工业和信息化普通高等教育“十三五”规划教材立项项目
21世纪高等教育计算机规划教材

C软件工程

Introduction to Software Engineering

钟珞 袁胜琼 袁景凌 李琳 主编

- 由浅入深，通俗易懂，符合大学生学习特点
- 把工程理念贯彻到软件开发全过程
- 引入案例，增强对软件工程情景的理解和认知

中国工信出版集团
人民邮电出版社
POSTS & TELECOM PRESS

工业和信息化普通高等教育“十三五”规划教材立项项目
21世纪高等教育计算机规划教材

软件工程

Introduction to Software Engineering

钟珞 袁胜琼 袁景凌 李琳 主编

人民邮电出版社
北京

图书在版编目（CIP）数据

软件工程 / 钟珞等主编. -- 北京 : 人民邮电出版社, 2017.1
21世纪高等教育计算机规划教材
ISBN 978-7-115-44649-7

Ⅰ. ①软… Ⅱ. ①钟… Ⅲ. ①软件工程－高等学校－教材 Ⅳ. ①TP311.5

中国版本图书馆CIP数据核字(2017)第005881号

内 容 提 要

本书舍弃传统软件工程的一些烦琐理论，着重叙述面向对象软件工程的基本原理和基本概念，并对敏捷开发方法以及软件模式等重用技术以较多篇幅加以阐述。本书把工程理念贯彻到软件开发全过程，在阐述软件工程技术方法的同时，引入案例，增强读者对软件工程情景的理解和认知，并以数字传播工程为契机，探讨面向特定领域的软件工程。

本书可以作为计算机科学以及软件工程专业本科教学的教材，也可作为软件开发人员或对软件工程感兴趣的人员自学的参考资料。

◆ 主　　编　钟　珞　袁胜琼　袁景凌　李　琳
责任编辑　吴　婷
责任印制　沈　蓉　彭志环

◆ 人民邮电出版社出版发行　　北京市丰台区成寿寺路 11 号
邮编　100164　　电子邮件　315@ptpress.com.cn
网址　http://www.ptpress.com.cn
北京圣夫亚美印刷有限公司印刷

◆ 开本：787×1092　1/16
印张：15.75　　2017 年 1 月第 1 版
字数：410 千字　　2017 年 1 月北京第 1 次印刷

定价：39.80 元

读者服务热线：(010)81055256　印装质量热线：(010)81055316
反盗版热线：(010)81055315

前言

随着计算机技术的飞速发展，软件开发新技术、新方法层出不穷，软件管理日益复杂。为摆脱软件危机，软件工程从20世纪60年代末开始迅速发展起来，现在已经成为计算机科学技术的一个重要的独立分支。20世纪90年代以来，软件工程从方法论的角度为开发人员和管理人员提供了可见的结构和有序的思考。此外，基于大量成功软件总结提炼出来的设计经验和开发模式，使得软件开发人员可以充分利用设计模式、框架、组件等进行重用开发，并最终将软件以服务的形式提供给用户使用。软件工程的相关理论与技术，得到了不断的完善以及广泛应用。

软件工程现在是一个非常大的领域，任何一本书都不可能涵盖软件工程的所有内容。本书在回顾近年来软件开发的重要技术，尤其是基于Web应用程序的开发技术的基础上，着重叙述了面向对象软件工程的基本原理和概念，对敏捷开发方法和软件框架、软件模式等重用技术给予了更多的篇幅加以描述。在阐述软件工程相关理论知识的同时，强调具体案例分析。本书给出较完整的开发案例，使软件工程的理论和方法更易于理解、模仿和应用。此外，还以数字传播工程为契机，探讨了面向特定领域的软件工程的发展。

本书编写中舍弃传统软件工程的一些烦琐理论，代之以简洁实用的软件工程新知识、新方法，增加了教材的实用性和可读性。通过对软件工程以及软件开发热点问题展开讨论，学生能把握前沿，尽早确定研究方向。本书以案例分析贯穿全书，适宜教师开展项目式教学。因此，本书适合作为计算机科学以及软件工程专业的本科教学的教材，也可以作为软件开发人员或对软件工程感兴趣的人员自学的参考资料。

本书主要包含五个部分：第一部分是对软件工程的一般性介绍，包括软件及软件工程过程等的基本概念；第二部分主要介绍面向对象的设计及设计模式的使用；第三部分介绍团队开发管理和敏捷开发方法；第四部分介绍面向特定领域的软件工程——数字传播工程的兴起；第五部分给出具体、详尽的开发案例。

