责任编辑：付寒冰
封面设计：黄燕美

软件工程

RUANJIAN GONGCHENG

扫码关注

“北京希望电子出版社”微信公众号
微信公众号回复8775，获取更多资源

微信公众号

扫码下载资料包

北京希望电子出版社网址：www.bhp.com.cn
电话：010-82626270
投稿：xiaohuijun@bhp.com.cn

ISBN 978-7-83002-877-0

定价：58.00元

软件工程

主编 王菊

北京希望电子出版社

软件工程

主编 王菊

RUANJIAN GONGCHENG

北京希望电子出版社
Beijing Hope Electronic Press
www.bhp.com.cn

CX-8775

软件工程

主　编　王　菊

副主编　罗雨滋

北京希望电子出版社
Beijing Hope Electronic Press
www.bhp.com.cn

内 容 简 介

本书全面深入地介绍了软件工程的基本概念、原理以及典型方法。全书共分为 9 个模块，涵盖软件工程概述、可行性分析与需求分析、软件设计、软件编码与实现、面向对象方法、软件测试、软件维护、软件管理及软件工程标准与文档编制等内容。

本书结构严谨，内容翔实，并提供了最新的国家软件工程标准的文档编制指南，为读者在软件开发各个阶段编写文档提供了有效指导。每个模块都附有丰富的习题，便于学生思考与巩固所学知识。

本书既可作为计算机相关专业的教材，也可作为从事软件分析、设计与开发、维护人员的参考书。

图书在版编目（CIP）数据

软件工程 / 王菊主编. -- 北京 : 北京希望电子出版社, 2024. 8. -- ISBN 978-7-83002-877-0

Ⅰ. TP311.5

中国国家版本馆 CIP 数据核字第 20247EF563 号

出版：北京希望电子出版社
地址：北京市海淀区中关村大街 22 号
中科大厦 A 座 10 层
邮编：100190
网址：www.bhp.com.cn
电话：010-82620818（总机）转发行部
010-82626237（邮购）
经销：各地新华书店

封面：黄燕美
编辑：付寒冰
校对：毕明燕
开本：787 mm × 1 092 mm 1/16
印张：16.5
字数：391 千字
印刷：三河市骏杰印刷有限公司
版次：2024 年 9 月 1 版 1 次印刷

定价：58.00 元

前言 PREFACE

在信息技术迅猛发展的今天，软件已成为社会运行的重要部分。从智能手机到企业系统，软件的应用广泛而深入，对日常生活和工作产生了革命性的影响。随着技术的进步，软件系统的复杂性日益增加，这就要求开发人员、项目管理者及维护团队具备更高的技能和更系统的管理方法。在这种背景下，软件工程作为一门科学被提出和发展，目的是通过系统化、规范化的方法来提高软件的质量和开发效率。

本书全面介绍了软件工程的各个方面，从软件开发的初始阶段到项目的最终维护，覆盖了软件生命周期中的每一个关键环节。本书力求让读者了解软件工程的基本理论，并结合具体的实例和案例分析，帮助读者把握这些理论在实际中的应用。

本书共分为 9 个模块，每个模块均围绕软件工程的一个重要领域展开。模块 1 为软件工程概述，介绍了软件工程的基本概念、历史背景以及软件开发的基本模型和方法。模块 2 ～ 7 依次涉及软件可行性与需求分析、软件设计、软件编码与实现、面向对象方法、软件测试、软件维护等关键阶段。模块 8 重点介绍了软件工程管理的内容，包括质量保证、项目计划和能力成熟度模型等，这些都是确保项目成功的关键因素。模块 9 则专注于软件工程的标准和文档编写，强调标准化过程和良好文档对于项目管理和产品维护的重要性。

本书融入了工程教育认证标准的理念，旨在培养学生解决复杂工程问题的能力，包括问题分析、研究、设计开发、使用现代工具、可持续发展、项目管理等方面，适合作为计算机相关专业的教材，也可作为软件开发、维护人员提升工作技能的学习参考书。

考虑到当今软件技术的快速发展和应用的广泛性，本书在技术时效性、行业趋势、项目管理工作等方面存在一定欠缺。建议读者结合最新的行业报告、研讨会和社区论坛，以获得最前沿的软件工程知识和技能。

由于作者水平有限，不足之处在所难免，敬请广大读者批评指正。

编　者

2024 年 5 月

目录 CONTENTS

模块 1

软件工程概述

学习目标

- 理解软件、软件工程的概念。
- 了解软件工程及产品的周期、流程的特点和内容，理解其中涉及的工程管理问题。
- 理解软件生存周期模型，包括快速原型法、螺旋模型、构件组装模型等。
- 了解软件工程的开发方法、软件工具与集成化开发环境。

1.1 软件工程的背景

软件工程的产生背景与计算机的普及和软件的发展密切相关。随着计算机技术的不断发展，软件变得越来越复杂，出现了所谓的“软件危机”，即软件项目经常超出预算、延期交付或无法满足用户需求的问题，因而人们需要一种新的方法来系统地开发和维护软件。通过引入工程化的方法和管理原则，使软件开发过程更加可控和高效，这就是软件工程。

要学习软件工程，必须先了解软件的概念和特点，以及出现“软件危机”的原因。

1.1.1 软件的定义与特点

1. 软件的定义

一个完整的计算机系统由两部分组成：硬件（hardware）和软件（software）。

硬件是一系列电子、机械和光电元件组成的各种物理装置的总称，是计算机系统运行的物质基础。软件是计算机系统中程序、数据及其相关文档的总称。其中，程序是按事先设计的功能和性能要求编写的指令序列，是软件的重要组成部分，也是软件的主要表现形式；数据是描述程序的处理对象；文档是与程序开发、维护和使用有关的图文材料，是对软件开发和维护全过程的书面描述和记录。

计算机系统的硬件和软件互相依存，硬件是物质基础，软件控制硬件的运行，指挥硬件完成各种计算和处理各种事务，它们相互配合、共同完成人们预先设定好的操作，成为人们日常工作和生活的得力助手。

2. 软件的特点

从操作系统到应用程序，从专业工具到游戏，都是计算机软件，可以说软件已渗透到人们工作生活中的方方面面。计算机软件种类繁多，功能各异。与硬件相比，计算机软件具有以下特点：

- 复杂性：软件系统的复杂性来源于其功能的多样性和实现的复杂性。一个大型的软件系统可能包含数百万行代码，涉及多种编程语言和技术路线。
- 无形性：与硬件不同，软件是一种无形的产品。它没有具体的物理形态，但可以通过存储介质如光盘、硬盘或网络传输。
- 可复制性：软件的复制成本极低，一旦开发完成，就可以无限复制而不会降低质量。这种特性使得软件可以大规模分发，但同时也要求开发者采取措施保护知识产权，防止非法复制和盗版。
- 可变性：软件可以根据用户的需求进行修改和更新，用户也可以根据自己的需求和

偏好来定制软件的功能和界面。许多软件提供个性化设置，允许用户调整操作方式、界面布局和功能选项，这使得软件具有很高的灵活性，可以获得更佳的使用体验。

- 依赖性：软件功能的发挥通常依赖于特定的硬件和操作系统环境，不同的软件对硬件的需求不同。如果没有硬件的支持，软件将失去实用价值。
- 应用性：软件的价值在于应用。软件的设计和开发都是为了实现特定的功能，以满足用户的不同需求。无论是文字处理、图像编辑还是数据分析，每种软件都有其独特的功能集合。应用性是软件价值的核心体现。

