高等学校计算机应用规划教材

软件测试技术与实践

蔡建平 叶东升 康 妍
王爱菲 周百顺 李大奎 编著

清华大学出版社

北 京

内 容 简 介

本书共分为软件测试基础、软件测试管理、软件测试方法与技术三部分，覆盖了软件测评的各个环节和知识点，内容包括软件及软件测试的基本概念、软件测试分类与分级、软件缺陷管理、软件全生命周期测试、软件测试及其过程管理、软件静态测试与动态测试，以及面向对象软件测试的方法等。对于其中的一些重要环节，设计了基于案例驱动，利用典型开源工具进行软件测试实践的教学内容，如缺陷管理、测试管理、静态测试、单元测试、集成测试、系统测试(包括功能测试及性能测试)等。

本书可作为高等院校计算机相关专业的教材和参考书籍，还可作为软件测试应用型人才的培训教材，也可供软件测试、软件质量保证及软件开发和软件项目管理从业人员参考。

图书在版编目(CIP)数据

软件测试技术与实践 / 蔡建平 等编著. —北京：清华大学出版社，2018
(高等学校计算机应用规划教材)
ISBN 978-7-302-48688-6

Ⅰ. ①软… Ⅱ. ①蔡… Ⅲ. ①软件－测试－高等学校－教材 Ⅳ. ①TP311.5

中国版本图书馆 CIP 数据核字(2017)第 271394 号

责任编辑：王　军　李维杰
装帧设计：孔祥峰
责任校对：牛艳敏
责任印制：

出版发行：清华大学出版社
网　　址：http://www.tup.com.cn，http://www.wqbook.com
地　　址：北京清华大学学研大厦 A 座　　邮　　编：100084
社 总 机：010-62770175　　邮　　购：010-62786544
投稿与读者服务：010-62776969, c-service@tup.tsinghua.edu.cn
质 量 反 馈：010-62772015, zhiliang@tup.tsinghua.edu.cn
印 刷 者：
装 订 者：
经　　销：全国新华书店
开　　本：185mm×260mm　　**印　　张**：29　　**字　　数**：706 千字
版　　次：2018 年 1 月第 1 版　　**印　　次**：2018 年 1 月第 1 次印刷
印　　数：1~3000
定　　价：79.80 元

产品编号：

序

当前及今后，“软件定义一切”已充分说明了软件的重要性。软件作为产品及产品中的核心，其质量仍然可以认为是产品的生命。

在国家大力推进信息化与工业化融合之际，信息软件及工业软件的质量保证决定了“两化融合”的成败。软件质量保证的最重要手段之一就是软件测试。《软件测试技术与实践》作为航天中认推出支持两化软件测试的系列教材(《软件测试技术与实践》、《信息软件系统测试与实践》及《嵌入式软件测试与实践》)的基础，就显得非常重要。软件测试集技术、工程及实践于一身，如果没有好的技术基础和工程意识，眼高手低，那么在开展信息软件测试或工业软件测试时就会力不从心，多走弯路，无法把好质量关。

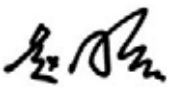
蔡建平教授长年从事软件工程、软件测试以及软件质量保证的研究、实践和教学，并为编写此书做了较长时间的准备，也出版了多本这方面的教材。特别是《软件测试方法与技术》被评为“全国工程硕士专业学位教育指导委员会推荐教材”和“软件工程专业核心课程系列教材”。本教材是在《软件测试方法与技术》和《软件测试实践教程》基础上改编而成的，具有如下主要特点：

(1) 该教材重要知识点的组织和讲述满足国内企业，特别是国内各种评测机构或组织对现代软件测试人才培养的要求；

(2) 该教材在传统软件测试技术和方法的基础上，特别强调软件测试是质量把控的重要手段之一，必须要与软件质量度量和评价相结合，要满足软件工程全生命周期软件测试的要求，要充分重视软件开发方法和应用方式对软件测试的影响，要注意软件测试工具对软件测试支持的重要作用等；

(3) 该教材还为高等院校和企业培训提供了软件测试课程实践教学方案、平台和案例，满足应用型软件测试人才培养的需要，满足软件测试工作人员的自学需要。

总之，该教材在软件测试技术的学习、普及、推广和软件测试应用型人才培养以及软件测试理论教学知识体系及实践教学体系的建立等方面进行了有益的探索和尝试。该教材内容全面、详实，实用性强。该教材的改版，将有益于国内软件测试人员和计算机相关专业的本科生及研究生的学习与能力的培养，有益于推动现代软件测试技术和方法的研究、教学和实践的进一步发展，同时对我国软件测试业的发展和软件测试紧缺人才的培养起到积极的促进作用。



航天中认软件测评科技(北京)有限责任公司总经理

2017 年 8 月 30 日于北京

前　言

当前，软件测试已从传统的软件工程瀑布模型中测试阶段的软件测试变化为覆盖包括需求分析、系统设计、详细设计、程序编码、内部测试、系统测试、系统安装、确认验收以及系统维护整个软件工程生命周期的软件测试；从过去单纯的测试概念发展到包括静态分析、质量度量与评价在内的评测结合的软件评测思想；从传统的测试内容分类到基于质量特性、子特性的测试内容分类；从传统的结构化程序测试方法到面向对象的软件测试方法；从早期的单机或桌面测试到网络应用测试及嵌入式应用测试；从以手工测试为主发展到离不开测试工具支持的测试及管理。事实上，软件测试也成为耗费人力、财力和时间的一项复杂的工作，对测试人员提出了高素质、专业化的要求。对软件测试人员不但要求精通各种软件测试技术和方法，有一定的软件测试工程实践经验；还要求他们熟悉软件开发技术和软件开发流程，具有快速学习专业知识或领域知识、掌握新技术和应用新工具的能力；另外，软件测试人员要有团队合作意识，善于和人沟通与交流，并能承受被人误解和指责的心理素质。

随着计算机技术的快速发展，软件越来越普遍地应用到各个领域和各个方面，且应用规模越来越大，应用形式越来越复杂，软件质量要求越来越高，软件测试越来越重要。对于高等院校而言，人才的培养是其核心的工作，鉴于高素质的软件测试专业人才越来越奇缺，航天中认积极开展校企合作，与大连理工大学软件学院共建软件测试课程和实验室，大力开展教学实践、教学改革的探索和实践，编写专业教材，凝练软件测试人才培养的成果。

本书是在《软件测试方法与技术》和《软件测试实践教程》基础上改编而成，将两本书的内容压缩、整合和完善，并试图将本书重点突出在两方面：软件测试技术与软件测试实践。本教材既包括成熟的理论基础，也包括软件测试工具的使用，同时还可以通过书中案例实践的学习巩固学习成果。本书是作者多年从事软件测试技术研究、软件测试课程教学及软件项目测试成果和经验的总结。全书共分 12 章，分为 3 部分：第Ⅰ部分(第 1 至第 3 章)是软件测试基础，涉及软件测试的一些基本概念和基础知识，如软件与软件危机、软件测试基本概念、软件测试分类与分级；第Ⅱ部分(第 4 至第 6 章)是本书的重点内容之一，详细讲述生命周期的软件测试方法与技术，包括软件缺陷与缺陷管理、软件测试及其过程管理、软件测试管理工具的使用及案例实践；第Ⅲ部分(第 7 至第 12 章)也是本书的重点内容之一，详细讲述软件测试的方法与技术，包括软件静态测试及软件动态测试。其中，基于 McCabe 设计与集成复杂性的集成测试方法解决了集成覆盖测试的理论问题。

本书系统全面地从软件测试基础理论到软件测试全生命周期实践角度，系统地为读者解决从事软件测试工作的问题。另外，本书几乎在各章对支撑该章软件测试方法和技术应用的开源软件测试工具详细地进行了应用介绍，并配有具体测试案例。这些工具及案例的

应用对于支持高校软件测试课程实践是有意义的。最后，本书取材新颖、内容翔实、通俗易懂、技术实用、覆盖面广、指导性强，既可作为软件测试相关课程的研究生(特别是工程硕士专业学位研究生)与本科生的教材，同时还可供软件测试培训和软件测试人员自学的书籍。

致谢，再一次感谢《软件测试方法与技术》和《软件测试实践教程》中提到的人员，感谢取材互联网上的有关原创作者，感谢清华大学出版社的大力支持和帮助。

2017 年 8 月 28 日于北京

作 者 简 介

蔡建平，在军队从事教学与全军军用共性软件、软件工程、软件质量保证等项目的论证及研究工作 20 多年，获军队科技进步一等奖一项、二等奖两项、三等奖两项，编著《Ada 程序设计语言高级教程》，发表各类学术文章 20 多篇。

