《软件工程导论》教案

（翻转教学法）

张毅

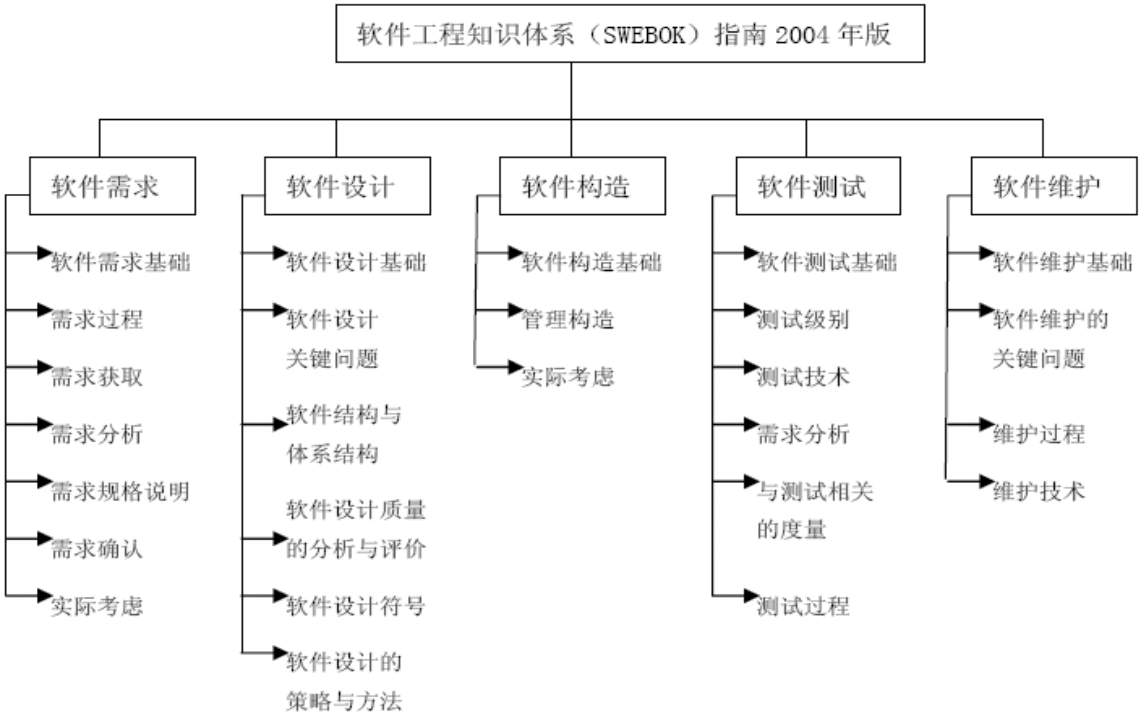
软件工程的目标是提高软件的质量与生产率，最终实现软件的工业化生产。

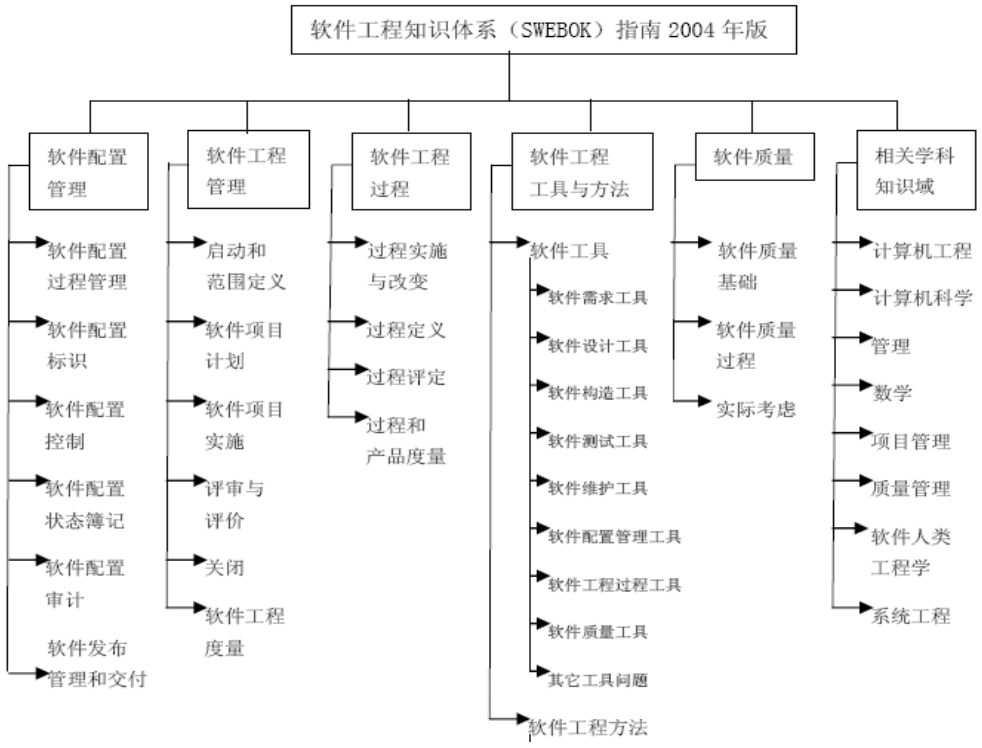
—— 质量是软件需求方最关心的问题，用户即使不图物美价廉，也要求个货真价实。

—— 生产率是软件供应方最关心的问题，老板和员工都想用更少的时间挣更多的钱。

重庆大学　软件学院

软件工程知识体系（SWEBOK）





前　言

软件工程导论是软件工程专业中的一门重要核心课程，图是软件工程专业必修的一门专业课程。本课程从软件生命周期的角度讲解软件工程的基本概念、基本原理和基本方法，主要内容包括软件过程、软件需求、软件设计实现和测试维护等几个部分的相关基础知识，强调软件工程的根本性和永久性原则，关注软件系统的复杂性问题以及迭代式的开发方法，重视分析问题与解决问题的能力以及软件工程实践训练，培养学生良好的工程化开发习惯。

本课程的教学采用“翻转课堂式”教学模式，学生可以通过互联网使用“学堂在线”网上优质教育资源，学生在课外观看视频中教师的讲解，回到课堂上师生面对面交流和完成作业的这样一种教学形态。“翻转课堂式”教学模式不再单纯地依赖授课老师去教授知识，而课堂和老师的角色则发生了变化，老师更多的责任是去理解学生的问题和引导学生去运用知识。

“翻转课堂式”教学模式增加学生和教师之间的互动和个性化的接触时间，让学生对自己学习负责的环境。如果学生课堂缺席，就会被甩在后面，给下一阶段学习带来困难。

通过本课程的学习，使学生能够获得软件工程的基本概念和知识，培养学生从工程的角度出发，认识和理解软件工程规范和标准的意义与作用，理解和掌握软件工程中的典型方法和技术，了解和认识现代软件工具和环境及其在工程中的应用，理解和掌握软件工程管理的基本技术和方法，培养学生能够运用软件工程的原理、方法和技术从事软件产品研发的能力，培养学生能够使用软件的规范和标准描述软件的各种文档的能力。