3. 软件的分类

软件种类繁多，人们可以根据不同的需要选择使用不同的软件。按不同的分类标准，软件可分为不同的类型。

按软件的功能划分，软件可分为系统软件、支撑软件、应用软件等。

按软件的规模划分，软件可分为微型、小型、中型、大型、超大型软件等。

按软件的工作方式划分，软件可分为实时软件、分时软件、交互式软件、批处理软件等。

按软件服务对象的范围划分，软件可分为项目软件和产品软件。

1.1.2　软件的发展历程与软件危机

1. 软件发展的 4 个阶段

自 1946 年世界上的第一台计算机 ENIAC 诞生以后，就有了程序的概念，因此可以认为，程序是软件的前身。经历了七十多年的发展，人们对软件有了更为深刻的认识。在这几十年中，计算机软件的发展大致经历了 4 个阶段，如表 1-1 所示。

表 1-1　软件发展的阶段

	程序设计阶段	程序系统阶段	软件工程阶段	现代软件工程阶段
年代	20 世纪 50 年代至 60 年代中期	20 世纪 60 年代中期至 70 年代中期	20 世纪 70 年代中期至 80 年代中期	20 世纪 80 年代中期至今
软件的范畴	程序	程序及说明书	产品软件（项目软件）	项目工程
主要程序设计语言	汇编及机器语言	高级语言	高级程序设计语言	面向对象可视化设计语言
软件工作范围	程序编写	包括设计和测试	软件生存期	整个软件生存期
需求者	程序设计者本人	少数用户	市场用户	面向所有用户

2. 人们对软件的认识

在计算机诞生的初期，软件多以简单的代码和命令形式存在，功能有限，用户群体极小。当时的人们普遍认为软件是专业人士的专属工具，与日常生活关系不大。然而，随着

计算机硬件的发展，个人计算机的普及使用以及图形界面的出现，尤其是互联网的兴起，使软件的发展进入了一个新的阶段。软件不再仅仅是单机程序，而是开始向网络服务转变。这一时期，人们对软件的认识也逐渐深化，开始意识到软件的强大潜力。软件不再是单一的工具，而是一个连接人与信息的桥梁。

21 世纪初，智能手机和平板电脑的普及，使软件变得更加个性化、社交化和移动化。人们对软件的认识也随之升级，软件被视为是提升工作效率、丰富娱乐生活的重要手段。

近年来，人工智能和大数据技术的兴起，使得软件发展进入了智能化阶段。软件不仅能够执行命令，还能够学习和适应用户的行为，提供更加精准的服务。人们对软件的认识进一步拓展，开始将其视为解决问题的伙伴，甚至是决策的参谋。在这个过程中，人们对软件的认识也在不断深化。从最初的专属工具到复杂系统，再到智能化服务，软件的角色发生了根本性的转变。软件不仅仅是执行任务的程序，它还蕴含着数据、知识和智慧。软件的价值不仅仅在于其功能，更在于其对数据的处理能力和对用户需求的理解能力。

3. 软件危机

（1）软件危机的背景

在软件技术发展的第二阶段，随着计算机硬件技术的进步，计算机的存储容量、速度和可靠性有了明显提高，硬件成本的降低为计算机的广泛应用创造了极好的条件。在这一形势下，软件的需求量不断增加，软件的规模也越来越大，复杂度也不断增加，然而软件开发技术的进步却一直未能满足形势发展所提出的要求，导致在软件开发中遇到的问题找不到解决的方法，致使问题积累起来，形成了日益尖锐的矛盾，导致“软件危机”的出现。

“软件危机”是一种现象，是指在计算机软件的开发和维护过程中出现的一系列严重问题的现象。这些问题反映在软件可靠性没有保障、软件维护的工作量大、费用不断上升、进度无法预测、成本增长无法控制、程序人员无限度增加等各个方面，以至于形成人们难以有效控制软件开发的局面。

（2）软件危机的主要表现

具体来说，软件危机主要有以下表现：

① 软件开发进度难以预测，拖延工期几个月甚至几年的现象并不罕见。这种情况会导致软件开发的投资者或软件的用户对软件开发组织的工作既不满意，也不信任，从而大大降低软件开发组织的信誉。

② 软件开发成本难以控制，常常导致投资一再追加，实际成本甚至比预算成本高出一个数量级。

③ 软件开发人员与用户之间沟通不畅。软件开发人员不能真正了解用户的需求，而用户也不了解计算机求解问题的模式与能力，双方无法用共同熟悉的语言进行交流和描述，导致双方沟通不顺畅。双方对需求的理解出现偏差，可能导致软件无法满足用户的需求。

④ 软件产品的质量无法保证，系统中的错误难以完全消除。另外，软件是一种无形

的逻辑产品，质量问题难以用统一的标准度量，因而造成质量控制困难。

⑤ 软件产品难以维护。软件产品本质上是开发人员的代码化的逻辑思维活动，他人难以替代，除非是开发者本人，否则难以及时检测、排除系统故障。为使系统适应新的操作环境，维护人员往往需要在原系统中增加一些新的功能，这样又可能产生新的错误，使得产品更难以维护。

⑥ 文档资料不齐全，也不合格。计算机软件不仅仅是程序，还应该有一整套文档资料。这些文档资料应该是在软件开发过程中产生出来的，而且应该是“最新的”（即和程序代码完全一致的）。软件的文档资料是软件必不可少的重要组成部分，缺乏必要的文档资料或者文档资料不合格，将给软件开发和维护带来许多严重的困难与问题。

（3）软件危机的产生原因

软件危机的产生是由软件本身的特点以及开发软件的方式、方法、技术和人员引起的。产生的原因主要包括以下几点：

① 软件规模越来越大，结构越来越复杂。随着计算机的普及，应用范围不断扩大，需要开发的软件越来越多，软件结构也越来越复杂。现实问题的复杂性，加上用户和市场对软件产品的需求往往带有不切实际的期望，导致了软件开发项目的目标过于宏大和复杂。开发者为了满足这些期望，可能会采取过度复杂的设计，增加了软件的不稳定性和维护难度。

② 用户需求不明或需求变更频繁。在软件开发出来之前，用户自己不清楚软件开发的具体需求，或者对需求的描述不精确，可能有遗漏、二义性甚至有误；开发人员对用户需求的理解与用户的愿望有差异；项目在开发过程中，用户需求往往会发生变更。频繁的需求变更会使得项目难以控制，增加开发的难度和成本。同时，不断的变更也会影响到软件的稳定性和可维护性。

③ 技术复杂性与开发人员的技能不足。随着技术的发展，软件系统变得越来越复杂。这种复杂性不仅来自于软件本身的功能需求，还包括与其他系统的集成、多种技术的融合等。因此需要开发者具备多方面的知识和技能。软件开发是一个高度专业化的工作，然而，现实中很多项目因为缺乏经验丰富的开发人员而陷入困境。开发新手可能由于技能的不足及缺乏对软件工程原则的理解，导致开发出的软件质量不高。复杂的技术环境使得开发和维护工作变得更加困难。

④ 缺乏标准化。软件开发缺乏统一的标准和规范，不同的团队可能采用不同的开发方法和工具。这种缺乏一致性的做法导致了软件质量和兼容性的问题。

⑤ 开发技术与开发工具落后。尽管软件开发工具比 30 年前已经有了很大的进步，但直到今天，软件开发仍然离不开工程人员的个人创作与手工操作，软件生产仍不可能像硬件生产那样达到完全的自动化，因此软件生产效率低，且软件质量也难以保证。