在企业工作期间，主持开发了嵌入式软件工程和软件测试工具，这些工具已成功地用于航空、航天等国防项目的测试和软件工程化，极大地保证了这些项目的质量。

在北京工业大学工作期间，在软件学院的学科、专业、实验室、“211 工程”、教育部和北京市特色专业、科技创新平台以及学科交叉等建设方面做了大量的工作，取得了突出成果。获国家教育教学成果二等奖。“软件测试”及“高级软件编程技术”分别评为学校精品课程和研究生重点建设课程，《软件测试大学教程》、《软件测试实验指导教程》、《嵌入式软件测试实用技术》、《软件综合开发案例教程》4 部教材和专著已在清华大学出版社出版发行。其中《软件测试大学教程》于 2013 年被评为全国工程硕士专业学位教育指导委员会推荐教材。

科研上，发表各类论文 20 多篇，申请专利、软著多项，指导的学生科技活动成果获第十二届“挑战杯”全国大学生课外学术科技作品竞赛三等奖，指导的两篇硕士论文被评为校优秀论文。

作为惠普国际软件人才及产业基地的学术总监，负责全国各高校共建专业合作论证及顶层策划与设计，培养方案、课程体系、实训方案的设计与制定，与实训课程配套教材的研发组织及各门课程的研发组织，基地师资队伍及课程团队建设，以及 1000 多名学生实习/实训的组织与实施。

目前，在航天中认负责公司的咨询、研究及对内和对外的技术培训等业务，负责和参与了交通部、体育总局、中海油、大连理工大学、浪潮、长虹、美的、小天鹅、格力、轨道交通、国家电网、中国质量认证中心、汽车电子、医疗电子、家用电器等信息化建设项目及嵌入式系统项目的软件工程化、软件质量保证、软件测试以及配套实验室建设的咨询与培训。

蔡建平教授还是国家科学技术奖励、国家专利奖励、山东省科学技术奖励、北京市科学技术奖励、海淀区科学技术奖励、北京市文化创意产业、海淀区文化创意产业等专家库成员。

目　　录

第 I 部分　软件测试基础篇

1947 年，计算机还是由机械式继电器和真空管驱动的、有房间那么大的庞然大物，由哈佛大学制造的 MARK-II 则是体现当时技术水平的计算机。在一次整机运行中，它突然停止了工作。技术人员爬到计算机上找原因，发现是一只飞蛾受光和热的吸引飞到了计算机内部一组继电器的触点之间，然后被高电压击死。由此，计算机的缺陷产生了，虽然最后该缺陷被技术人员解决了，但是我们从此认识到了它就是缺陷。

如今软件已经渗透到我们的日常生活中，从办公设备到家用电器，从通信工具到航空航天事业，软件无处不在，然而却又很难完美无缺。

1994 年的秋天，迪士尼公司发布了第一个面向儿童的多媒体光盘——《狮子王动画故事书》(*The Lion King Animated Storybook*)。对此，迪士尼公司做了大量的宣传。因此，销售额非常可观。然而圣诞节过后，公司接到了大量的投诉电话，称游戏不能运行。经证实，造成这种后果的原因是迪士尼公司未对市面上使用的许多不同类型的 PC 机型进行测试，软件只能在少数系统中运行。

同样是 1994 年，英特尔奔腾浮点除法缺陷事件，不仅使英特尔公司的形象受到严重影响，并且为自己处理软件缺陷的行为付出了 4 亿多美元的代价。

类似的还有美国航天局火星极地登陆者号探测器事件、爱国者导弹防御系统事件、千年虫问题等，这些事件的后果有的是带来不便，例如游戏玩不成，有的可能是灾难性的后果——导致机毁人亡。它们的发生都是由于在软件中隐藏着错误。

软件为什么会频繁出问题，如何杜绝或将它们减至最少，将影响降至最低呢？在论述软件测试概念之前先介绍一下软件、软件危机及软件工程等概念，然后再讲解软件测试的相关知识。

第1章　软件与软件危机

我们都知道软件的重要意义：软件是信息化的核心，现代国民经济、国防建设、社会发展及人民生活都离不开软件。软件产业是增长最快的朝阳产业，是高投入、高产出、无污染、低能耗的绿色产业。软件产业关系到国家经济和文化安全，体现了国家综合实力，是决定未来国际竞争地位的战略性产业。

但软件到底什么？它具有什么样的特性？它能够干什么？

1.1.1　软件特性

软件是人通过智力劳动产生的，软件产品是人的思维结果，是一个逻辑部件，而不是一个物理部件。因此，软件生产水平最终在相当程度上取决于软件人员的教育、训练和经验的积累。所以，软件具有与硬件不同的一些特点。

1. 软件与硬件的不同

软件与硬件的不同或差别主要反映在以下几个方面。

1) 表现形式不同

硬件有形，有色，有味，看得见，摸得着，闻得到。而软件无形，无色，无味，看不见，摸不着，闻不到。软件大多存在于人们的脑海里或纸面上，它的正确与否，是好是坏，一直要到程序在机器上运行才能知道。这就给设计、生产和管理带来许多困难。

2) 生产方式不同

软件开发是智力的高度发挥，而不是传统意义上的硬件制造。尽管软件开发与硬件制造之间有许多共同点，但这两种活动是根本不同的。在两种活动中，通过好的设计能够得到好的质量，但硬件制造阶段可能引入的质量问题在软件开发中却不会出现，反之亦然。这两种活动都依靠人，但人的作用和工作专长之间的关系是完全不同的。因为软件是逻辑产品，如几个人共同完成一个软件项目时，人与人之间就有一个思想交流的问题，称之为通信关系。通信是要付出代价的，不仅要花费时间，现实中由于通信中的疏忽常常会使错误增加。人虽然是最聪明的，但人也是最容易犯错误的。

3) 要求不同

硬件产品允许有误差，如加工一根轴，其外径精度要求为 Φ500±0.1。生产时，只要达到规定的精度要求就算合格。而软件产品却不允许有误差，要 1 就是 1。如美国金星探测器水手 1 号，导航程序的一条语句的语法正确，但语义错了，结果飞行偏离航线，最终导致试验的失败。又如，阿波罗宇宙飞船飞行控制软件，由于粗心，把一个逗号写成一个句号，又没能检查出来，几乎造成悲剧性的后果。这就给软件开发和维护，以及它的质量保证体系提出了很高的要求。

4) 维护不同

硬件是会用旧、用坏的这是因为硬件在使用过程中，由于受到环境的影响，如灰尘、温湿度变化、空气污染、振动等因素而使产品产生腐蚀或磨损，使硬件故障率增加，甚至损坏，以致不能使用。解决的办法，换上一个相同的备件就是了。而软件不受那些引起硬件损坏的环境因素的影响。因此，在理论上，软件不会用旧、用坏。但实际上，软件也会变旧、变坏。因为在软件的整个生命周期中，一直处于改变(维护)状态。而随着某些缺陷的改变，很可能引入一些新的缺陷，因而使软件的故障率增加，品质变坏。硬件某一部分变坏，可以使用备件，而软件则不存在这种备件，因为软件中任何缺陷都会在机器上导致同样的错误。所以，软件维护要比硬件维护复杂得多。

从上我们可以总结出软件具有同传统的工业产品相比的一些特性。

2. 软件的特性

1) 软件是一种逻辑实体，具有抽象性

这个特点使它与其他工程对象有着明显的差异。人们可以把它记录在纸上、内存中和磁盘、光盘上，但无法看到软件本身的形态，必须通过观察、分析、思考、判断，才能了解它的功能、性能等特性。

2) 软件没有明显的制造过程

一旦研制开发成功，就可以大量拷贝同一内容的副本。所以对软件的质量控制，必须着重在软件开发方面下工夫。

3) 软件在使用过程中，没有磨损、老化的问题，但有退化问题

软件在生命周期后期不会因为磨损而老化，但会为了适应硬件、环境以及需求的变化而进行修改，而这些修改又不可避免地会引入错误，导致软件失效率升高，从而使得软件退化。当修改的成本变得难以接受时，软件就被抛弃。