本书作者一直以来从事软件工程课程的教学工作，积累了丰富的教学经验和教学心得，并有大量软件开发设计实践经验，对软件工程技术及其发展前沿有较深刻的认识。

本书由钟珞、袁胜琼、袁景凌和李琳主编，参加编写的有梁媛、朱阁、陈明、柳杨、孙悦清等。另外，刘永坚教授提供了数字传播工程相关的实例，在此表示感谢！

恳请专家学者提出意见和建议，以期做进一步的完善。

编　者

2016年10月

目 录

第1章 软件工程概述

软件工程是用科学理论来指导软件开发、管理、标准化、自动化的过程，对于培养学生的软件素质、提高学生的软件开发能力和软件项目管理能力具有重要的意义。目前，比较公认的软件工程（Software Engineering）的定义是美国电气与电子工程师协会（IEEE）给出的：将系统化的、严格约束的、可量化的方法应用于软件的开发、运行和维护，即将工程化应用于软件，并研究这个过程中的方法。

1.1 软件及其特性

世界上第一个写软件的人是阿达（Augusta Ada Lovelace）。他在19世纪60年代尝试为巴贝奇（Charles Babbage）的机械式计算机编写软件。尽管限于当时的制造条件，巴贝奇最终没有造成理想中的计算机，但巴贝奇和阿达对后来计算机技术的诞生和发展同样产生了深远的影响，他们的名字永远载入了计算机发展的史册。

在20世纪中叶，软件伴随着第一台电子计算机的问世而诞生。以编写软件为职业的人也开始出现，他们多是经过训练的数学家和电子工程师。20世纪60年代，美国大学里开始出现授予计算机专业的学位，教学生如何编写软件。软件产业从零开始起步，在短短的50多年的时间里迅速发展成为推动人类社会发展的龙头产业，并造就了一批百万、亿万富翁。随着信息产业的发展，软件对人类社会越来越重要。

现在的世界正在进入一个“软件无处不在”的时代，我们每天的生活，时刻都离不开这样或那样的软件。软件（software）是计算机系统中与硬件（hardware）相互依存的另一部分，它包括程序（program）、相关数据（data）及其说明文档（document）。其中，程序是按照事先设计的功能和性能要求执行的指令序列，数据是程序能正常操纵信息的数据结构，文档是与程序开发维护和使用有关的各种图文资料。

Fred Brooks教授，是软件工程领域非常有影响力的人物。他曾经担任IBM OS360系统的项目经理，在计算机体系结构、操作系统以及软件工程方面做出了杰出的贡献，并因此于1999年获得了图灵奖。

Brooks教授在1987年发表了一篇题为“没有银弹（No Silver Bullet）”的文章。在这篇文章中他指出：“软件具有复杂性、一致性、可变性和不可见性等固有的内在特性，这是造成软件开发困难的根本原因。”

（1）软件的复杂性。软件是复杂的，是人类思维和智能的一种延伸，它比任何以往人类的创

造物都要复杂得多。今天，我们已经进入云计算时代。在互联网的集群环境下，系统规模更大更复杂，可以说，软件是人类有史以来生产的复杂度最高的工业产品。这种复杂性，对软件工程师提出了很高的要求，给软件开发管理和质量保证带来了很多困难。

（2）软件的一致性。软件不能独立存在，需要依附于一定的环境（如硬件、网络以及其他软件）。因此，软件必须遵从人为的惯例并适应已有的技术和系统，随着接口的不同而改变。

（3）软件的可变性。软件经常会遭受到持续的变更压力。相对于建筑和飞机等工程制品来说，软件的变更似乎更加频繁，这也许是由于建筑和飞机修改成本太高所致。人们总是以为软件很容易修改，但是却忽视了修改带来的副作用。软件不断变化，每一次的修改都会造成故障率的升高，同时也可能给软件的结构带来破坏。尽管如此，成功的软件都是会发生演化的。软件的可变性，给开发带来了很多难题，但同时也给软件本身带来了生命力。

（4）软件的不可见性。软件是一种逻辑产品，看不见摸不着，它的客观存在不具有空间的形体特征，因此缺少合适的几何表达方式。这种不可见性，不仅限制了软件的设计过程，同时严重地阻碍了人与人之间的相互交流，从而对开发过程的管理造成很大困难。

综上所述，复杂性、一致性、可变性和不可见性，是软件的本质特性。这些特性，使得软件开发的过程变得难以控制，开发团队如同在焦油坑里挣扎的巨兽，挣扎得越猛烈，焦油纠缠得越紧，最后有可能沉没到坑底。因此，我们需要寻找解决问题的有效方法，以保证软件开发过程的高效、有序和可控。