⑥ 项目管理不当。软件项目的管理不善是导致软件危机的一个重要原因。缺乏有效的项目管理方法、工具和经验，会导致项目计划不准确、进度控制不力、资源分配不合理等。此外，项目风险评估不足，也会导致问题的产生。

4. 消除软件危机的途径

软件危机产生的原因是多方面的，涉及项目管理、技术复杂性、人员技能、沟通协作等多个层面。要解决软件危机，需要提高项目管理能力，加强技术培训，改善沟通机制，稳定需求，推广标准化方法，等等。综合起来，主要是从两方面入手：一是管理方面，二是技术方面。只有通过综合施策，才能提高软件开发的效率和质量，减少软件危机的发生。

从管理方面看，必须充分认识到软件开发不是某种个体劳动的神秘技巧，而应该是一种组织良好、管理严密、各类人员协同配合、共同完成的工程项目；必须充分吸取和借鉴人类长期以来从事各种工程项目所积累的行之有效的原理、技术和方法。

从技术方面看，应该开发和使用更好的工具软件。正如机械工具可以“放大”人类的体力一样，软件工具可以“放大”人类的智力。在软件开发的每个阶段都有许多烦琐重复的工作需要做，在适当的软件工具辅助下，开发人员可以把这类工作做得既快又好。

总之，为了消除软件危机，既要有技术措施（方法和工具），又要有必要的组织管理措施，这就是软件工程（software engineering）研究的主要内容。

1.1.3 软件工程

为了更有效地开发和维护软件，消除软件危机，1968 年，在由北大西洋公约组织（NATO）召开的国际会议上提出了“软件工程”的概念，借助工程化的方法和管理手段来开发软件，从此，诞生了一门新兴的学科——软件工程。软件工程即是从管理和技术两方面研究如何更好地开发和维护计算机软件的一门学科。

1. 什么是软件工程

软件工程的概念提出已经有 50 多年，但直到目前为止，软件工程概念的定义并没有统一。不同的学者、组织机构都分别给出了自己认可的定义，例如：

- 1968 年的会议上给出的定义是：“软件工程是为了经济地获得可靠的且能在实际机器上有效运行的软件而建立和使用的完善的工程化原则。”
- 1983 年，美国电气电子工程师学会（IEEE）给出的定义是：“软件工程是开发、运行、维护和修复软件的系统方法。”
- 1993 年，IEEE 给出的定义是：“①把系统化的、规范化的、可量化的方法应用于软件开发、运行和维护中，即将工程化方法应用于软件；②对于①中所述方法的研究。”
- 国家标准《信息技术软件工程术语》（GB/T 11457—2006）中的定义为：应用计算机科学理论和技术以及工程管理原则和方法，按预算和进度，实现满足用户要求的软件产品的定义、开发、发布和维护的工程或进行研究的学科。

概括地说，软件工程是一门指导计算机软件开发和维护的工程学科。它采用工程的概念、原理、技术和方法来开发与维护软件，把管理技术和开发技术有效地结合起来，以便经济地开发出高质量的软件并有效地维护它。

软件工程提出了一系列的概念、方法、原理以及开发模型，其研究的核心问题是在给

定的成本和进度前提下，使用什么方法开发软件能获得可使用、可行性好、易于维护、成本合适的软件。

软件工程是一门综合性的交叉学科，它涉及计算机科学、数学、工程科学和管理科学等领域，综合利用了这些学科的很多理论和知识，以求高效地开发高质量的软件。

软件工程的知识结构主要有三个：计算机科学和数学、工程科学、管理科学。其中，计算机科学和数学用于构造算法，工程科学用于指定规范、设计泛型、评估成本及确定权衡，管理科学用于计划、资源、质量和成本的管理。

现代软件工程是指采用项目化的思想，利用现代化的分析、开发、测试等辅助工具来实现软件的工程活动。

2. 软件工程的目标

软件工程是一门工程性学科，目的是成功开发出用户需要的软件产品。软件工程的目标往往是基于软件项目目标的成功实现而提出的，主要体现在以下几个方面：

- 付出较低的开发成本。
- 达到要求的软件功能。
- 取得较好的软件性能。
- 软件的可靠性较高。
- 软件易于使用、维护和移植。
- 能够按时完成开发任务，并及时交付使用。

在实际开发中，企图让以上几个质量目标同时达到理想的程度往往是不太现实的，如低开发成本与高可靠性、易于维护和移植的目标是彼此冲突的。因此，要根据软件项目的实际要求做目标的合理取舍，找到一个相对平衡的目标方案。

3. 软件工程的基本原则

软件工程的基本原理是指导软件开发与维护的最高准则和规范。为了确保软件质量和开发效率，必须遵守一些基本的原则。

（1）严格管理

在软件设计、开发、使用与维护的漫长的生命周期中，需要完成许多性质各异的工作。因此应该把软件生命周期划分成若干个阶段，并相应地制订出切实可行的计划，然后严格按照计划对软件的开发与维护工作进行管理。

（2）阶段评审

软件的质量保证工作不能等到编码阶段结束之后才进行。据统计，设计错误占软件错误的比例约 63%，编码错误仅占约 37%，错误发现与改正得越晚，所需付出的代价就越高。因此，在每个阶段都进行严格的评审，才能尽早发现在软件开发过程中所犯的错误，减少改正错误所需付出的代价。

（3）产品控制

一切有关修改软件的提议，必须严格按照规程进行评审，获得批准以后才能实施修改，绝不能谁想修改就可以修改。

（4）采用现代程序设计技术

程序设计尽可能采用当今成熟且先进的技术，以提高软件开发和维护的效率，同时也可以提高软件产品的质量。

（5）结果审查

软件产品不同于一般的物理产品，软件开发人员（或开发小组）的工作进展情况可见性差，难以准确度量，从而使得软件产品的开发过程更难以评价和管理。为了提高软件开发过程的可见性，更好地进行管理，应该根据软件开发项目的总目标及完成期限，规定开发组织的责任和产品标准，从而使得到的结果能够被清楚地审查。

（6）人员应该少而精

软件开发小组的组成人员的素质应该高，且人数不宜过多。合理安排人员可以减少开发成本并提高工作效率。

（7）不断改进软件工程实践

不断总结经验，勇于尝试新的软件技术和工具，以提高工作效率和产品质量。

4. 软件工程的内容

软件工程是一门新兴的交叉学科，涉及的学科多，研究的范围广。归结起来，软件工程的主要研究内容有以下几方面：方法与技术、工具与环境、软件工程管理、标准与规范。

方法与技术主要研究软件开发的各种方法与工作模型、软件开发过程及具体实现技术。

工具与环境主要指软件开发工具与开发环境，包括计算机辅助软件工程。

软件工程管理是指对软件工程全过程的控制和管理，包括计划安排、成本估算、项目管理、软件质量管理等。

标准与规范是指软件工程标准化与规范化，使得各项工作有章可循，以保证软件生产效率和软件质量。

1.2 软件的生命周期及其开发模型

软件开发是一个过程，为方便管理，借鉴工程领域中产品生存周期的概念，引入了软件生命周期的概念。

1.2.1 软件生命周期

软件生命周期（life cycle）也称软件生存周期，是指一个软件从提出开发要求开始，经过开发、交付使用，直到该软件报废为止的整个时期。在软件生存周期中，将软件的开发过程分为若干阶段，使得每个阶段有明确的任务，从而方便对规模大、结构和管理复杂的软件开发进行控制和管理。这对于软件生产的管理和进度控制有着非常重要的意义。