4) 软件对硬件和环境有着不同程度的依赖性

这导致软件移植的问题。

5) 软件的开发至今尚未完全摆脱手工作坊式的开发方式，生产效率低

6) 软件是复杂的，而且以后会更加复杂

软件是人类有史以来生产的复杂度最高的工业产品。软件涉及人类社会的各行各业、方方面面，软件开发常常涉及其他领域的专门知识，这对软件工程师提出了很高的要求。

7) 软件的成本相当昂贵

软件开发需要投入大量、高强度的脑力劳动，成本非常高，风险也大。现在软件的开销已大大超过了硬件的开销。

8) 软件工作牵涉很多社会因素

许多软件的开发和运行涉及机构、体制和管理方式等问题，还会涉及人们的观念和心理。这些人的因素，常常成为软件开发的困难所在，直接影响到项目的成败。

1.1.2　软件种类

软件已经渗透到我们的日常生活，用于各行各业。具体地说，软件按如下形式分类：

(1) 系统软件(如操作系统、数据库管理系统、设备驱动程序、通信处理程序等)；

(2) 应用软件(如事务软件、实时软件、工程和科学软件、嵌入式软件、娱乐软件、个人计算机软件、人工智能软件等)；

(3) 工具软件(如文本编辑软件、文件格式化软件、磁盘向磁带传输数据的软件、程序库系统以及支持需求分析、设计、实现、测试和管理的软件等)；

(4) 可重用软件。

1.2　软件危机

前面提到的美国于 20 世纪 60 年代初飞向金星的第一个空间探测器(水手 I 号)，因其飞行舱内计算机导航程序中一条语句的语义出错，致使偏离航线无法取得成功。这条语句的错误并不是语法错误，而是具有完全不同于程序员所期望的意思——语义错误。可以称得上是世界上最精心设计，并花费了巨额投资的美国阿波罗登月飞行计划的软件，仍然没有避免出错；另外，阿波罗 8 号太空飞船的一个计算机软件错误，造成存储器的一部分信息丢失；还有，阿波罗 14 号在飞行的 10 天里，出现了 18 个软件错误。当时，软件系统的可靠性得不到保证，几乎没有不存在错误的软件系统。

那时，计算机已有近 20 年的历史，一方面硬件成本每隔两到三年降低一半，内存和外存则成本每年降低 40%左右，硬件性能价格比每十年提高一个数量级，但所需的软件很少能在成本、时间进度、功能规模、维护能力等方面达到要求，特别是可靠性难以符合人们的需要。正如 E.E.David 指出的那样，大型系统的软件生产已经成为管理人员担惊受怕的项目，于是，人们在 20 世纪 60 年代后期惊呼发生了软件危机(software crisis)。

1.2.1　软件危机的分析

产生软件危机的原因是多方面的，但不管怎样，软件危机有它内在的或本质上的原因。下面就软件危机产生的诸多原因进行分析，揭示软件危机内在或本质上的原因。

1. 早期编程的特点

从 20 世纪 40 年代开始，人们从在 MARK-I 和 ENIAC 计算机上编制程序，到软件危机发生时为止的 20 多年时间里，对软件开发的理解就是编程，且编程是在一种无序的、崇尚个人技巧的状态中完成的。

和今天的软件开发相比，那时的编程具有一些特点。

1) 软件规模相对较小

原因有二：

① 人们对软件可能达到的功能认识有限，那时最为关心的是计算机硬件的发展。作为一名计算机专业人员，他不太关心软件问题(只有为数不多的专业人员才去关心软件)，

但他必须懂得计算机的结构。作为一个机构，其大量资金也被用于计算机硬件开销，软件只是作为展现其软件性能的一种手段而投入少量资金。

② 硬件性能从某种意义上左右着人们对软件的需求，人们总是不自觉地在心中盘算着硬件支持的可能性，这种现象从根本上阻碍了软件的广泛应用，从而限制了软件规模。

2) 编程作为一门技艺

大部分软件技术人员不太关心他人的工作，他们往往陶醉于自己的编程技巧，并且那时也无编程规范与标准，这也是由软件规模决定的。决定软件质量的唯一因素就是编程人员的素质，时间进度亦是如此。根据 H.Sackman 等人的调查，素质好的人与素质差的人，在软件生产效率上的比例是 10∶1，在程序处理速度和存储容量上的比例是 5∶1。

3) 缺少有效方法与软件工具的支持

在当时几乎谈不上有效的编程方法，使用最多的亦只是简单的控制流图。软件工具只有子程序库、装入程序、编辑程序、排错程序以及汇编和编译程序(后来才有链接程序)，以至于许多程序员沉溺于编程技艺的掌握和使用上。

4) 不重视软件开发的管理

由于人们重视个人技能，再加上软件开发过程能见度低，许多管理人员甚至根本不知道软件技术人员在干什么，究竟做得如何，从而造成管理活动几乎不存在。直到今天，这种观念在一些机构中仍有残留，为此，我们对软件开发的管理问题必须给予更多的重视。

5) 软件开发后的维护工作很难进行

由于人们重视个人技能，一旦需要做某些修改，就要原编程人员进行修改。如果他已离开本机构，就需要别人去读懂他的程序，再进行修改和补充。此项工作极其辛苦，说不定修改了一处，反而出现上百处漏洞，变得得不偿失。

上述编程特点导致出现人们常说的软件危机现象。

2. 大型软件开发问题

进入 20 世纪 60 年代，应客观需求需要制作一些大型软件，这样就出现了像第一个空间探测器所描述的例子。国外在开发一些大型软件系统时遇到了许多困难，有些系统最终彻底失败了；有些系统虽然完成了，但比原定计划推迟了好几年，而且费用大大超出了预算；有些系统未能达到用户当初的期望；有些系统则无法进行修改、维护。IBM 公司 OSS/360 系统和美国空军某后勤系统都花费了几千人/年努力，历尽艰辛，但结果令人失望。

随着软件开发应用范围的增广，软件开发规模越来越大。大型软件开发项目需要组织一定的人力共同完成，而多数管理人员缺乏开发大型软件开发系统的经验，而多数软件开发人员又缺乏管理方面的经验。各类人员的信息交流不及时、不准确，有时还会产生误解。软件开发项目开发人员不能有效地、独立自主地处理大型软件开发的全部关系和各个分支，因此容易产生疏漏和错误。这也是导致软件危机产生的一个原因。

3. 软件生产的知识密集和人力密集的特点

由于计算机技术和应用发展迅速，知识更新周期加快，软件开发人员经常处在变化之中，不仅需要适应硬件更新的变化，而且还要涉及日益扩大的应用领域问题研究；软件开

发人员所进行的每一项软件开发几乎都必须调整自身的知识结构以适应新的问题求解的需要，而这种调整是人所固有的学习行为，难以用工具代替。

软件生产的这种知识密集和人力密集的特点是造成软件危机的根源所在。从软件危机的种种表现和软件作为逻辑产品的特殊性，可以发现软件危机的具体原因。

4. 用户需求难以明确

对于大型软件往往需要许多人合作开发，甚至要求软件开发人员在软件开发过程中深入应用领域对相关问题进行研究，了解用户需求。这样就需要在用户与软件开发人员之间以及软件开发人员之间相互通信。在此过程中难免发生理解上的差异，导致用户需求不明确的问题产生。

用户需求不明确问题主要体现在四个方面：

① 在软件开发出来之前，用户自己也不清楚软件开发的具体需求；

② 用户对软件开发需求的描述不精确，可能有遗漏，有二义性，甚至有错误；

③ 在软件开发过程中，用户还提出修改软件开发功能、界面、支撑环境等方面的要求；

④ 软件开发人员对用户需求的理解与用户本来的愿望有差异。这些需求问题将导致后续软件错误的设计或实现，而要消除这些需求上的误解和错误往往需要付出巨大的代价。这同样是产生软件危机的一个原因。

5. 缺乏正确的理论指导，缺乏有力的方法学和工具方面的支持

由于软件开发不同于大多数其他工业产品，其开发过程是复杂的逻辑思维过程，其产品极大程度地依赖于开发人员高度的智力投入。由于过分地依靠程序设计人员在软件开发过程中的技巧和创造性，因此加剧了软件开发产品的个性化，这也是发生软件开发危机的一个重要原因。

6. 软件开发复杂度越来越高

软件开发不仅仅是在规模上快速地发展扩大，而且其复杂性也急剧地增加。软件开发产品的特殊性和人类智力的局限性，导致人们无力处理复杂问题。所谓复杂问题的概念是相对的，一旦人们采用先进的组织形式、开发方法和工具提高了软件开发效率和能力，新的、更大的、更复杂的问题又摆在人们的面前。