1.2 软件工程的产生与发展

1.2.1 软件危机

20 世纪 60 年代末至 70 年代初，“软件危机”一词在计算机界广为流传。事实上，软件危机几乎从计算机诞生的那一天起就出现了，只不过到了 1968 年，北大西洋公约组织的计算机科学家在联邦德国召开的国际学术会议上第一次提出了“软件危机”（software crisis）这个名词。

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严重问题。这类问题绝不仅仅是“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类问题。

美国 Standish 集团是一家专门跟踪软件项目的研究机构。该机构对 1994—2010 年期间开发的软件项目进行了调查统计，结果如图 1-1 所示。

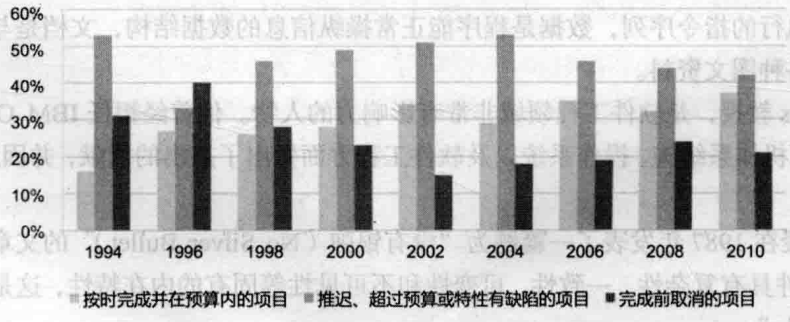

图 1-1 美国 Standish 集团调查报告

研究表明，软件项目的平均成功率大概在 30%。大概 50%的项目超出预算和最后期限，或者存在特定缺陷。另外，还有 20%左右的项目彻底失败。最终完成的项目也总是存在着错误多、性能低、不可靠、不安全等质量问题。

软件的错误可能导致巨大的财产损失。1996 年 6 月 4 日，欧洲航天局 Ariane 5 火箭在发射 37 秒之后，偏离了飞行路径，突然发生爆炸，火箭上载有价值数亿美元的通信卫星。事后的调查显示，导致事故的原因是程序中试图将 64 位浮点数转化成 16 位整数的时候，产生了溢出，而系统又缺乏对数据溢出的错误处理程序。

Windows Vista 系统，是曾经被微软公司寄予厚望的一个桌面操作系统，也是微软公司历史上最艰难曲折、开发时间最长的一个项目。这个系统从 2001 年开始研发，整个过程历时 5 年，耗资数十亿美元，代码规模超过 5000 万行。由于系统过于庞杂，给整个开发带来了很大的困难，很多的时间用在了互相沟通和重新决策上。本应该在 2003 年面世的 Vista 系统，一再地推迟，最后在取消了一些高级功能之后，于 2006 年 11 月正式发布。即使这样，Vista 系统在面世之后，仍然暴露出运行效率低、兼容性差、死机频繁等严重缺陷。

显然，软件开发一直面临着诸多的挑战，主要表现在以下 4 个方面。

（1）客户不满意。软件产品的交付质量难以保证，许多功能不是用户所需要的，用户在使用的过程中遭遇很多 Bug。

（2）项目过程失控。由于客户需求的不确定性和持续的变化，给整个开发过程带来了不可控性。

（3）风险与成本问题。开发团队专注于技术，忽视对风险的管理，从而造成整个开发成本的超支。

（4）无力管理团队。无法评估开发人员的能力以及工作进度。如何提升团队的能力和效率，一直是软件开发中存在的难题。

之所以出现软件危机，其主要原因一方面是与软件本身的特点有关，另一方面是与软件开发和维护的方法不正确有关。

软件的特点前面已经有一个简单介绍。软件开发和维护的不正确方法主要表现为：忽视软件开发前期的需求分析；开发过程没有统一、规范的方法论的指导，文档资料不齐全，忽视人与人的交流；忽视测试阶段的工作，提交给用户的软件质量差；轻视软件的维护。这些大多数是软件开发过程管理上的原因。

1968 年秋季，NATO（North Atlantic Treaty Organization，北大西洋公约组织）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”（software engineering）这个概念。当时的会议报告中提到，“我们特意选择‘软件工程’这个颇具争议性的词，是为了暗示这样一种意见：软件的生产，有必要建立在某些理论基础和实践指导之上。在工程学的某些成效卓著的分支中，这些理论基础和实践指导早已成为了一种传统。”