软件生命周期一般可分为三个主要阶段：定义（计划）阶段、开发阶段和运行维护阶

段。这三个阶段往往又可细分为一些子阶段。

1. 定义（计划）阶段

定义阶段可根据实际情况分为两个子阶段：软件计划和需求分析。

软件计划：这一阶段的主要任务是明确要解决的问题并确定工程的可行性，分析软件系统项目的主要目标和开发该系统的可行性，估计完成该项目所需的资源和成本，最后还要提出软件开发的进度安排，提交软件计划文档。软件计划又可细分为问题定义和可行性分析两个阶段。

需求分析：这一阶段的任务不是具体地解决问题，而是待开发软件的可行性分析与项目开发计划评审通过以后，进一步准确地理解用户的要求，确定软件系统必须“做什么”，并将其转成需求定义（确定软件系统必须具备哪些功能），提交软件的需求文档。

2. 开发阶段

开发阶段可分为三个子阶段：设计、编码和测试。

设计又可分为系统设计和详细设计两个阶段。系统设计是软件开发过程中的关键阶段，主要任务是将软件需求分解为系统架构及组件级别的设计，包括主要的模块、接口、数据流和数据存储。系统设计的目的是解决如何构建系统的问题，确保系统的各个部分能够协调工作，满足功能和非功能的需求，并为详细设计阶段奠定基础。详细设计是为每个模块完成的功能进行具体描述，即把功能描述转变为精确的过程描述。如模块的控制结构是怎样的，先做什么，后做什么，有什么样的条件判定等，并用相应的表示工具把这些控制结构表示出来。系统设计和详细设计都需要提交设计说明文档和测试文档。

编码就是按照设计说明选择合适的程序设计语言工具把每个模块的控制结构转换成计算机可接受的程序代码，即写成以某种特定程序设计语言表示的“源程序清单”。编码完成后会提交程序代码和相应文件。

测试就是根据测试计划对软件项目进行各种测试，其主要方式是使用测试用例检验软件的各个组成部分。测试的目的是发现软件中的问题，它是保证软件质量的重要手段。测试完成后需提交软件项目测试报告。

3. 运行维护阶段

运行维护阶段是软件生存周期中时间最长的阶段。软件经过评审确认后提交给用户使用，就进入了运行维护阶段。软件产品投入运行后，接下来的主要工作就是维护了。维护始终贯穿于软件运行期间，它可以持续几年甚至几十年，维护的过程漫长，维护的内容广泛（不仅要改正软件运行中出现的错误，还包括修改软件以适应环境的变化，根据用户需求扩充软件的功能等），因此，软件维护的工作量很大。

1.2.2　软件开发模型

根据软件生产工程化的需要，生存周期的划分也有不同，从而形成了不同的软件生存周期模型（life cycle model, LCM），或称软件开发模型（软件过程模型），它是描述软件

生产过程中各种活动如何执行的模型，是一种对软件过程的抽象表示。软件开发模型有多种，常用的有瀑布模型、快速原型法模型、螺旋模型、构件组装模型和智能模型等。

1. 瀑布模型

瀑布模型（waterfall model, WM）遵循软件生存周期的划分，明确规定每个阶段的任务，各个阶段的工作按顺序展开，恰如奔流不息逐级下落的瀑布，如图 1-1 所示。

瀑布模型把软件生存周期分为软件定义、软件开发、软件运行维护三个时期。这三个时期又可细分为若干阶段：软件定义可分为问题定义、可行性分析、需求分析三个阶段，软件开发可分为系统设计、详细设计、编码、测试等阶段，软件投入运行后进入运行维护阶段。

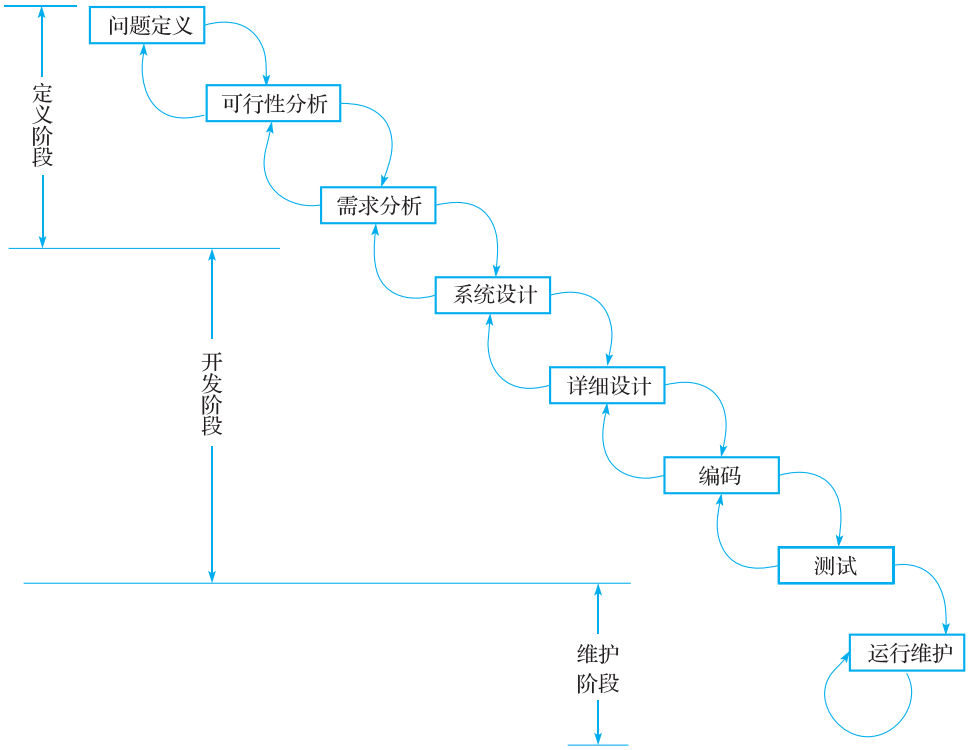

图 1-1　典型的瀑布模型

瀑布模型的优点如下：

- 清晰地提供了软件开发的全过程，明确了每一阶段的任务与目标。
- 阶段间的有序性和依赖性。有序性是指前一阶段的工作完成后，后一阶段的工作才能开始，有序地开展工作，避免了过程中的随意状态。依赖性指后一阶段的工作依赖于前一阶段的结果。
- 每个阶段都形成相应的文档，便于阶段性的进度管理。
- 质量保证。强调文档的作用，并对文档进行评审，强调尽早发现问题并及时改正；严格的计划性也会保证软件产品的按时交付。

瀑布模型的缺点如下：

- 缺乏灵活性，不能适应用户需求的不断变化。
- 如果大的错误在生存周期的后期才被发现，将可能导致灾难性的后果。
- 瀑布模型是一种以文档为驱动的模型，管理工作主要通过各种文档来反映和跟踪，大量规范性文档和严格的评审工作增加了项目的工作量。
- 应用范围有一定的局限性，主要适用于功能、性能明确且无重大改变、规模相对较小的软件开发。