通过理论教学和实践环节，体现软件工程课程的客观要求和工程教育的基本要求，培养学生团队协作及工程项目研发能力，为今后深入研究软件工程理论和从事软件工程实践奠定良好的基础。

《软件工程导论》翻转教学法

课程安排

序号	知识点	学时	理论课	讨论课	实验课	备注
第 1 部分	第 1 章 初识软件工程	4	2	2		
第 2 部分	第 2 章 软件过程	4	1	3		
第 3 部分	第 7 章 需求获取	6	2	4	2	
第 4 部分	第 12 章 软件体系结构	4	2	2	2	
第 5 部分	第 4 章 单元测试	2	1	1	2	
第 6 部分	第 15 章 软件系统测试	2	1	1	2	
第 7 部分	第 16 章 软件交付与维护	2	1	1		
第 8 部分	第 6 章 敏捷开发与配置管理	2	0	2		可选
第 9 部分	第 14 章 微信抢票应用案例	2	0	2		
合计		28	10	18	8	

目 录

第 1 部分 初识软件工程

1.1 本章的教学目标

通过本章的学习，使学生初识软件工程基本概念，了解计算机发展历程、软件危机、软件工程技术发展及趋势；初步学习计算机学科与软件工程学科范畴、软件工程专业知识体系。

1.2 本章的教学方法

本章的教学方法采用“翻转课堂式”教学模式，学生可以通过互联网使用“学堂在线”网上优质教育资源，选择由清华大学刘强副教授主讲的“软件工程”课程的“第 1 章 初识软件工程” 课外观看学习，网址：
http://www.xuetangx.com/courses/course-v1:TsinghuaX+34100325X+sp/about ，通过视频中教师的讲解，回到课堂上教学交流讨论，并相关的完成习题。