1.2.2　软件工程的发展

统计数据表明，大多数软件开发项目的失败，并不是由于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，至今还提不出管理软件开发的通用指导原则。

在软件的长期发展中，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。软件工程的发展大概经历了 4 个阶段。

（1）1968 年以前，属于软件工程的史前阶段。在这个时期，没有什么工程化的开发方法可循，更多的是个人作坊式的开发。当时的软件几乎都是为每个具体应用而专门编写的，编写者和使用者往往是同一个或同一组人。这些个体化的软件设计环境，使软件设计成为在人们头脑中进行的一个隐含过程，最后除了程序清单外，没有其他文档资料保存下来。于是 20 世纪 60 年代末，爆发了软件危机。

（2）从 1968 年开始，一直到 20 世纪 80 年代末，软件工程进入了一个新的时期。1968 年首次提出了“软件工程”的概念。瀑布模型成为软件开发的经典模型，整个软件开发过程被划分成需求、设计、编码、测试等不同阶段，并且这些阶段都是严格按照线性的方式执行的。

（3）从 1983 年到 1995 年，人们逐步意识到过程质量对产品质量的重大影响。这个时期面向对象的方法和软件过程改进运动逐渐盛行，提出了 CMM/ISO9000/SPICE 等质量标准体系。

（4）从 20 世纪 90 年代至今，互联网技术和应用迅速发展。为了应对需求变化和快速交付的需要，人们开始尝试一种新型的敏捷开发方法。这种方法采用迭代和增量的开发过程，强调更紧密的团队协作。目前，敏捷开发方法已经广泛地应用于软件企业之中，给软件行业带来了巨大的变化。

今天，尽管“软件危机”并未被彻底解决，但软件工程已经成为现代软件产业一个关键的技术，并且正在向成熟发展，在未来对网络时代的软件开发将有更大的推动力。

1.3 软件工程的基本概念

1.3.1 什么是软件工程

所谓的“工程”，就是创造性地运用科学原理设计和实现建筑、机器、装置或生产过程，或者是在实践中使用一个或多个这些实体，或者是实现这些实体的过程。

远古时期，人们互相协作建造了不少工程奇迹，比如希腊雅典的帕特农神庙、古罗马帝国的罗马水道、中国的长城等。我们可以想象这些工程在设计和建造的过程中一定涉及了大量的计算、计划、各类角色的协作，以及成百上千的人、动物、机械经年累月的劳作。这些因素在后来出现的诸如化学工程、土木工程等各类“工程”中依然存在。

顾名思义，软件工程就是把工程化的方法应用到软件之中，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去进行软件的开发和维护的学科。人们曾经对“软件工程”给过许多定义，下面是两个比较典型的。

1968 年 NATO 会议上首次提出：“软件工程是为了经济地获得可靠的和能在实际机器上高效运行的软件，而建立和使用完善的工程原理。”这个定义不仅指出了软件工程的目标是经济地开发出高质量的软件，而且强调了软件工程是一门工程学科，它应该建立并使用完善的工程原理。

1993 年 IEEE 进一步给出了一个更全面更具体的定义：“软件工程是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对①中所述方法的研究。”

美国南加州大学的巴里·贝姆（Barry Boehm）教授总结了国际上软件工程的发展历程：20世纪50年代的类似硬件工程（Hardware Engineering）、60年代的软件手工生产（Software Crafting）、70年代的形式化方法和瀑布模型（Formality and Waterfall Processes）、80年代的软件生产率和可扩展性（Productivity and Scalability）、90年代的软件并发和顺序进程（Concurrent vs. Sequential Processes）以及21世纪初的软件敏捷性和价值（Agility and Value）。我国北京大学杨芙清院士也系统地回顾了起步于1980年的中国软件工程的研究与实践，代表性工作包括软件自动化系统、XYZ系统4、MLIRF系统和青鸟工程等，都在国内外有广泛的影响。

总的来说，软件工程包括软件开发技术和软件项目管理两方面内容。其中，软件开发技术包括软件开发方法学、软件工具和软件工程环境，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。

1.3.2　软件工程的基本要素

软件工程是相当复杂的。涉及的因素很多，不同软件项目使用的开发方法和技术也是不同的，而且有些项目的开发无现成的技术，带有不同程度的试探性。一般说来，软件工程包含方法（methodologies）、工具（tools）和过程（procedures）三个关键元素。