2. 快速原型模型

原型是指一个具体的可执行模型，它实现了系统的若干功能。

快速原型模型（rapid prototype model, RPM）是指不断地运行系统“原型”来进行启发、揭示和判断的系统开发方法，也称为快速原型法。正确的需求定义是系统成功的关键，但是许多用户在开始时往往不能准确地叙述他们的需要，软件开发人员需要反复多次地和用户交流信息，才能全面、准确地了解用户的要求。当用户实际使用了目标系统以后，也常常会改变原来的某些想法，对系统提出新的需求。

快速原型法的主要思路如下：

- 根据用户的需求迅速构造一个低成本的用于演示及评价的试验系统（原型）。
- 由用户对原型进行评价。
- 在用户评价的基础上对原型进行修改或重构。

快速原型法的目标：用户对所用的原型满意。

快速原型模型的开发过程如图 1-2 所示。

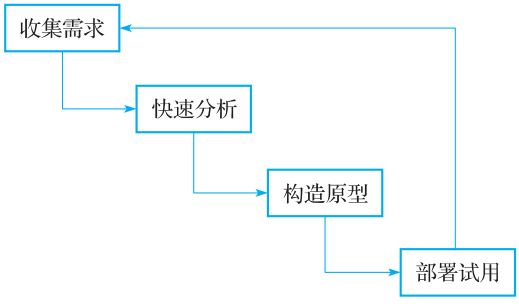

图 1-2　快速原型模型的开发过程

快速原型法有如下优势：

- 有直观的系统开发过程。
- 可以逐步明确用户需求。
- 用户参与系统开发的全过程，可以直接掌握系统的开发进度。
- 原型模型使用户接受程度高，系统更易维护。

快速原型法存在如下不足：

- 不适用于拥有大量计算或控制功能的系统。
- 不适用于大型或复杂的系统。

- 容易掩盖需求、分析、设计等方面的问题。
- 对资源规划和管理较为困难，随时更新文档也会带来麻烦。
- 对整体的考虑较少。

快速原型法的主要适用范围如下：

- 解决有不确定因素的问题。
- 对用户界面要求高的系统。
- 决策支持方面的应用。
- 中型系统。

快速原型法能尽早获得更正确、更完整的需求，可以减少测试的工作量，提高软件质量。当快速原型法使用得当时，它能减少软件开发的总成本，缩短开发周期，因此是目前比较流行的一种实用的开发模式。但是，快速原型法有其适用范围，对于嵌入式软件、实时控制软件、科技数值计算类软件以及大型的复杂系统来说，快速原型法可能并不适用。

3. 螺旋模型

螺旋模型（spiral model, SM）是一种引入了风险分析与规避机制的过程模型，它将瀑布模型与快速原型模型的迭代特征结合起来，并加入了两种模型都忽略的风险分析，弥补了两者的不足。螺旋模型是瀑布模型、快速原型模型和风险分析方法的有机结合，如图1-3所示。

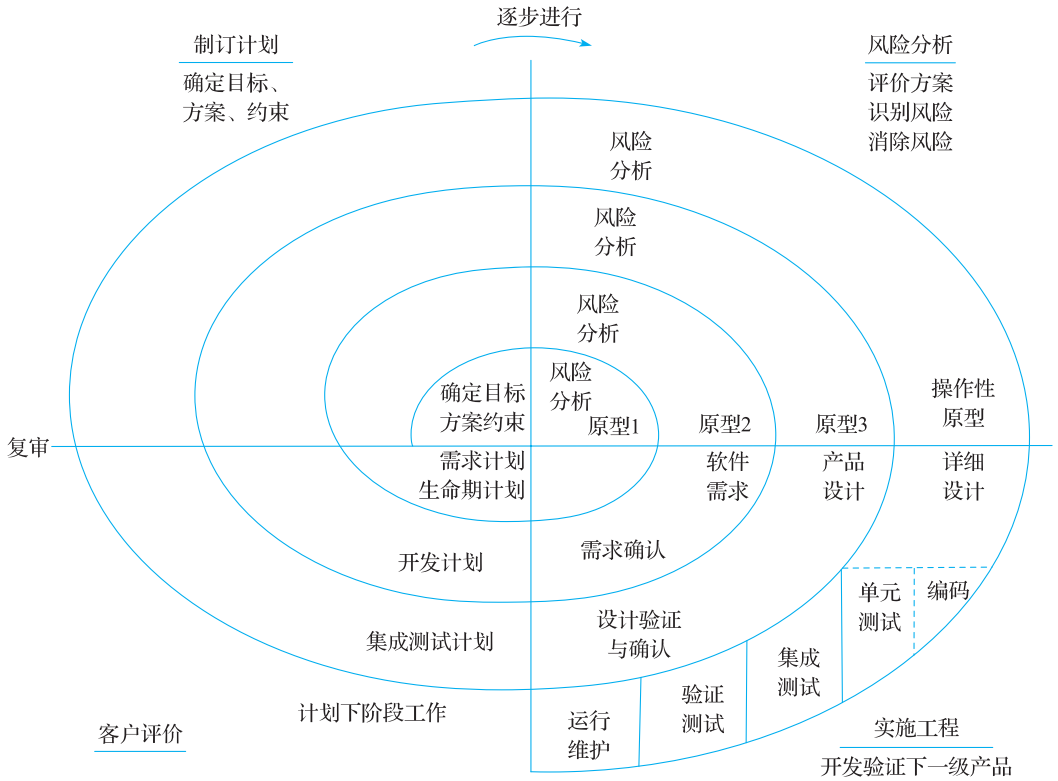

图 1-3　螺旋模型

螺旋模型用螺旋线表示软件项目的进展情况，螺旋线中的每个回路表示软件过程的一个阶段。最里面的回路与项目可行性有关，接下来的一个回路与软件需求定义有关，再下一个回路则与软件系统设计有关，以此类推。

螺旋模型将开发过程分为几个螺旋回路（螺旋周期），每个回路被分成以下 4 个工作步骤：

① 制定计划：确定项目阶段目标，选定实施方案，理清项目开发的限制或约束条件。

② 风险分析：分析所选方案，识别和消除风险，列出可能出现的问题及问题的严重程度，并估计可能产生的后果。

③ 实施工程：实施软件开发的过程。

④ 用户评估：评价开发工作，提出修改建议，并与开发人员一起讨论制订下一步的开发计划。

在螺旋模型中，沿螺旋线自内向外每旋转一圈便开发出一个更为完善的新的软件版本，自内向外逐步延伸，最终得到所期望的系统。

螺旋模型具有如下特点：

- 螺旋模型更适合人们认识事物的规律——由粗到细、由表及里、逐步深化。
- 系统开发的各个阶段可以回溯和重复，逐步发展，每一个螺旋周期都是对上一周期的深化和细化。
- 螺旋模型特别强调原型的可扩充性，原型的进化贯穿整个软件的生存周期。
- 螺旋模型是一种风险驱动模型，为项目管理人员及时调整决策方案提供了方便，降低了开发风险。

采用螺旋模型进行开发，需要有相当丰富的风险评估经验及专门知识。如果风险较大而又未能及时发现，则可能导致重大损失，因此，要求开发队伍具备较高的水平。

螺旋模型适用于内部的大规模软件的开发，而不太适合用于一般的合同软件的开发。

4. 构件组装模型

如今，大多数软件开发项目采用面向对象的方法，面向对象的软件开发过程将重点放在软件生存周期的定义阶段。这是因为在开发早期就定义了一系列面向问题域的对象，并在整个开发过程中统一使用这些对象，不断地充实和扩展对象模型。定义阶段得到的对象模型同样适用于设计和实现阶段，并在各个阶段都使用统一的概念和描述符号。因此，面向对象的软件开发过程具有以下特点：开发阶段的界限模糊，不同阶段之间没有明显的界限，使整个开发过程更为灵活；开发过程逐步求精，随着项目的进展逐步细化；开发活动反复迭代，不断改进和完善软件每次迭代都会增加或明确一些目标系统的性质，而不是对前期工作本质性的改动，这样才能减少不一致性，降低出错的可能性。