7. 软件危机的本质原因

从前面软件危机的原因分析来看，软件危机产生的本质原因主要有两点：

① 与软件本身的特点有关；

② 与软件的开发人员有关。

从上我们可以看出，软件危机是指在计算机的开发和维护过程中所遇到的一系列严重问题。这些问题绝不仅仅是不能正常运行的软件才具有的，实际上，几乎所有软件都不同程度地存在这些问题。

1.2.2　软件危机现象

事实上，软件危机包含着两方面的问题：

① 如何开发软件，以满足对软件日益增长的需求；

② 如何维护数量不断膨胀的已有软件。具体来说，软件危机主要有以下一些典型表现。

1. 对软件开发成本和进度的估计常常很不准确

实际成本比估计成本有可能高出一个数量级，实际进度比预期进度拖延几个月甚至几年的现象并不罕见。这种现象降低了软件开发组织的信誉。而为了赶进度和节约成本所采取的一些权宜之计又往往损害了软件产品的质量，从而不可避免地会引起用户的不满。

2. 用户对已完成的软件系统不满意的现象经常发生

软件开发人员常常在对用户要求只有模糊的了解，甚至对所要解决的问题还没有确切认识的情况下，就仓促上阵匆忙着手编写程序。软件开发人员和用户之间的信息交流往往很不充分，闭门造车必然导致最终的产品不符合用户的实际需要。

3. 软件产品的质量往往靠不住

软件可靠性和质量保证的确切定量概念刚刚出现不久，软件质量保证技术还没有坚持不懈地应用到软件开发的全过程中，这些都导致软件产品发生质量问题。

4. 软件常常是不可维护的

很多程序中的错误是非常难改正的，实际上不可能使这些程序适应新的硬件环境，也不能根据用户的需要在原有程序中增加一些新的功能。可重用的软件还是一个没有完全做到的、正在努力追求的目标，人们仍然在重复开发类似的或基本类似的软件。

5. 软件通常没有适当的文档资料

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，而且应该是最新式的(即和程序代码完全一致)。软件开发组织的管理人员可以使用这些文档资料作为里程碑，来管理和评价软件开发工程的进展状况；软件开发人员可以利用它们作为通信工具，在软件开发过程中准确地交流信息；对于软件维护人员而言，这些文档资料更是至关重要、必不可少的。缺乏必要的文档资料或者文档资料不合格，必然给软件开发和维护带来许多严重的困难和问题。

6. 软件成本在计算机系统总成本中所占的比例逐年上升

由于微电子学技术的进步和生产自动化程度的不断提高，硬件成本逐年下降，然而软件开发需要大量人力，软件成本随着通货膨胀以及软件规模和数量的不断扩大而持续上升。美国在 1985 年软件成本大约已占计算机系统总成本的 90%。

7. 软件开发生产率提高的速度，既跟不上硬件的发展速度，也远远跟不上计算机应用迅速普及深入的趋势

软件产品供不应求的现象使人类不能充分利用现代计算机硬件提供的巨大潜力。

以上列举的仅仅是软件危机的一些明显的表现，与软件开发和维护有关的问题远远不止这些。总之，软件危机一方面与软件本身的特点有关，另一方面与软件开发和维护的方法不正确有关。

1.2.3　避免软件危机的方法

如何避免软件危机的产生是一个非常大的课题，是否存在非常有效、十分管用的方法现在还未有结论。以下两点是在软件开发过程中为避免软件危机问题的出现通常要求大家要努力做到的。

1) 应该对计算机软件有一个正确的认识

应该彻底清除在计算机系统早期发展阶段形成的软件就是程序的错误观念，一个软件必须由一个完整的配置组成。事实上，软件是程序、数据及相关文档的完整集合。其中，程序是能够完成预定功能和性能的可执行的指令序列；数据是使程序能够适当地处理信息的数据结构；文档是开发、使用和维护程序所需要的图文资料。1983 年 IEEE 为软件下的定义是：计算机程序、方法、规则、相关的文档资料以及在计算机上运行程序时所必需的数据。虽然表面上看，这个定义列出了软件的五个配置成分，但是，方法和规则通常是在文档中说明并在程序中实现的。

2) 必须充分认识到软件开发不是某个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、概念、技术和方法，特别要吸取几十年来人类从事计算机硬件研究和开发的经验教训。极力推广和使用在实践中总结出来的开发软件的成功技术和方法，并且研究探索更好、更有效的技术和方法，使用更好的开发工具。这样，在软件开发中就会最大限度地消除软件危机的出现。

1.3　软件工程

1968 年秋季，NATO(北大西洋公约组织，简称北约)的科技委员会召集了近 50 名一流的编程人员、计算机科学家和工业界巨头，讨论和制定摆脱软件危机的对策。在那次会议上第一次提出了软件工程这个概念。到现在，软件工程走过了约 50 年的历程。

在这约 50 年的发展历程中，人们针对软件危机的表现和原因，经过不断的实践和总结，越来越认识到：按照工程化的原则和方法组织软件开发工作，是摆脱软件危机的一条主要出路。

1.3.1　软件工程定义

今天，尽管软件危机并未被彻底解决，但软件工程近 50 年的发展仍可以说是硕果累累。下面我们给出软件工程的定义，然后简单讨论一下软件工程所包括的内容。

软件工程是一门研究如何用系统化、规范化、数量化等工程原则和方法进行软件的开

发和维护的学科。它作为一门新兴的工程学科，主要研究软件生产的客观规律性，建立与系统化软件生产有关的概念、原则、方法、技术和工具，指导和支持软件系统的生产活动，以期达到降低软件生产成本、改进软件产品质量、提高软件生产率水平的目标。软件工程学从硬件工程和其他人类工程中吸收了许多成功的经验，明确提出了软件生命周期的模型，发展了许多软件开发与维护阶段适用的技术和方法，并应用于软件工程实践，取得了良好的效果。

1. 软件工程的具体含义

软件工程的具体含义体现在四个方面：

(1) 把软件开发看成一项有计划、分阶段、严格按照标准或规范进行的活动(软件工程是指导计算机软件开发和维护的工程学科，软件工程方法=工程方法+管理技术+技术方法)；

(2) 将系统的、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(将工程化应用于软件中，并研究提到的上述途径)；

(3) 要求采用适当的软件开发方法和支持环境及编程语言来表示和支持软件开发各阶段的各种活动，并使开发过程条令化、规范化，使软件产品标准化、开发人员专业化；

(4) 用工程学的观点进行费用估算，制定进度，制定计划；用管理科学中的方法和原理进行软件生产的管理；用数学的方法建立软件开发中的各种模型和各种算法。

2. 软件工程三要素

软件工程包括三个要素：方法、工具和过程。

1) 软件工程方法

软件工程方法为软件开发提供了如何做的技术。它包括了多方面的任务，如项目计划与估算，软件系统需求分析，数据结构、系统总体结构的设计，具体算法的设计、编码、测试以及维护等。

2) 软件工具

软件工具为软件工程方法提供了自动的或半自动的软件支撑环境。目前，已经推出了许多软件工具，这些软件工具集成起来，建立起称之为计算机辅助软件工程(Computer Aided Software Engineering，CASE)的软件开发支撑系统。CASE 将各种软件工具、开发机器和存放开发过程信息的工程数据库组合起来，形成软件工程环境。

3) 软件工程过程

软件工程过程则是将软件工程方法和软件工具综合起来以达到合理、及时地进行计算机软件开发的目的。过程定义了方法使用的顺序、要求交付的文档资料、为保证质量和协调变化所需要的管理及软件开发各个阶段完成的里程碑。

软件工程是一种层次化的技术。任何工程方法(包括软件工程)必须以有组织的质量保证为基础。全面的质量管理和类似的理念刺激了不断的过程改进，正是这种改进导致更加成熟的软件工程方法的不断出现。支持软件工程的根基就在于对质量的关注。

3. 软件工程基本原理

著名软件工程专家 B.W.Boehm 综合有关专家和学者的意见并总结多年来开发软件的经验，于 1983 年在一篇论文中提出了软件工程的七条基本原理。

1) 用分阶段的生命周期计划进行严格的管理

统计表明，不成功的软件项目中有 50%左右是由于计划不周造成的。应该把软件生命周期划分为若干阶段，并制定出相应的切实可行的计划，严格按照计划对开发和维护进行管理。B.W.Boehm 认为，应制定和严格执行 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) 坚持进行阶段评审