软件方法提供如何构造软件的技术，包括与项目有关的计算和各种估算、系统和软件需求分析、数据结构设计、程序体系结构、算法过程、编码、测试以及维护等内容。软件工程的方法通常引入各种专用的图形符号以及一套软件质量的准则。概括地说，软件工程方法规定了明确的工作步骤与技术、具体的文档格式、明确的评价标准。

软件开发方法（见图1-2）的发展经历了面向过程、面向对象、面向构件和面向服务4个阶段。

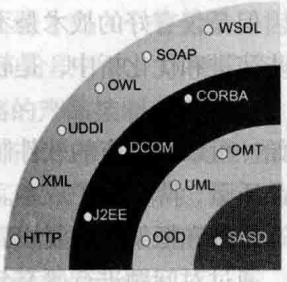

图1-2　软件开发方法

（1）面向过程。以算法作为基本构造单元，强调自顶向下的功能分解，将功能和数据进行一定程度的分离。

（2）面向对象。以类为基本程序单位，对象是类的实例化，对象之间以消息传递为基本手段。

（3）面向构件。寻求比类的粒度更大且易于复用的构件，期望实现软件的再工程。

（4）面向服务。在应用表现层次上将软件构件化，即应用业务过程由服务组成，而服务由构建组装而成。

代码封装的力度从函数到类，再到粒度更大的构件以及在应用表现层次上的服务，软件的复用程度逐步提升，开发效率也越来越高。

软件工具是人类在开发软件的活动中智力和体力的扩展和延伸，为方法和语言提供自动或半自动化的支持。软件工具最初是零散的，后来根据不同类型软件项目的要求建立了各种软件工具

箱，支持软件开发的全过程。更进一步，人们将用于开发软件的软、硬件工具和软件工程数据库（包括分析、设计、编码和测试等重要信息的数据结构）集成在一起，建立集成化的计算机辅助软件工程（Computer-Aided Software Engineering）系统，简称 CASE。

现在开源的工具非常多，贯穿于整个开发过程。具体来说，软件建模工具可以支持建立系统的需求和设计模型；软件构造工具包括程序编辑器、编译器、解释器和调试器；软件测试工具可以帮助人们分析代码质量，执行软件测试和评价产品的质量；在软件维护阶段，一些代码分析工具和重构工具，可以帮助人们理解和维护代码。除此之外，还有一些软件工程管理工具，帮助人们有效管理开发过程，控制代码的更改，支持团队进行协作开发。

软件过程贯穿于软件开发的各个环节，它定义了方法使用的顺序、可交付产品（文档、报告以及格式）的要求、为保证质量和协调变化所需要的管理以及软件开发过程各个阶段完成的标志。

软件开发过程一般包括一系列基本的开发活动，这些活动将用户的需求转化为用户满意的产品。通过对开发过程中各个活动环节质量的有效控制，来保证最终产品的质量。首先要研究和定义用户的问题；确定和分析用户的实际需求；设计整个系统的总体结构；编程实现系统的各个部分；最后，将各个部分集成起来进行测试，最终交付出用户满意的产品。除此之外，还应该包括一些开发过程管理等支持性的活动。

从内容上说，软件工程包括软件开发理论和结构、软件开发技术以及软件工程管理和规范。其中，软件开发理论和结构包括程序正确性证明理论、软件可靠性理论、软件成本估算模型、软件开发模型以及模块划分原理，软件开发技术包括软件开发方法学、软件工具以及软件环境，软件工程管理和规范包括软件管理（人员、计划、标准、配置）以及软件经济（成本估算、质量评价）。即软件工程可分为理论、结构、方法、工具、环境、管理、何规范等。理论和结构是软件开发的基础；方法、工具、环境构成软件开发技术，好的工具促进方法的研制，好的方法能改进工具；工具的集合构成软件开发环境；管理是技术实现与开发质量的保证；规范是开发遵循的技术标准。

软件工程几十年的发展，已经积累了许多开发方法。但是仅有好的战术是不够的，还需要在实践中运用良好的开发策略。软件复用、分而治之、逐步演进和优化折中，是软件开发的四个基本策略。

（1）软件复用。构造一个新的系统，不必都从零开始。可以将已有的软件制品，直接组装或者合理修改形成新的软件系统，从而提高开发效率和产品质量，降低维护成本。软件复用也不仅仅是代码的复用，还包括对系统类库、模板、设计模式、组件和框架等的复用。