面向对象技术是将事物实体封装成包含数据和数据处理方法的对象，并将这些对象抽象为类。经过适当设计和实现的类或类的集合可以称为构件。由于构件具有一定的通用性，因而它们可以在不同的软件系统中被复用。在基于构件复用的软件开发中，软件由这些预定义的构件装配而成，就像使用标准零件组装汽车一样。

构件复用技术能带来更好的复用效果，并且具有良好的工程特性，更加适应软件按照工业流程生产的需要。构件组装模型（component assembly model, CAM）如图 1-4 所示。

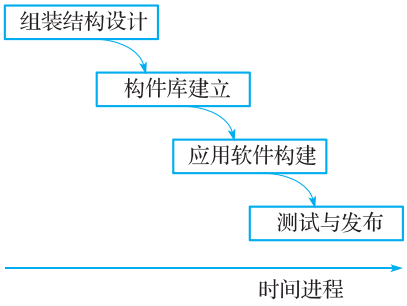

图 1-4　构件组装模型

构件组装模型有以下特点：

- 应用软件可由预先编好的、功能明确的产品构件定制而成，并可实现应用的扩展和更新。
- 利用模块化方法，将复杂的难以维护的系统分解为互相独立、协同工作的构件，并努力使这些构件可反复重用。
- 突破时间、空间及不同硬件设备的限制，利用客户和软件之间统一的接口实现跨平台的互操作。

构件组装模型的优点和缺点如下：

- 优点：由于构件的复用，减少了开发的工作量，缩短了软件开发周期，提高了软件的开发效率，并且提高了软件开发质量，降低了开发风险。
- 缺点：可重用性和软件高效性不易协调；需要精干的有经验的分析和开发人员，一般的开发人员可能不易上手；客户的满意度低。

构件组装模型主要用于面向对象开发方法中，因而是一种面向对象的软件过程模型；而瀑布模型、快速原型模型、螺旋模型等大都建立在传统的软件生存周期概念基础上，通常称它们为传统的软件过程模型。

5. 智能模型

智能模型（intelligent model, IM）也称为基于知识的软件开发模型，是知识工程与软件工程在开发模型上结合的产物，是以瀑布模型与专家系统的综合应用为基础建立的模型。该模型通过系统的知识和规则帮助设计者认识一个特定软件的需求和设计。这些专家系统已成为开发过程的伙伴，并对其有指导作用。智能模型如图 1-5 所示。

从图 1-5 中可以看到，智能模型与其他模型不同，它的维护并不在程序级别上进行，这样就大大降低了问题的复杂性。

智能模型的优点包括：

- 通过领域的专家系统，可使需求说明更加完整、准确和无二义性。
- 通过软件工程的专家系统提供一个设计库支持，在开发过程中成为设计者的助手。
- 通过对软件工程知识和特定应用领域的知识和规则的应用为开发提供帮助。

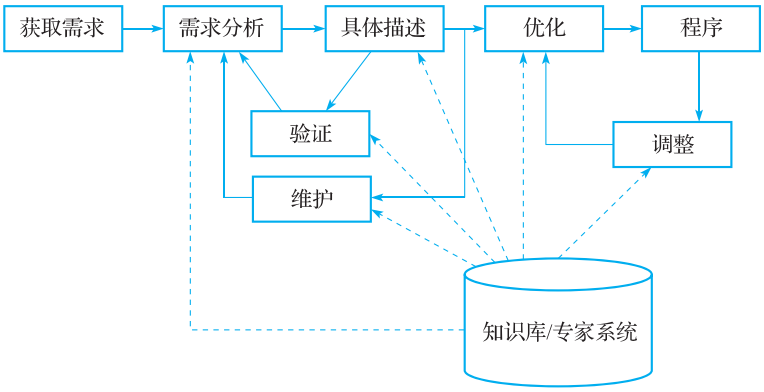

图 1-5　智能模型

建立适合于软件设计的专家系统，或建立一个既适合软件工程又适合应用领域的知识库，都是非常困难的。目前，在软件开发中已经开始应用人工智能（artificial intelligence, AI）技术，并取得了局部进展，如在计算机辅助软件工程（computer-aided software engineering, CASE）系统中使用专家系统。

1.3　软件开发方法

软件开发的目标是在规定的投资和时间内，开发出符合用户需求的高质量的软件。为此需要有成功的开发方法。对软件开发方法的重视不够也是导致软件危机产生的原因之一，因此自软件工程的概念诞生以来，人们就非常重视对软件开发方法的研究，已经提出了多种软件开发方法和技术，对软件工程及软件产业的发展起到了重要的作用。

软件开发方法可分为两大类：面向过程的开发方法和面向对象的开发方法。下面将着重介绍结构化开发方法、原型化开发方法、面向对象的开发方法及敏捷开发方法等几种常用的软件开发方法。

1. 结构化开发方法

结构化开发方法（structured developing method）是一种传统的软件开发方法，它以系统工程的思想和工程化的方法为基础，强调用户至上、结构化、模块化，并自顶向下地对信息系统进行分析与设计。

结构化开发方法的指导思想是“自顶向下，逐步求精”，其基本原则是功能的分解与抽象。它是软件工程中最早出现的开发方法，相应的支持工具也较多，是软件开发方法中发展很成熟、应用非常广泛的方法。

结构化开发方法由三部分组成：结构化分析、结构化设计和结构化程序设计。结构化分析是根据分解与抽象的原则，按照系统中数据处理的流程，通过数据流图来建立系统的功能模型，完成需求分析工作。结构化设计是根据模块独立性准则、软件结构准则将数

据流图转换为软件的体系结构，用软件结构图来建立系统的物理模型，实现系统的系统设计。结构化程序设计是将每个模块的功能用相应的程序语言的标准控制结构〔顺序、选择和循环（重复）三种基本控制结构〕表示出来，从而构造出程序。

结构化开发方法的特点如下：

- 面向数据流：特别适用于数据处理领域的问题，如数据库管理系统、信息管理系统等。
- 模块化：系统被划分为多个模块，每个模块负责一部分功能，便于维护和升级。
- 信息隐蔽：每个模块内部的信息对其他模块是隐蔽的，降低了模块间的耦合度。
- 模块独立：模块之间的依赖性最小化，每个模块可以独立开发和测试。

总的来说，结构化开发方法是一个成熟且被广泛应用的开发模式，适用于规模适中、需求明确、复杂度不高的项目。由于它是一种面向数据流的开发方法，因此在数据处理领域尤为适用，如财务系统、库存管理系统等。但这种方法不适合大规模的、特别复杂的项目，也难以适应需求快速变化的场景。

2. 原型化开发方法

原型化开发方法又称快速原型法（rapid prototyping），是一种迭代式的软件开发过程，其基本思想是花费少量代价建立一个可运行的系统，使用户及早获得试用和学习的机会。该方法强调在开发的早期阶段创建可交互的模型或原型，这些原型可以是简化的版本，用于演示系统的关键特性和功能，以便用户和开发者能够评估其可行性和实用性。然后，通过原型的不断改进和完善，直到满足用户的所有需求。原型化方法的核心在于通过不断的反馈和迭代来完善产品。

