根据软件工具的功能、作用、使用方式等对其进行分类。一般来说，按照 CASE 工具的功能，可将其划分为以下 9 类。

1) 事务系统规划工具(Business System Planning Tools)

这类工具为定制事务信息系统规划提供元模型。利用元模型可以生成专用事务信息系