（2）分而治之。是人们处理复杂性的一个基本策略。通过对问题进行研究分析，将一个复杂的问题分解成若干个可以理解并能够处理的小问题，然后逐个予以解决。

（3）逐步演进。软件更像是一个活着的植物，其生长是一个逐步有序的过程。软件开发应该遵循软件的客观规律，不断进行迭代式增量开发，最终交付符合客户价值的产品。

（4）优化折中。软件工程师应当把优化当成一种责任，不断改进和提升软件质量。但是优化是一个多目标的最优决策，在不可能使所有目标都达到最优时，需要进行折中来实现整体的最优。

Wasserman 规范给出了对软件工程发展有重大影响的若干技术，这些技术分别是抽象、软件建模方法、用户界面原型化、软件体系结构、软件过程、软件复用、度量、工具和集成环境。其中，抽象是一种降低复杂性的处理方法；软件建模方法可以帮助工程师理解和刻画系统的分析和设计结果，便于开发人员进行沟通和交流；用户界面原型化可以克服需求难以确定的困难；软件体系结构对产品质量是至关重要的；软件过程、软件复用和度量都是工程方法的组成部分；工具和集成环境对于提高软件开发效率是必不可少的。

Wasserman 指出，上述八个技术变化中的任何一个都对软件开发过程有着重大的影响，它们合在一起，改变了我们的工作方式。

在软件工程中，软件的可靠性是软件在所给条件下和规定时间内，能完成所要求的功能的性质。软件工程的软件可靠性理论及其评价方法，是贯穿整个软件工程各个阶段所必须考虑的问题。

软件工程的目标在于研究一套科学的工程化方法，并与之相适应，发展一套方便的工具与环境，供软件开发者使用。

1.3.3　软件工程的基本原理

自从 1968 年提出“软件工程”这一术语以来，研究软件工程的专家学者们陆续提出了许多关于软件工程的准则或信条。美国著名的软件工程专家 Boehm 综合这些专家的意见，并总结了 TRW 公司多年的开发软件的经验，于 1983 年提出了软件工程的七条基本原理。

Boehm 认为，这七条基本原理是确保软件产品质量和开发效率的原理的最小集合。它们是相互独立的，是缺一不可的最小集合；同时，它们又是相当完备的。下面简要介绍软件工程的七条基本原理。

1. 用分阶段的生命周期计划严格管理

这一条是吸取前人的教训而提出来的。统计表明，50%以上的失败项目是由于计划不周而造成的。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。这条原理意味着，应该把软件生命周期分成若干阶段，并相应制定出切实可行的计划，然后严格按照计划对软件的开发和维护进行管理。 Boehm 认为，在整个软件生命周期中应指定并严格执行项目概要计划、里程碑计划、项目控制计划、产品控制计划、验证计划、运行维护计划六类计划。

2. 坚持进行阶段评审

统计结果显示，大部分错误是设计错误，大约占 63%；错误发现得越晚，改正它要付出的代价就越大，相差大约 2 到 3 个数量级。因此，软件的质量保证工作不能等到编码结束之后再进行，应坚持进行严格的阶段评审，以便尽早发现错误。

3. 实行严格的产品控制

开发人员最痛恨的事情之一就是改动需求。但是实践告诉我们，需求的改动往往是不可避免的。这就要求我们要采用科学的产品控制技术来顺应这种要求，也就是要采用变动控制，又叫基准配置管理。当需求变动时，其他各个阶段的文档或代码随之相应变动，以保证软件的一致性。

4. 采纳现代程序设计技术

从 20 世纪六七十年代的结构化软件开发技术到最近的面向对象技术，从第一、第二代语言到第四代语言，人们已经充分认识到：方法比气力更有效。采用先进的技术既可以提高软件开发的效率，又可以减少软件维护的成本。

5. 结果应能清楚地审查

软件是一种看不见、摸不着的逻辑产品。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产品标准，以使所得到的标准能清楚地审查。

6. 开发小组的人员应少而精

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，开发人员应该少而精。这一条基本原理基于两点原因：高素质开发人员的效率比低素质开发人员的效率要高几倍到几十倍，开发工作中犯的错误也要少得多；当开发小组为 N 人时，可能的通信信道为 N(N–1)/2，可见随着

人数 N 的增大，通信开销将急剧增大。

7. 承认不断改进软件工程实践的必要性

遵从上述七条基本原理，就能够较好地实现软件的工程化生产。但是，它们只是对既有经验的总结和归纳，并不能保证赶上技术不断前进发展的步伐。因此，Boehm 提出应把承认不断改进软件工程实践的必要性作为软件工程的第七条基本原理。根据这条原理，不仅要积极采纳新的软件开发技术，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。这些数据既可以用来评估新的软件技术的效果，也可以用来指明必须着重注意的问题及应该优先进行研究的工具和技术。