（1）原型化开发方法的步骤

① 需求收集：与所有相关方进行沟通，收集和定义初步的用户需求和目标。

② 原型设计：基于需求创建初始原型，这通常涉及界面设计和交互流程的草图。

③ 用户测试：让用户与原型互动，收集用户的反馈和建议。

④ 分析和迭代：根据用户反馈对原型进行分析，确定需要改进或调整的地方，并进行相应的迭代开发。

⑤ 重复测试和迭代：重复上述的用户测试和分析迭代过程，直到原型满足用户需求为止。

⑥ 最终开发：一旦原型被验证，就可以进入完整的软件开发阶段，将原型转化为成熟的产品。

（2）原型化开发方法的优势

与结构化开发方法相比，原型化开发方法具有以下优势：

- 快速验证想法：通过原型，开发团队可以迅速验证核心概念和设计选择，避免在错误的方向上浪费资源。
- 增强沟通：原型为用户或非技术人员提供了一个直观的理解工具，帮助他们更好地理解产品的功能和外观。
- 降低风险：早期的原型测试有助于识别潜在的问题和风险，从而减少后期开发中的

成本和时间损失。

- 提高用户满意度：用户参与原型的试用和评估过程，可确保产品更加符合用户的需求和期望。

（3）原型的类型

按照功能划分，通常原型有三种类型：界面原型、功能原型、性能原型。

原型化开发方法是一种强大的工具，它可以帮助团队快速学习和适应，创造出更符合用户需求的产品。通过有效的需求管理、设计、测试和迭代，原型化方法能够显著提高软件开发的效率和成功率。然而，为了克服挑战并充分发挥这种方法的优势，团队需要精心规划并采取灵活的策略来管理整个原型化过程。

3. 面向对象的开发方法

面向对象（object-oriented）的开发方法简称 OO 方法，是 20 世纪 90 年代至今的主流软件开发方法。面向对象的开发方法的基本出发点是尽可能按照人类认识世界的方法和思维方式来分析和解决问题。客观世界是由许多具体的事物、事件、概念和规则组成的，这些均可被看成对象。面向对象的方法正是以对象作为最基本的元素，对象是分析问题、解决问题的核心。

面向对象的方法追求的是软件系统对现实世界的直接模拟，它将现实世界中的事物抽象成对象，直接映射到解空间，以消息驱动对象实现操作。面向对象的方法符合人类的认识规律。

（1）面向对象方法的组成

面向对象的方法包括面向对象分析（object-oriented analysis, OOA）、面向对象设计（object-oriented design, OOD）、面向对象程序设计（object-oriented programming, OOP）。其中，OOA 解决的是“做什么”的问题，它的基本任务是要建立起对象模型，定义系统的类和对象及其属性与操作；定义系统内部数据的传送处理；定义对象状态的变化，即时序。OOD 是在需求分析的基础上，进一步解决“如何做”的问题，也分为系统设计与详细设计。OOP 解决的是系统的实现问题。

面向对象的方法以对象为核心，强调模拟现实世界中的概念而不是算法，尽量用符合人类认识世界的思维方式来渐进地分析、解决问题，对软件开发过程的所有阶段进行综合考虑。这种方法能有效降低软件开发的复杂度，提高软件的可复用性和可扩充性，并能更好地适应需求的变化，从而提高软件质量。

（2）面向对象编程的优势

面向对象编程之所以受到广泛欢迎，主要得益于它具有的优势。

- 模块化：通过封装，对象成为独立的模块，易于理解和修改。
- 重用性：类和继承机制使得代码可以在多个项目中重用，提高了开发效率。
- 易于维护：对象的责任分明，修改一个对象通常不会影响到其他对象。
- 灵活性：对象的多态性允许程序在运行时动态地决定要调用的方法，增加了系统的灵活性。

面向对象开发方法在软件工程中具有显著的优势，但也存在一些需要注意的问题，如复杂度高，对团队合作要求高。因此，在实际应用中，应根据项目的需求和团队的实际情况来选择合适的方法，并充分利用其优点，尽量克服其缺点。

4. 敏捷软件开发方法

敏捷软件开发方法又称敏捷开发，其核心思想在于“敏捷”，即能够快速适应变化。在传统的软件开发过程中，往往采用瀑布模型等线性流程，从需求分析、设计、编码、测试到部署，每个阶段都有严格的顺序和界限。然而，这种方式在面对需求变化时往往显得力不从心，常常导致项目延期、成本超支等问题。敏捷开发则打破了这种僵化的流程，通过迭代和增量的方式，不断交付可用的软件产品，并根据用户的反馈进行调整和优化，提倡只编写“必需、够用”的文档，而不把过多的精力放在编写详尽的文档上。

（1）敏捷开发的核心原则

- 个体和互动优于流程和工具：这一原则强调团队成员之间的直接交流与合作比严格的流程和工具更为重要，鼓励团队成员进行面对面的沟通，以促进更高效的协作和问题的解决。
- 可运行的软件优于完备的文档：敏捷开发更重视能够运行的软件而不是详尽的文档，这意味着开发工作的重点是创建可以交付给客户的实际产品，而不是编写大量的计划或文档。
- 与客户合作优于合同谈判：与客户建立紧密的合作关系比坚持严格的合同条款更为重要。敏捷团队倾向于与客户一起工作，以确保产品满足他们的需求，并迅速响应任何变化。
- 响应变化优于遵循计划：敏捷方法强调适应性和灵活性。团队应该准备好随时调整方向，以适应新的情况，而不是死板地遵循原始计划。

由此可见，敏捷开发更强调与客户的协作、人与人之间的交互与团队合作，更注重不断地向用户提交可运行的软件，而不在编写详细的文档上花费过多的精力，尤其强调对软件需求变化的快速应变能力。

（2）敏捷开发的主要方法

按照敏捷开发的思想和原则，业界已推出了不少敏捷开发的具体实践方法，常见的敏捷开发方法包括 Scrum、极限编程（XP）、精益开发、特性驱动开发（FDD）、水晶方法（Crystal）等。这些方法各有特点，在实际应用中，团队可以根据项目的特点和需求选择合适的方法，并结合实际情况进行灵活调整。

（3）敏捷开发的主要特点

- 适应性：敏捷方法强调适应性而非预测性，因而能够更好地应对需求变化。
- 与客户合作：与客户紧密合作，确保产品满足客户需求。
- 交付频率：通过频繁的迭代和交付，可以不断获得用户的反馈并快速改进产品，快速交付可运行的软件。
- 团队协作：团队成员之间进行紧密的合作，共享信息，共同解决问题。

总之，敏捷软件开发方法以其灵活、协作和快速响应变化的特点，为现代软件工程提供了一种有效的实践方式，已经得到了业界的广泛认可和应用。当然，敏捷软件开发方法并非万能，也有其局限性。例如，对于大型的复杂项目，可能需要更加严格的控制和规划；同时，敏捷开发对于团队成员的素质和能力要求较高，需要他们具备较高的自我管理、协作和创新能力。

在未来，随着技术的不断进步和需求的不断变化，敏捷开发方法将会继续发挥重要作用，推动软件工程的持续发展。

1.4 软件工具与集成化开发环境

软件工具是用于支持和辅助软件开发、运行、维护、管理等活动的特殊软件。使用功能强大、方便适用的软件工具可以降低软件开发和维护的成本，减轻开发人员的劳动强度，提高软件生产的效率，改善软件产品的质量。因此，软件工具是软件工程研究中的重要内容之一。