设计的错误占软件错误的 63%，编码错误只占 37%。而且在后期纠正错误的代价非常高。因此，必须严格坚持阶段评审，及早发现和纠正错误。

3) 实行严格的产品控制

在现实中，由于外部原因要求对需求等进行修改是难免的，但必须有严格的管理制度和措施。

4) 采用现代程序设计技术

如结构化程序分析、结构化程序设计和面向对象程序设计等。

5) 软件工程结果应能清楚地审查

由于软件是一种看不见、摸不着的逻辑产品，对它的检验和审查很困难。因此，应提供可视化的检验标准和方法。

6) 开发小组的人员应该少而精

软件开发小组的人员应该是素质高，人员不宜过多。人员素质低和人员过多，都会导致软件的错误率高，且开发效率下降，成本增加。

7) 承认不断改进软件工程实践的必要性

软件工程是一门不断迅速发展的学科，必须学习和跟踪先进的技术和方法，也要不断地总结经验、改进方法，要不断进行技术创新。

B.W.Boehm 指出，遵循前六条基本原理，能够实现软件的工程化生产；按照第七条原理，不仅要积极主动地采纳新的软件技术，而且要注意不断总结经验。

4. 软件工程框架

软件工程(Software Engineering)框架可概括为：目标、过程和原则。

1) 软件工程的目标

软件工程的目标是生产具有正确性、可用性、开销适宜并且软件项目成功的软件产品。正确性指软件产品达到预期功能的程度；可用性指软件基本结构、实现及文档为用户可用的程度；开销适宜是指软件开发、运行的整个开销满足用户要求的程度；软件项目成功指开发成本低、功能与性能满足需求、易于移植、维护方便以及按时完成开发任务并及时交付软件产品。这些目标的实现不论在理论上还是在实践中均存在很多待解决的问题，它们

形成了对过程、过程模型及工程方法选取的约束。

2) 软件工程的过程

软件工程的过程是指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，主要包括开发过程、运作过程、维护过程。它们覆盖了需求、设计、实现、确认以及维护等活动。需求活动包括问题分析和需求分析。问题分析获取需求定义，又称软件需求规约；需求分析生成功能规约。设计活动一般包括概要设计和详细设计。概要设计建立整个软件系统结构，包括子系统、模块以及相关层次的说明、每一模块的接口定义；详细设计产生程序员可用的模块说明，包括每一模块中的数据结构说明及加工描述。实现活动把设计结果转换为可执行的程序代码。确认活动贯穿于整个开发过程，实现完成后的确认，保证最终产品满足用户的要求。维护活动包括使用过程中的扩充、修改与完善。伴随以上过程，还有管理过程、支持过程、培训过程等。

3) 软件工程的原则

软件工程的原则是指围绕工程设计、工程支持以及工程管理在软件开发过程中必须遵循的原则，具体为：

① 采取适宜的开发模型，用以控制易变的需求。

② 采用合适的设计方法，支持软件的模块化、抽象与信息隐藏、局部化、一致性以及适应性等设计要求。

③ 提供高质量的工程支持，特别要强调软件工具和环境对软件过程的支持。

④ 重视开发过程的管理，要有效利用可用的资源、生产满足目标的软件产品、提高软件组织的生产能力等。

6. 软件工程的本质特征

基于软件特性及软件危机产生的原因，我们可以清楚地了解软件工程的本质特征，即：

① 软件工程关注于大型程序的构造。

② 软件工程的中心课题是控制复杂性。

③ 软件经常变化(控制和管理)。

④ 开发软件的效率非常重要(工具与环境)。

⑤ 和谐地合作是开发软件的关键(团队精神)。

⑥ 软件必须有效地支持它的用户。

⑦ 在软件工程领域中是由一种文化背景的人为另一种文化背景的人创造产品。

7. 软件开发技术和软件项目管理

软件工程包括软件开发技术和软件项目管理两方面的内容。

1) 软件开发技术与开发方法

软件开发技术是为了完成软件生命周期各阶段的任务所必须具备的技术手段。软件开发技术包括：①软件开发方法(是一种使用早已定义好的技术集及符号表示习惯来组织软件生产过程的方法，一般表述成一系列的步骤，每一步都与相应的技术和符号相关，目的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)。②软件工具(是为了支持软

件人员开发和维护而使用的软件，它可以放大人类的智力、提高工作效率、便于管理实施，并为软件开发方法提供了自动的或半自动的软件支撑环境，辅助软件开发任务的完成。当前，在软件开发过程中人们越来越重视工具的使用，用以辅助进行软件项目管理与技术生产)。③软件环境(是将软件生命周期各阶段使用的软件工具有机地集合成为一个整体，形成能够连续支持软件开发与维护全过程的集成化软件支援环境，用以开发软件，提高开发效率和软件质量，降低开发成本，以期从管理和技术两方面解决软件危机问题)。

在软件开发过程中常用的软件开发方法有：

(1) 面向数据流的结构化程序开发方法(最终关注程序结构)。

- 指导思想：自顶向下，逐步求精。
- 基本原则：功能的分解与抽象。
- 适合于数据处理领域的问题。

(2) 面向数据结构的开发方法——Jackson 方法。

- JSP(Jackson Structured Programming)：首先描述问题的输入/输出数据结构，分析其对应性，然后推出相应的程序结构，从而给出问题的软件过程描述。以数据结构为驱动。
- JSD(Jackson Structured Design)：首先建立现实世界的模型，再确定系统的功能需求。以事件为驱动，基于进程的开发方法。

(3) 支持程序开发的形式化方法(基于模型的方法)——维也纳方法。

将软件系统当做模型来给予描述，把软件的输入、输出看作模型对象，把这些对象在计算机内的状态看做该模型在对象上的操作。

(4) 面向对象开发方法

- 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。
- 面向对象开发方法包括面向对象分析、面向对象设计、面向对象实现。

2) 软件项目管理

软件项目管理包括软件度量、项目估算、进度控制、人员组织、配置管理、项目计划等。

3) 软件工程中的技术复审和管理复审

每阶段结束前要进行技术复审和管理复审。

(1) 技术复审是从技术角度确保质量，降低软件成本(尽早发现问题)。审查过程包括准备(如成立审查小组)、简要介绍情况、阅读被审文档、开审查会、返工、复查等。

(2) 管理复审主要是从管理的角度对成本、进度、经费等进行复审，以保证项目正常开展。

在软件工程理论的指导下，发达国家已经建立起较为完备的软件工业化生产体系，形成了强大的软件生产能力。软件标准化与可重用性得到了工业界的高度重视，在避免重用劳动、缓解软件危机方面起到了重要作用。

1.3.2　软件生命周期

软件工程的传统解决途径强调使用生命周期方法学和各种结构分析及结构设计技术。

它们是在 20 世纪 70 年代为了应付应用软件日益增长的复杂程度、漫长的开发周期以及用户对软件产品经常不满意的状况而发展起来的。

人类解决复杂问题时普遍采用的一个策略就是各个击破，也就是对问题进行分解，然后再分别解决各个子问题的策略。软件工程实际上是从时间角度对软件开发和维护的复杂问题进行分解，把软件生存的漫长周期或把用户的要求转变成软件产品的过程。

1. 什么是软件生命周期

同任何事物一样，软件产品或软件系统也要经历孕育、诞生、成长、成熟、衰亡等阶段，一般称为软件生命周期(又称软件生存周期)。把整个软件生命周期划分为若干阶段，使得每个阶段有明确的任务，使规模大、结构复杂和管理复杂的软件开发变得容易控制和管理。通常，软件生命周期包括可行性分析与开发项计划、需求分析、设计(概要设计和详细设计)、编码、测试、维护等活动，可以将这些活动以适当的方式分配到不同的阶段去完成。

这种按时间划分的思想方法是软件工程中的一种思想原则，即按部就班、逐步推进，每个阶段都要有定义、工作、审查、形成文档，以供交流或备查，以提高软件的质量。但随着新的面向对象设计方法和技术的成熟，软件生命周期设计方法的指导意义正在逐步减小。