通过理论教学、课堂讨论、适当的布置课外习题作业来完成学习。

1.3 本章的教学内容

1.3.1 软件的本质特性

现在的世界正在进入一个“软件无处不在”的时代，我们每天的生活都离不开软件。在用户来看，软件就是一系列实现各种功能的图形化界面。

什么是软件？软件在计算机内部是如何执行的呢？

首先，计算机内部实际运行着一些程序代码，这些代码负责向计算机发出动作指令，实现不同的功能。

其次，在计算机运行这些程序的过程中，还需要操作诸如学生活动和电子票等数据。

另外，为了后续的维护和开发以及方便用户使用可能还需要一些描述技术实现细节的开发文档和系统使用的用户文档

即：软件是：程序、数据和文档构成的。

- 程序是计算机可以接受的一系列指令，运行的时候可以提供所要求的功能和性能。
- 数据是使程序能够适当地操作信息的一些数据结构。
- 文档则是描述程序的研制过程方法和使用的图文资料。

现在思考一个问题，程序、数据和文档是否代表了软件的真正含义呢？

软件的行为只有通过运行才能呈现出来，也就是说在程序运行的过程中，才能观察到软件的功能和性能，而软件提供的信息和服务又融合了买票这个业务的领域知识，从这个方面来看，软件更像是嵌入式的数字化知识。

如：飞机和软件都是人类的创造物。

飞机是有形的物体，一架大型客机由数百万个单独的部件组成，需要上千人组装，但是通常都能够按时按预算交付使用，而软件是人类思维的创造物。

微软于 1989 年 11 月发布的 Word1.0 版本，这个软件大概有 25 万行的源代码，

但是开发软件却花费了整整 55 个人/年，并一再地推迟交付，比原计划晚了 4 年。直观上来看，建造大型的飞机似乎要比开发一个简单地 Word 软件要复杂得多，但实际情况却大相径庭。

我们不禁要问，开发软件和建造客机到底有什么本质性的区别呢？

FredBrooks 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，由于在计算机体系结构、操作系统和软件工程方面的杰出贡献，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”的文章，在这篇文章中指出，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。

软件是人类思维和智能的一种延伸，它比以往人类的任何创造物都要复杂得多。

今天我们已经进入了云计算时代，在互联网的集群环境下，系统规模更大更复杂，像 Google 的搜索引擎，亚马逊的云计算中心以及阿里云等，其规模都超过了百万台服务器。可以这样说，软件是人类创造的最复杂的物体。这种复杂性会给软件开发管理和质量保证带来很多困难。

软件开发的另一个特点是**人为设计**带来的复杂性，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，但是这些变化都是**人为设计**的结果，可以用“随心所欲”毫无规则来形容。

就拿微信应用来说，用户使用不同的终端，发出服务请求，微信服务器接收请求，并把它们转发给主服务器，主服务器处理响应要求很高的抢票请求，把其他的请求转发给辅助服务器。当然在整个的过程，系统还有可能和一些身份认证等其他系统进行交互。

从这个应用来看，它需要通过与微信系统、身份验证系统等打交道，因此，就必须遵从这些系统定义的接口，而这些接口都是软件工程师**人为设计**的。任何接口的变化，都会带来微信应用的修改。所以，要保持与其他系统的接口一致性，也会造成软件设计的复杂性。

通过微信的演化历史，看一下软件的可变性，微信开发是在 2010 年 11 月正式立项，2011 年 1 月和 3 月分别发布了 1.0 和 1.2 版本，实现文字短信和图片分享功能，但是当时市场反应冷淡，2011 年 5 月推出语音通信功能，发布了 2.0 版本，微信开始快速地流行和传播，2011 年 10 月提供了“查看附近人”的功能，这个功能也成为微信用户增长的一个爆发点，2012 年 4 月微信发布 4.0 版本，先后推出了“朋友圈”，视频通话以及微信公众平台，开始建立手机上的熟人社交圈，2013 年 8 月以后，微信 5.0 及其后续版本，提供了如支付、表情、游戏等一系列贴近用户生活的服务，并且专门开辟了服务号，打造了一种更具体验效果的，移动互联网的生活方式。