1. 软件工具

软件工具种类繁多，涉及面广，按软件开发过程可分为软件开发工具、软件维护工具、软件管理和支持工具三大类。

（1）软件开发工具

按软件开发阶段可分为分析工具、设计工具、编码工具和测试工具。

- 分析工具：用于需求分析阶段，辅助系统分析员完成软件系统需求分析的活动，包括根据需求的定义，生成完整、准确、一致的需求分析说明。常用的分析工具有 PingCode、Worktile、ONES、JIRA、DOORS Next 等。
- 设计工具：用于帮助软件设计人员完成软件系统的设计活动的工具。通常，软件设计工具包括三类：基于图形或语言描述的设计工具、基于形式化描述的设计工具和面向对象的设计工具。常用的设计工具有 Enterprise Architect、Rational Rose 等。
- 编码工具：用于编程阶段的工具，包括语言程序、编译程序、解释程序等。这些编码工具可以是独立的应用程序，也可以是一个集成的程序开发环境，集成了源代码的编辑、编译和链接程序，以及用于源代码排错的调试程序和可供发布产品的发布程序。典型的集成程序开发环境有 Microsoft Visual Studio、Eclipse、PyCharm 等。
- 测试工具：用于测试阶段的工具。软件的测试包含很多方面，从不同的角度有不同的划分，如单元测试、集成测试、功能测试、性能测试、安全测试等。不同的测试的关注点不同，因而测试工具也有很多种，不同的测试工具实现的测试功能不同。例如，常用的功能测试工具有 WinRunner，性能测试工具有 LoadRunner、

WebLOAD、JMeter 等，单元测试工具有 Junit、NUnit、PyTest 等，接口测试工具有 JMeter 等。

（2）软件维护工具

软件维护的主要任务是在软件产品投入运行以后，纠正软件开发过程中未发现的错误，改进和完善软件的功能和性能，以适应用户新的需求，延长软件产品的使用寿命。这一阶段的软件工具包括版本控制工具、文档管理工具、逆向工程工具和再工程工具。

- 版本控制工具：用于存储、更新、恢复和管理某个软件的多个版本。常用的版本控制管理工具有 Visual SourceSafe（VSS）、git、SVN、PingCode、ONES 等。
- 文档管理工具：用于对软件过程中形成的文档进行分析、组织、维护和管理，这对提高软件的质量和软件维护具有重要的意义。常用的文档管理工具软件有 Confluence、ONES Wiki 等。
- 逆向工程工具：软件的逆向工程是指对已有的软件进行分析，获取比源代码更高级的表现形式，如提取出数据结构、体系结构、程序总体设计等各种有用的软件开发信息。早期的逆向工程工具有反汇编工具、反编译工具等。现在的逆向工程工具能够分析出高级程序设计语言的源程序，恢复程序的控制结构、流程图、PAD 图等更高级的抽象信息，为软件的理解和维护提供方便。常用的逆向工程工具有 IDA Pro。
- 再工程工具：软件的再工程是指在通过逆向工程获得软件设计等信息的基础上，利用这些信息修改或重构软件系统，增加新的功能和改进性能。再工程工具可以用来辅助软件开发人员重构一个功能和性能更为完善的软件系统。再工程工具的使用主要集中在代码重构、程序结构重构和数据结构重构等方面。

（3）软件管理和支持工具

软件产品管理与支持工具用于确保软件产品的质量和软件产品的开发效率，这类工具主要包括项目管理工具、配置管理工具、软件评价工具和风险分析工具。

- 项目管理工具：辅助软件管理人员进行项目计划、成本估算、资源分配、质量监控等。常用的项目管理工具有 MS Project、ONES、PingCode 等。
- 配置管理工具：对软件配置项进行标识、版本控制、变化控制的工具。常用的配置管理工具有 Visual SourceSafe、Git、SVN 等。
- 软件评价工具：对软件质量进行评价的工具。如 ISO 软件质量度量模型、McCall 软件度量模型等。
- 风险分析工具：风险管理对于一个大型项目是极为重要的。风险分析工具可以通过提供风险标示和分析，使项目管理者能有效地对软件开发过程中出现的风险进行控制和规避。

2. 集成化开发环境

现在的软件开发工具往往不是单一功能的，而是按照一定的开发模式或者开发方法组织起来、在一定的领域内使用的一个辅助软件开发的工具集合，这个工具集合就是“集成

化开发环境”。例如，Microsoft Visual Studio、Eclipse、PyCharm 等都是编程人员熟悉的集成开发环境。集成化开发环境也有一些不同的叫法，如“工具箱”“工具盒”“软件支撑环境”“软件工程环境”等，这些指的都是集成化开发环境。

软件工程工具和软件工程的理论是相辅相成的，正确地使用这些工具，需要掌握软件工程的基本概念、原理、方法和技术。利用这些工具辅助软件开发，将提高软件开发的效率和质量，同时也会让使用者加深对软件工程的概念、原理、方法和技术的认识。

习题 1

一、填空题

1. 软件工程是一门_____学科，其知识结构主要有_____、_____、_____。
2. 软件开发模型有多种，常用的有_____、_____、_____、_____和_____。
3. 软件开发工具按软件开发阶段可分为_____、_____、_____和_____四类。

二、选择题

1. 软件是一种_____产品。

A. 物质　　B. 逻辑　　C. 工具　　D. 文档

2. 软件工程是一门_____学科。

A. 理论性　　B. 原理性　　C. 工程性　　D. 心理性

3. 软件工程着重于_____。

A. 理论研究　　B. 原理探讨　　C. 建造软件系统　　D. 原理的理论

4. 软件危机的主要表现之一是_____。

A. 软件成本太高　　B. 软件产品的质量低劣

C. 软件开发中人员明显不足　　D. 软件生产率低下

5. 适用于企业内部大型软件开发方法的主要工作模型有_____。

A. 螺旋模型　　B. 循环模型　　C. 瀑布模型　　D. 专家模型

6. 准确地解决“软件系统必须做什么”是_____阶段的任务。

A. 可行性研究　　B. 需求分析　　C. 详细设计　　D. 编码

7. 软件生存周期中时间最长的是_____阶段。

A. 需求分析　　B. 系统设计　　C. 测试　　D. 维护

8. 下列不属于软件开发工具的是_____。

A. 分析工具　　B. 设计工具　　C. 编码工具　　D. 项目管理工具

9. 下列描述中正确的是_____。

A. 软件工程只是解决软件项目的问题

B. 软件工程主要解决软件产品的生产率问题

C. 软件工程的主要思想是强调在软件开发过程中需要运用工程化的原则

D. 软件工程主要解决软件开发中的技术问题

10. 下列描述中正确的是_____。

A. 程序就是软件　　B. 软件开发不受计算机系统的限制

C. 软件既是逻辑实体，又是物理实体　　D. 软件是程序、数据与相关文档的集合

三、简答题

1. 软件危机产生的原因是什么？
2. 软件工程的目标和内容是什么？
3. 软件生存周期有哪几个阶段？
4. 软件生存周期模型有哪些主要模型？
5. 常见的软件过程模型有哪几种，各种模型都有什么特点？
6. 软件开发方法主要有哪几种？
7. 软件工具分为哪几类？软件开发工具分为哪几类？