2. 软件生命周期的阶段划分

1) 软件生命周期的阶段概念

可以从以下几个方面了解软件生命周期的阶段概念。

(1) 采用生命周期方法开发软件的时候，从对任务的抽象逻辑分析开始，一个阶段一个阶段地进行开发。

(2) 前一个阶段任务的完成是开始进行后一个阶段工作的前提和基础，而后一阶段任务的完成通常是使前一阶段提出的解法更进一步具体化，加入了更多的物理细节。

(3) 每一个阶段的开始和结束都有严格标准，对于任何两个相邻的阶段而言，前一阶段的结束标准就是后一阶段的开始标准。

(4) 在每一个阶段结束之前都必须进行正式、严格的技术审查和管理复审，从技术和管理两方面对这个阶段的开发成果进行检查，通过之后这个阶段才算结束(如果检查通不过，则必须进行必要的返工，并且返工后还要再经过审查)。

(5) 审查的一条主要标准就是每个阶段都应该交出最新式的(即和所开发的软件完全一致的)、高质量的文档资料，从而保证在软件开发工程结束时有完整、准确的软件配置交付使用。

(6) 文档是通信的工具，它们清楚、准确地说明了到这个时候为止，关于该项工程已经知道了什么，同时确立了下一步工作的基础。此外，文档也起备忘录的作用，如果文档不完整，那么一定是某些工作忘记做了，在进入生命周期的下一阶段之前，必须补足这些遗漏的细节。在完成生命周期每个阶段的任务时，应该采用适合该阶段任务特点的系统化的技术方法，如结构分析或结构设计技术。

2) 软件生命周期阶段划分的意义

软件生命周期阶段划分具有以下意义：

(1) 软件生命周期划分成若干个阶段，每个阶段的任务相对独立，而且比较简单，便于不同人员分工协作，从而降低了整个软件开发工程的困难程度。

(2) 在软件生命周期的每个阶段都采用科学的管理技术和良好的技术方法，而且在每个阶段结束之前都从技术和管理两个角度进行严格的审查，合格之后才开始下一阶段的工作，这就使软件开发工程的全过程以一种有条不紊的方式进行，保证了软件的质量，特别是提高了软件的可维护性。

总之，采用软件工程方法论可以大大提高软件开发的成功率，软件开发的生产率也能明显提高。

3) 软件生命周期阶段划分的方法

目前划分软件生命周期阶段的方法有许多种，软件规模、种类、开发方式、开发环境以及开发时使用的方法论都影响软件生命周期阶段的划分。在划分软件生命周期的阶段时应该遵循的一条基本原则就是使各阶段的任务彼此间尽可能相对独立，同一阶段各项任务的性质尽可能相同，从而降低每个阶段任务的复杂程度，简化不同阶段之间的联系，有利于软件开发工程的组织管理。一般说来，软件生命周期由软件定义、软件开发和软件维护三个时期组成，每个时期又进一步划分成若干个阶段。

(1) 软件定义时期的任务

软件定义时期的任务是：①确定软件开发工程必须完成的总目标；②确定工程的可行性，导出实现工程目标应该采用的策略及系统必须完成的功能；③估计完成该项工程需要的资源和成本，并且制定工程进度表。

这个时期的工作通常又称为系统分析，由系统分析员负责完成。软件定义时期通常进一步划分成三个阶段，即问题定义、可行性研究和需求分析。

(2) 开发时期的主要任务

开发时期的主要任务是具体设计和实现在前一个时期定义的软件，它通常由下述四个阶段组成：总体设计、详细设计、编码和单元测试、综合测试。

(3) 维护时期的主要任务是使软件持久地满足用户的新需求。具体地说：

① 当软件在使用过程中发现错误时应该加以改正；

② 当环境改变时应该修改软件以适应新的环境；

③ 当用户有新要求时应该改进软件以满足用户的新需求。通常对维护时期不再进一步划分阶段，但是每一次维护活动本质上都是一次压缩和简化的定义和开发过程。

软件生命周期中软件项目管理是自始至终的，软件项目管理包括软件度量、项目估算、进度控制、人员组织、配置管理、项目计划等。

统计数据表明，大多数软件开发项目的失败，并不是软件开发技术方面的原因。它们的失败是由于不适当的管理造成的。遗憾的是，尽管人们对软件项目管理重要性的认识有所提高，但在软件管理方面的进步远比在设计方法学和实现方法学上的进步小，至今还提不出一套管理软件开发的通用指导原则。

3. 软件生命周期模型

任何软件都是从最模糊的概念开始的。从概念提出的那一刻开始，软件产品就进入了软件生命周期。在经历需求分析、系统设计、实现、部署后，软件将被使用并进入维护阶

段，直到最后由于缺少维护费用而逐渐消亡。这样的一个过程，称为生命周期模型(Life Cycle Model)。

典型的几种生命周期模型包括瀑布模型、迭代式模型、快速原型模型、增量模型、螺旋模型等。

1) 瀑布模型

瀑布模型由于酷似瀑布而闻名。在该模型中，首先确定需求，并接受客户和软件质量保证(Software Quality Assurance，SQA)小组的验证。然后拟定规格说明，同样通过验证后，进入计划阶段。可以看出，瀑布模型中至关重要的一点是：只有当一个阶段的文档已经编制好并获得软件质量保证小组的认可后，才可以进入下一个阶段。这样，瀑布模型通过强制性的要求提供规约文档来确保每个阶段都能很好地完成任务。但是实际上往往难以办到，因为整个模型几乎都是以文档驱动的，这对于非专业的用户来说是难以阅读和理解的。但对于应用结构化软件开发方法的大型软件系统的开发，瀑布模型有其天生的优势，如图 1-1 所示。

2) 迭代式模型

迭代式模型是 RUP(Rational Unified Process，统一软件开发过程，简称统一软件过程)推荐的周期模型。在 RUP 中，迭代就是为了完成一定的阶段性目标而从事的一系列开发活动，在每个迭代开始前都要根据项目当前的状态和所要达到的阶段性目标制定迭代计划，整个迭代过程包含需求分析、设计、实施(编码)、部署、测试等各种类型的开发活动，迭代完成之后需要对迭代完成的结果进行评估，并以此为依据来制定下一次迭代的目标，如图 1-2 所示。

图 1-1　瀑布模型示意图

图 1-2　迭代式模型示意图

3) 快速原型模型

快速原型(Rapid Prototype)模型在功能上等价于产品的一个简单原型。瀑布模型的缺点就在于不够直观，快速原型法就解决了这个问题。一般来说，根据客户的需要在很短的时间内满足用户的最迫切需要，完成一个可以演示的产品。这个产品只是实现部分的(最重要的)功能。它最重要的目的是确定用户的真正需求，如图 1-3 所示。

4) 增量模型

增量模型是一种非整体开发的模型，是瀑布模型的顺序特征和快速原型模型的迭代特征相结合的产物。该模型具有较大的灵活性，适合于软件需求不明确、设计方案有一定风

险的软件项目。

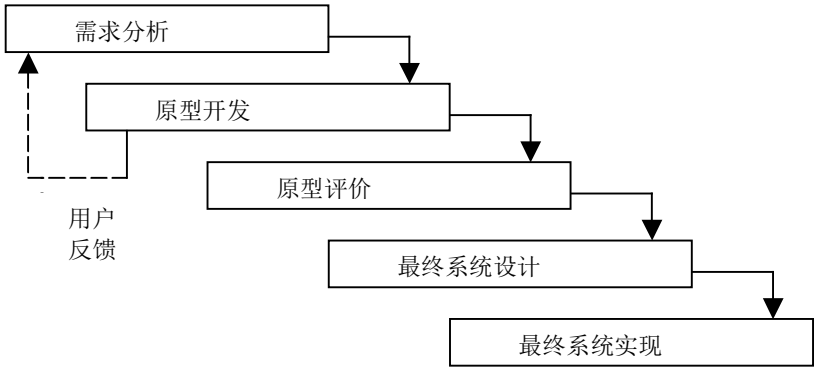


图 1-3　快速原型模型示意图

增量模型的特点是：在前面增量的基础上开发后面的增量，每个增量的开发可用瀑布或快速原型模型迭代的思路。其优点是：如果在项目既定的商业要求期限不可能找到足够的开发人员，这种情况下增量模型显得特别有用。早期的增量可以由少量的人员实现。同时，增量模型可以规避技术风险，如图 1-4 所示。

图 1-4　增量模型示意图

5) 螺旋模型

螺旋模型的基本思想是，使用原型及其他方法以尽可能地降低风险。理解这种模型的一种简易方法是，把它看作在每个阶段之前都增加了风险分析过程的快速原型模型，如图 1-5 所示。