这些事例告诉我们，软件只要是使用，就会一直在变，而这个改变是随需而变，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，这也许是因为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，理想的情况下随着软件的使用，其故障率会逐渐降低，逐渐达到一个稳定地质量，但是软件在不断变化的，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，带来破坏不断的修改，最终可能导致软件发生退化，从而结束其生命周期。

尽管如此，成功的软件都是会发生演化的，**没有任何变化的软件一定是没有用的。**虽然软件的可变性，给开发带来了很多难题，但也给软件本身带来了生命力。所以，我们要用积极地态度和有效地方法来控制软件变更，使软件在演化过程中保证高

质量。

软件还有一个特性，就是“看不见”“摸不着”。像建筑 飞机这些有形的物体，我们可以用平面的图形，抽象出其结构形态，但是软件是逻辑的产品，没有空间的形体特征，因此，缺少合适的几何表达方式。开发人员在接收到用户需求的时候，整个开发的过程只能看到源代码，而代码并不是软件本身，因此，在开发完成之前，很难看到软件是如何执行的，这种不可见性，不仅限制了软件的设计过程，同时严重地阻碍了相互之间的人与人的交流，从而对开发过程的管理造成很大困难。

Brooks 教授在著名的《人月神话》一书中，把软件人员形容成“皇帝的新衣”故事里的裁缝，开发人员一直在编织，却看不到编织的织物，最后，也许是编织出来一堆无用的废物，或者什么也没有做出来。

通过前面的分析我们可以看到，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，需要寻找解决问题的有效方法，从而保证软件开发过程的高效、有序和可控。

1.3.2 软件工程的产生与发展

1.3.2.1 软件工程的产生

软件具有复杂性、一致性、可变性和不可见性。这些特性使软件开发及其管理变得很难控制，最终的产品质量也难以保证。

美国 Standish 集团是一个专门跟踪调查软件项目的研究机构，如图 1 所示为跟踪调查软件项目完成情况。

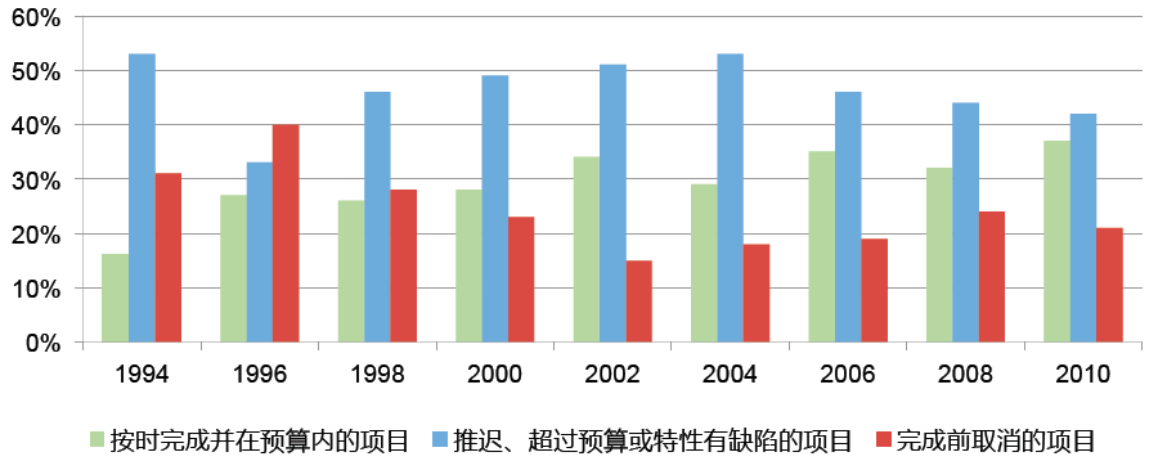


图 1

这幅图显示了该机构在 1994－2010 年期间对软件开发项目的调查统计结果，研究数据表明，软件项目的平均成功率大概在 30%左右，这里成功的含义是指在计划的时间和预算内，实现项目的目标，大概有一半左右的项目超出预算和最后期限，或者存在这样那样的缺陷，另外还有 20%左右的项目彻底失败了，即使投入了极大努力，最终完成开发的软件也是存在着错误多、性能低、不可靠、不安全等质量问题，有的甚至造成了严重的后果。

下面我们看几个软件失败的典型事例。

实例 1：欧洲航天局 Ariane 5 火箭控制软件。

1996 年 6 月 4 日欧洲航天局 Ariane 5 火箭，在发射 37 秒之后偏离了它的飞行

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，这段程序是开发人员复用了 Ariane 4 火箭的一段程序。