1.4 软件工程的现状与发展趋势

1.4.1 敏捷开发

敏捷软件开发（agile software development），又称敏捷开发，是从 20 世纪 90 年代开始逐渐引起广泛关注的一些新型软件开发方法，它是一类轻量级的软件开发方法，提供了一组思想和策略来指导软件系统的快速开发并响应用户需求的变化。

随着软件交付周期的日益加快，迭代式敏捷开发方法渐成标准，已经成为大多数软件开发团队的必选项。迭代对整个团队的需求、架构、协同及测试能力都提出了更高的要求，敏捷可以被看成是迭代式开发的一种导入方式，只不过敏捷的范围其实比迭代化开发更大一些。

简单地说，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。

敏捷方法有很多，包括 Scrum、极限编程、功能驱动开发以及统一过程（RUP）等。这些方法本质上是一样的。敏捷开发小组主要的工作方式可以归纳为以下 5 种。

1. 敏捷小组作为一个整体工作

项目取得成功的关键在于，所有项目参与者都把自己看成朝向一个共同目标前进的团队的一员。一个成功的敏捷开发小组应该具有“我们一起参与其中”的思想，“帮助他人完成目标”这个理念是敏捷开发的根本管理文化。当然，尽管强调一个整体，小组中应该有一定的角色分配，各种敏捷开发方法角色的起名方案可能不同，但原则基本上是一样的。

2. 敏捷小组按短迭代周期工作

在敏捷项目中，开发小组根据需要定义开发过程，在初始阶段可以有一个简短的分析、建模、设计。项目真正开始后，每次迭代都会进行同样的工作（分析、设计、编码、测试等）。迭代是受时间框限制的，也就是说即使放弃一些功能也必须结束迭代。迭代的时间长度一般是固定的，时间框设定较短，大概是 2～4 周。

3. 敏捷小组每次迭代交付一些成果

开发小组在一次迭代中要把一个以上的不太精确的需求声明，经过分析、设计、编码、测试，变成可交付的软件（称为功能增量）。当然并不需要把每次迭代的结果交付给用户，但目标是可以交付，这就意味着每次迭代都会增加一些小功能，但增加的每个功能都要达到发布质量。每次迭

代结束的时候让产品达到可交付状态十分重要，但这并不意味着要完成发布的全部工作，因为迭代的结果并不是真正发布产品。

4. 敏捷小组关注业务优先级

敏捷开发小组从两个方面显示出它们对业务优先级的关注。首先，它们按照产品所有者制定的顺序交付功能，而产品所有者一般会按照组织在项目上的投资回报最大化的方式来确定优先级，并且把它组织到产品发布中去。要达到这个目的，需要综合考虑开发小组的能力以及所需功能的优先级来建立发布计划。在编写功能的时候，需要使功能的依赖性最小化。功能之间完全没有依赖是不太可能的，但把功能依赖性控制在最低程度还是相当可行的。

5. 敏捷小组检查与调整

每次新迭代开始，敏捷小组都会结合上一次迭代中获得的新知识做出相应调整。如果认为一些因素可能会影响计划的准确性，也可能更改计划。迭代开发是在变与不变中寻求平衡，在迭代开始的时候寻求变，而在迭代开发期间不能改变，以期集中精力完成已经确定的工作。由于一次迭代的时间并不长，所以使得稳定性和易变性能够得到很好的平衡。在两次迭代期间改变优先级甚至功能本身，对于项目投资最大化是有益处的。从这个观点来看，迭代周期的长度选择就比较重要了，因为两次迭代之间是提供变更的机会，周期太长，变更机会就可能失去；周期太短，则会发生频繁变更，而且分析、设计、编码、测试这些工作都不容易做到位。综合考虑，对于一个复杂项目，迭代周期选择 4 周还是有道理的。

1.4.2　开放计算

随着互联网的不断发展和普及，软件工程开放式计算有了技术基础，更多的开放式资源使得软件工程有效地集成，在软件开发标准上形成了互联互通，对于文化、语言来说有所打破，真正地实现了软件开发的协作交流。Linux、Jazz、Android 等软件的开源，对于开放计算来说有了充分的促进，对于软件开发格局有所改变，并且随着互联网的不断普及和发展，对于软件开发计算来说迎来了前所未有的机遇，网络连接了原本分散的开发人员，真正地实现了在基础框架下的集体智慧的升华，能够更高效有序地开发出优秀的产品级软件。