螺旋模型将瀑布模型与快速原型模型结合起来，加入了两种模型均忽略的风险分析，弥补了这两种模型的不足。螺旋模型是一种风险驱动的模型。它将开发过程分为几个螺旋周期，每个螺旋周期大致和瀑布模型相符合。螺旋模型适合于大型软件的开发。

螺旋模型是一种迭代式模型，每迭代一次，螺旋线就前进一周。当项目按照顺时针方向沿螺旋移动时，每一个螺旋周期包含了风险分析，并且按以下 4 个步骤来进行：

(1) 确定目标，选定方案，设定约束条件，选定完成本周期所定目标的策略。

(2) 分析该策略可能存在的风险。必要时通过建立一个原型来确定风险的大小，然后据此决定是按原定目标执行，还是修改目标或终止项目。

(3) 在排除风险之后，实现本螺旋周期的目标。例如，第一圈可能产生产品的规格说明，第二圈可能产生产品设计等。

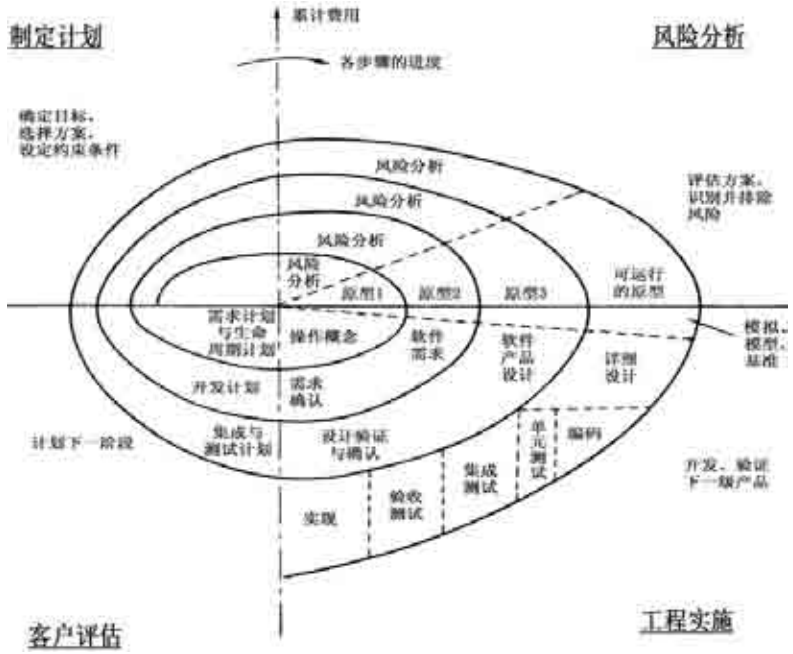


图 1-5　螺旋模型示意图

(4) 最后一步是评价前一步的结果，并且计划下一轮的工作。

螺旋模型的优点是：结合瀑布模型和原型模型的优点，风险分析可使一些极端困难的问题和可能导致费用过高的问题被更改或取消。

螺旋模型的缺点是：螺旋模型开发的成败，很大程度上依赖于风险评估的成败。需要开发人员具有相当丰富的风险评估经验和专业知识。

螺旋模型的一般使用场合：需求不能完全确定，同时又存在技术、资金或开发时间等风险因素的大型开发项目。

软件生命周期模型的发展实际上体现了软件工程理论的发展。在早期，软件的生命周期处于无序、混乱的情况。一些人为了能够控制软件的开发过程，就把软件开发严格地区分为多个不同的阶段，并在阶段间加上严格的控制和审查，确保质量。否则，就像图 1-6 所示的那样，在软件生命周期中发现问题和解决问题而付出的代价，将会随着时间和阶段的变化成倍或呈数量级增加。

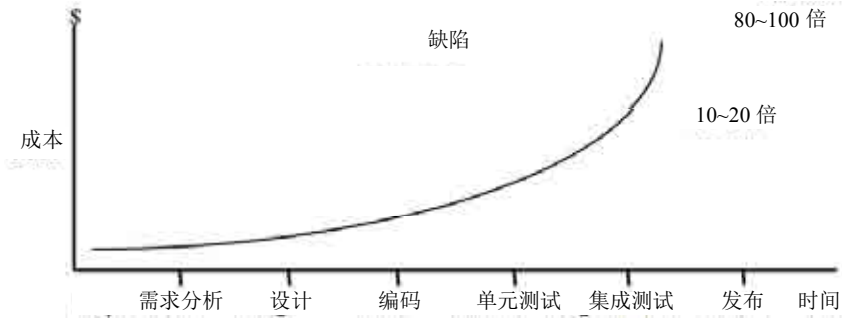


图 1-6　引入同一变化付出的代价随时间和阶段变化的趋势

1.3.3　敏捷开发过程

自从 20 世纪 70 年代软件工程产生以来，人们提出了很多软件开发方法，这些方法大都强调软件开发过程中必须遵守某些严格的规定。而且随着技术的发展，提出了在对需求和技术不断变化的情况下实现快速软件开发的要求。在 20 世纪 90 年代后期，一些软件开发人员开始强调灵活性在软件开发过程中所发挥的作用，提出一系列新的软件开发方法，这就是敏捷方法或敏捷开发。

1. 敏捷开发

敏捷开发被认为是软件工程的一次重要的发展。它强调软件开发应当能够对未来可能出现的变化和不确定性做出全面反应。

敏捷开发的总体目标是通过尽可能早地、持续地对有价值软件的交付，使客户满意。很多客户都有一些随着时间变化的业务需求，不仅表现在新发现的需求上，也表现在对市场变化做出反应的需求上。通过在软件开发过程中加入灵活性，敏捷开发可以使用户能够在开发周期的后期增加或改变需求。

敏捷开发主要是用于需求模糊或快速变化的前提下、小型开发团队的软件开发活动。敏捷开发能够在保证软件开发成功的前提下，尽量减少开发过程中的活动和产品，做到刚刚好，从而在满足所需的软件质量要求的前提下，力求提高开发的效率。

敏捷开发强调：

- 注重个人及互动胜于过程和工具
- 注重可用的软件胜于详尽的文档
- 注重客户协作胜于合同谈判
- 注重响应变化胜于恪守计划

敏捷开发是一种以人为核心、迭代、循序渐进的开发方法。在敏捷开发中，软件项目的构建被切分成多个子项目，各个子项目的成果都经过测试，具备集成和可运行的特征。换言之，就是把一个大项目分为多个相互联系，但也可独立运行的小项目，并分别完成，在此过程中软件一直处于可使用状态。

敏捷开发是针对传统的瀑布开发模型的弊端而产生的一种新的开发模式，是一种面临迅速变化的需求快速开发软件的能力，目标是提高开发效率和响应能力。为达到该目标，敏捷开发定义了 12 条原则：

(1) 最优先要做的是通过尽早、持续交付有价值的软件来使客户满意。

(2) 即使在开发的后期，也不拒绝需求变更(敏捷过程利用变更为客户创造竞争优势)。

(3) 经常交付可工作的软件(交付的间隔可以从几个星期到几个月，交付的时间间隔越短越好)。

(4) 在整个项目开发期间，业务人员和开发人员最好天天在一起工作。

(5) 强化激励机制，为受激励的个人构建项目(为他们提供所需的环境和支持，并且信任他们能够完成工作)。

(6) 在团队内部，最富有效果和效率的信息传递方法是面对面交谈。

(7) 可工作的软件是进度的首要度量标准。

(8) 敏捷过程提倡可持续的开发速度(责任人、开发者和用户应该保持一种长期、稳定

的开发速度)。

(9) 不断地关注优秀的技能和好的设计，增强敏捷能力。

(10) 尽量简化要做的工作。

(11) 好的架构、需求和设计出自于组织团队自身。

(12) 每隔一定时间，团队要反省如何更有效地工作，并相应地调整自己的行为。

敏捷方法有很多具体的方法，常用的敏捷方法有七种：

1) XP

XP(eXtreme Programming，极限编程)的思想源自 Kent Beck 和 Ward Cunningham 在软件项目中的合作经历。XP 注重的核心是沟通、简明、反馈和勇气。因为知道计划永远赶不上变化，XP 不需要开发人员在软件开始初期做出很多的文档。XP 提倡测试先行，以将后面出现 bug 的概率降至最低。