那 Ariane 4 火箭为什么没有出现问题呢？

主要原因在于，Ariane 5 火箭比 Ariane 4 的速度高出近 5 倍，这样在计算时超出了一个 16 位数的范围，发生了溢出。显然开发人员在设计 Ariane 5 火箭的时候，只是简单地重用了这部分程序，并没有检查它所基于的假设。

实例 2：微软 Windows Vista 系统。

Windows Vista 系统是曾经被微软寄予厚望的一个桌面操作系统，也是微软历史上最艰难、最曲折、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，很多的时间都用在了互相沟通和重新决策上，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、频繁死机等严重缺陷。可以说这是一款失败的软件产品。

实例 3：12306 网上订票系统

大家可能都在 12306 网站上买过火车票，这个系统历时两年研发成功，耗资 3 亿元人民币，于 2011 年 6 月投入使用，在 2012 年春运期间，一天网站点击量超过 14 亿次，系统出现了网站崩溃，登录缓慢、无法支付、扣钱不出票等严重的问题。

当年在中秋和“十一”黄金周，网站的日点击量又创新高，发售客票超过当年的春运，系统继续出现网络拥堵，重复排队等现象，另外在 2014 年春运期间，由于网站对身份证信息缺乏审核，出现了用虚假身份证，可以直接购票的现象，同时网站还曝出大规模串号，购票日期穿越等漏洞，显然，软件开发一直面临着诸多的挑战。

主要表现在以下方面：

- 软件产品的交付质量难以保证
- 许多功能不是用户需要的
- 用户使用的时候出现很多 Bug
- 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
- 开发团队专注于技术，忽视对风险的管理，从而造成了整个开发成本的超支

另外，如何提升团队的能力和效率，一直是一个难题。

软件工程就是致力于探索软件开发问题的解决之道。

1968 年北大西洋公约组织在德国的一个小镇召开国际会议，首次提出了“软件工程”这个术语，这也标志着一个新学科的开始，当时的会议报告中这样写道：“我们特意选择‘软件工程’这个颇具争议性的词，就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”。

1.3.2.2 软件工程的发展

软件工程的发展大概经历了四个阶段，如图 2 所示。

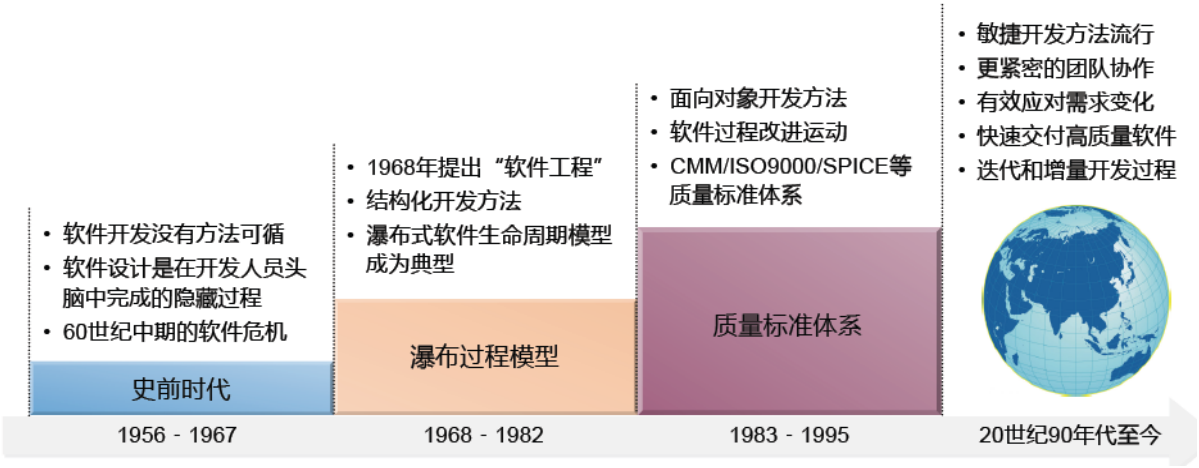


图 2 软件工程的发展四个阶段

第一阶段：1956 年到 1967 年

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，20 世纪 60 年代末，爆发了软件危机。

第二阶段：1968 年到 1982 年

从 1968 年开始，软件工程开始了一个新的时期，一直到 20 世纪 80 年代末，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，这些阶段也是按照线性的方式执行的.