开放计算主要融合了“开放标准”“开放架构”和“开源软件”三个方面。通过坚持“开放标准”，不同企业开发和使用的软件可以互连互通，不同的软件工程工具能够更好地集成，不同国界和不同文化能够更好地协作交流，用户的投资能够得到很好的保证，正是它为全球化趋势奠定了重要基础；“开放架构”通过一组开放的架构标准和技术，有效地解决了商业模式的创新对 IT 灵活性要求的增加和现有 IT 环境的复杂度之间的矛盾，第一次使 IT 和业务走得如此之近，其典型代表包括 SOA、REST 等；而“开源软件”不但书写了 Linux、Eclipse、Jazz 等一个又一个的神奇故事，而且有效地促进了开放标准的发展，同时有效利用社区驱动的开发与协作创新，优化软件设计中的网络效应，开源软件越来越被中小企业和个人用户所认可。

开源软件大量出现，软件外包将更加普及，主要特点如下。

（1）计算能力的增强，集成开发环境更加智能，获取现成的类库更加方便，应用软件开发变得更加容易。

（2）加上软件本身一次性投资的特点，很多的场合甚至可以使用软件替代硬件，使得软件开发需求增加。

（3）消费类电子产品与人们的生活更加息息相关，小的免费软件、小型桌面游戏的出现等，使得需要的软件开发人员数量急剧增长（组织形态是大量的小规模开发团队）。在这一因素以及降

低成本的压力下，开发外包变得非常普及。

（4）项目构建工具，资源依赖更加自动化，系统开发也不需要从零开始，而是利用业内的免费框架进行二次开发。

1.4.3 云计算

云计算（cloud computing）被称为继个人计算机、互联网之后的第三次信息化革命，它是基于互联网的相关服务的增加、使用和交付模式，通常涉及通过互联网来提供动态易扩展且经常是虚拟化的资源。云是网络、互联网的一种比喻说法。云计算是一种理念，是旧瓶子装新酒，它实际上是分布式技术、服务化技术、资源隔离和管理技术（虚拟化）的融合。

到底什么是云计算呢？不同的组织从不同的角度给出了不同的定义。例如：

一种计算模式。把 IT 资源、数据、应用作为服务通过网络提供给用户（如 IBM 公司）。

一种基础架构管理方法论。把大量的高度虚拟化的资源管理起来，组成一个大的资源池，用来统一提供服务（如 IBM 公司）。

以公开的标准和服务为基础，以互联网为中心，提供安全、快速、便捷的数据存储和网络计算服务（如 Google 公司）。

现阶段广为接受的是美国国家标准与技术研究院（NIST）定义，即云计算是一种按使用量付费的模式，这种模式提供可用的、便捷的、按需的网络访问，进入可配置的计算资源共享池（资源包括网络、服务器、存储、应用软件、服务），这些资源能够被快速提供，只需投入很少的管理工作，或与服务供应商进行很少的交互。

通俗意义上的云计算往往包含如图 1-3 所示的内容。开发者利用云 API 开发应用，然后上传到云上托管，并提供给用户使用，而不关心云背后的运行维护和管理以及机器资源分配等问题。

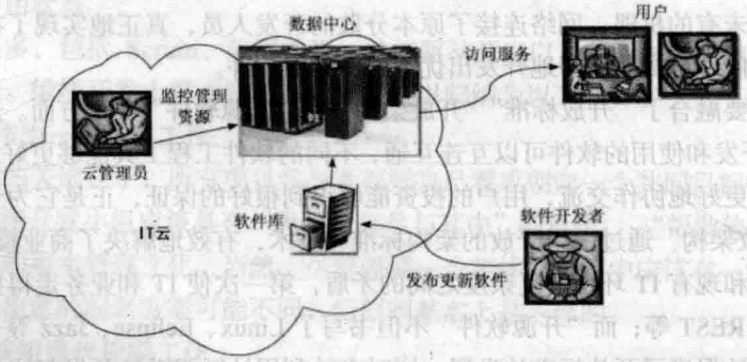

图 1-3　云计算示意图

虚拟化和服务化是云计算的主要表现形式（见图 1-4）。

虚拟化技术包括资源虚拟化、统一分配监测资源、向资源池中添加资源。虚拟化的技术非常多，有的是完全模拟硬件的方式去运行整个操作系统，比如我们熟悉的 VMWare，可以看作重量级虚拟化产品。也有通过软件实现的共享一个操作系统的轻量级虚拟化，比如 Solaris 的 Container、Linux 的 lxc。虚拟化的管理、运行维护多数是通过工具完成的，比如 Linux 的 VirtManager、VMWare 的 vSphere、VMWare 的 vCloud 等。