2) SCRUM

SCRUM 是一种迭代式的增量化过程，用于产品开发或工作管理。它是一种可以集合各种开发实践的经验化过程框架。SCRUM 中发布产品的重要性高于一切。

该方法由 Ken Schwaber 和 Jeff Sutherland 提出，旨在寻求充分发挥面向对象和构件技术的开发方法，是对迭代式面向对象方法的改进。

3) Crystal Methods

Crystal Methods(水晶方法族)由 Alistair Cockburn 在 20 世纪 90 年代末提出。之所以是一个方法系列，是因为他相信不同类型的项目需要不同的方法。虽然水晶方法族不如 XP 那样的高产出效率，但会有更多的人能够接受并遵循它。

4) FDD

FDD(Feature-Driven Development，特性驱动开发)由 Peter Coad、Jeff de Luca、Eric Lefebvre 共同开发，是一套针对中小型软件开发项目的开发模式。此外，FDD 是一个模型驱动的快速迭代开发过程，它强调的是简化、实用、 易于被开发团队接受，适用于需求经常变动的项目。

5) ASD

ASD(Adaptive Software Development，自适应软件开发)由 Jim Highsmith 在 1999 年正式提出。ASD 强调开发方法的适应性(Adaptive)，这一思想来源于复杂系统的混沌理论。ASD 不像其他方法那样有很多具体的实践做法，它更侧重为 ASD 的重要性提供最根本的基础，并从更高的组织和管理层次来阐述开发方法为什么要具备适应性。

6) DSDM

DSDM(Dynamic System Development Method，动态系统开发方法)是众多敏捷开发方法中的一种，它倡导以业务为核心，快速而有效地进行系统开发。实践证明，DSDM 是成功的敏捷开发方法之一。在英国，由于其在各种规模的软件组织中的成功，已成为应用最为广泛的快速应用开发方法。

DSDM 不但遵循敏捷方法的原理，而且非常适合那些前期有较好传统开发方法基础的软件组织。

7) 轻量型 RUP

RUP 其实是一个过程的框架，它可以包容许多不同类型的过程，Craig Larman 极力主张以敏捷方式使用 RUP。他的观点是：目前如此多的努力以推进敏捷方法，只不过是在接受能被视为 RUP 的主流面向对象开发方法而已。

2. 敏捷开发过程

敏捷可用于任何软件过程，实现要点是将软件过程设计为如下方式：允许项目团队调整并合理安排任务，理解敏捷开发方法的易变性并制定计划，精简并维持最基本的工作产品，强调增量交付策略，快速向客户提供适应产品类型和运行环境的可运行软件。因此，敏捷过程很容易适应变化并迅速做出自我调整，在保证质量的前提下，做到文档、度量适度。

下面我们以极限编程(eXtreme Programming，XP)为例，介绍敏捷开发过程。

XP 使用面向对象方法作为推荐的开发范型。XP 包含了策划、设计、编码和测试 4 个框架活动的规则和实践。图 1-7 描述了 XP 过程，并指出了与各框架活动相关的关键概念和任务。

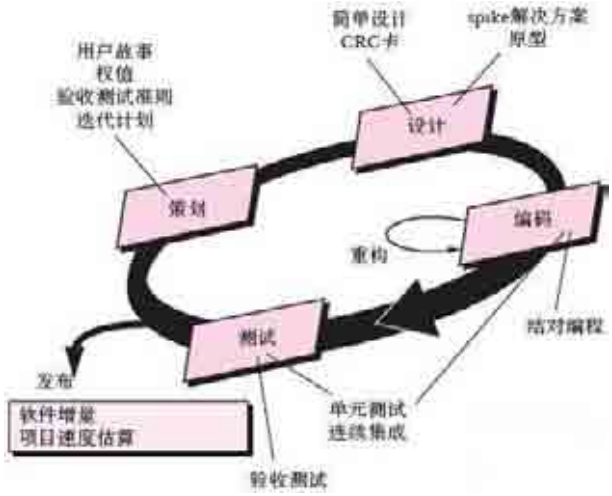


图 1-7　极限编程过程

1) 策划

策划活动开始于建立一系列描述待开发软件的必要特征与功能要求。每个功能要求由客户书写并置于一张索引卡上，客户根据对应特征或功能的全局业务价值标明权值(即优先级)。XP 团队成员评估每一个功能并给出以开发周数为度量单位的成本。如果某个功能的开发成本超过 3 周，将请客户对该功能进一步细分，重新赋予权值并计算成本。新的功能要求可以在任何时刻书写。

客户和 XP 团队共同决定如何把功能分组，并置于 XP 团队将要开发的下一个发行版本中。一旦形成关于一个发行版本的基本承诺，XP 团队将按以下三种方式之一对待开发的功能进行排序：

① 所有选定功能将在几周之内尽快实现；

② 具有最高价值的功能将移到进度表的前面并首先实现；

③ 高风险功能将首先实现。

项目的第一个发行版本发布之后，XP 团队计算项目的速度。简言之，项目速度是第一个发行版本中实现的用户功能个数。项目速度将用于帮助建立后续发行版本的发布日期和进度安排；确定是否对整个开发项目中的所有功能有过分承诺。一旦发生过分承诺，就调整软件发行版本的内容或者改变最终交付日期。

在开发过程中，客户可以增加功能、改变功能要求的权值、分解或去掉功能。接下来由 XP 团队重新考虑所有剩余的发行版本，并相应修改计划。

2) 设计

XP 设计严格遵循保持简洁(Keep It Simple，KIS)原则，使用简单而不是复杂的表述。另外，设计为功能提供不多也不少的实现原则，不鼓励额外功能性设计。

XP 鼓励使用类、责任、协作者(Class-Responsibility-Card，CRC)卡作为有效机制，在面向对象语境中考虑软件、CRC 卡的确定，组织和当前软件增量相关的对象和类。CRC 卡也是作为 XP 过程一部分的唯一的设计工作产品。

如果在某个功能设计中碰到困难，XP 推荐立即建立这部分设计的可执行原型，实现并评估设计原型，目的是在真正的实现开始时降低风险，对可能存在设计问题的功能确认最初的估计。

3) 编码

在功能开发和基本设计完成之后，团队不应直接开始编码，而是开发一系列用于检测本次(软件增量)发布的包括所有功能的单元测试。一旦建立起单元测试，开发者就可以更集中精力于必须实现的内容以通过单元测试。不需要添加任何额外的东西(保持简洁)。一旦编码完成，就可以立即完成单元测试，可由此向开发者提供即时反馈。

XP 编码活动中的关键概念之一是结对编程。XP 推荐两个人面对同一台计算机共同为一个功能开发代码。这一方案提供实时解决问题和实时质量保证的机制，同时也使开发者能集中精力于手头的问题。实施中不同成员担任的角色略有不同。例如，一名成员考虑特定设计的详细编码实现，而另一名成员确保编码遵循特定的标准，生成的代码符合该功能要求的接口设计。

结对的两人完成所开发代码和其他工作集成。在有些情况下，这种集成工作由集成团队按日实施。在另外一些情况下，结对者自己负责集成，这种“连续集成”策略有助于避免兼容性和接口问题，建立能及早发现错误的“冒烟测试”环境。

4) 测试

在编码开始之前建立单元测试是 XP 方法的关键因素。所建立的单元测试应当使用一个可以自动实施的框架，这种方式支持代码修改之后的即时回归测试策略。

一旦将个人的单元测试组织到一个“通用测试集”，每天就可以进行系统的集成和确认测试。这可以为 XP 团队提供连续的进展指示，也可在一旦发生问题的时候及早提出预警。

XP 验收测试也称为客户测试，由客户确定，将着眼于客户可见的、可评审的系统级的特征和功能，验收测试根据本次软件发布中所实现的用户功能而确定。

习题和思考题

1. 简述软件的定义，软件具有什么样的特性？

2. 什么是软件危机？产生软件危机的原因是什么？如何消除？

3. 什么是软件工程？什么是软件生命周期？它们都包含哪些内容？

4. 软件工程涉及哪些概念和名词？它们的关系如何？如何解释？

5. 运用软件工程的理论、技术和方法能够为我们解决什么问题？

6. 如何理解软件开发工具和软件工程环境在软件工程中的作用？

7. 软件项目管理涉及哪些方面，它的必要性是什么？

8. 软件复审的目的是什么，我们怎样进行软件技术审查？软件技术审查和管理复审的作用是什么？

9. 什么是软件开发模型？软件开发模型有几种？各有什么特点？

10. 什么是敏捷开发？敏捷开发有哪些方法？其基本过程是怎样的？