第三阶段：1983 年到 1995 年

从 1983-1995 年，人们开始意识到过程质量，对产品质量的重大影响，这个时期面向对象的方法和过程改进运动逐渐盛行，提出了 CMM/ISO9000/SPICE 等质量标准体系。

第四阶段：1996 年到至今

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，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。

1.3.3 软件工程的基本概念

1.3.3.1 软件工程的理解

随着人类文明的不断发展，人们开始建造出各种各样更大更复杂的“人造系统”，像高楼大厦、轮船、铁路、飞机等，所谓的“工程”就是应用有关的科学知识和技术手段，通过有组织的群体协作活动，建造具有预期使用价值的人造产品的过程。不管是建造高楼大厦、跨海大桥，还是生产轮船和飞机，工程活动一般都具有以下共同的特征：即大规模的设计和建造涉及到复杂的问题，目标分解需要团队的协作和过程的控制。

因此，软件工程就是把工程化的方法应用到开发软件中，也就是将系统性的、

规范化的、可定量的方法应用于软件的开发、运行和维护，并且，对这种方法进行研究。

软件工程最终实现的目标：是创造出“足够好的软件”。

软件开发的成本要低，并能够按时交付，产品实现客户要求的功能，具有良好的可靠性和可扩展性等，软件维护的费用低。

软件开发是一个复杂的过程，需要注重过程质量的控制，而先进的方法和工具可以有助开发高质量的产品。

因此，过程、方法和工具是软件工程的三个基本要素。

软件过程是一系列开发活动，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。

软件开发过程一般包括：一系列基本的开发活动，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，还应包括一些开发过程管理等支持性的活动，在软件开发过程中，先进的开发方法和技术手段是非常重要的，可以提高开发效率，有助于构建出高质量的产品。

软件开发方法的发展经历了面向过程、面向对象、面向构件和面向服务等不同阶段。

古人云“工欲善其事，必先利其器”，工匠想要做好自己的工作，一定要先让工具锋利。

软件开发也不例外，软件开发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，现在开源的工具非常多，贯穿于整个开发过程，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。

现在，有不同编程语言的构造工具，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，可以帮助理解和维护代码，还有一些软件工程管理工具，帮助有效管理开发过程，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，但是仅有好的战术还是不够，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。

构造一个新的系统，不必都从零开始，直接复用已有的构件，可以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。

分而治之是人们处理复杂性的一个基本策略，简单地说，就是将一个复杂的问题，分解成若干个简单的问题，然后逐个进行解决。

例如：微信系统，在前端、后端、数据存储等不同层次都划分成若干功能模块，这些模块，可以分配给不同的开发人员，进行并行开发，最后再组装成一个完整的系统。

微信软件的一个演化过程，从一个简单的文字通讯工具，逐渐演变成一个支持人们现代生活的服务平台，应该说，软件也像是一个活着的植物，它的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，通过不断的迭代式的增量开发，最终，发展成符合客户价值的产品。

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得达到最优的时候，需要进行折中来实现整体的最优。

1.3.3.2 软件工程的 Wasserman 规范

Wasserman 提出了软件工程中存在的 8 个基本概念，这些概念构成了有效的软件工程规范的基础。

1. 抽象

有时，在一个问题的"自然状态"（即如同客户和用户表达的那样）考虑这个问题是一件令人畏惧的事情。在问题的"自然状态"下，我们不可能发现以有效的或者甚至只是可行的方法处理问题的显而易见的方式。

抽象（abstraction）是在某种概括层次上对问题的描述，使得我们能够集中于问题的关键方面而不会陷入细节。

2. 分析和设计方法以及表示法

当设计一个作为课程作业的程序时，通常需要自己完成工作。产生的文档是一个正式描述，它告诉你自己为什么选择这个特定的方法、变量名的含义是什么以及实现的算法。但是，当与团队一起工作的时候，必须与开发过程中的其他参与者进行交流。大多数工程师，无论他们是做什么样的工程，都会使用标准的表示法来帮助他们进行交流以及文档化相关决策。例如，建筑师画了一张图或蓝图，任何其他的工程师都能够理解他画的图。更为重要的是，公共的表示法使得建筑承包商能够理解建筑师的意图和想法。

软件工程中没有类似的标准，由此产生的误解是当今软件工程中的一个关键问题。分析和设计方法不止是提供了交流媒介，还使我们能够建立模型，并检查模型的完整性和一致性。再者，我们可以更容易地从以前的项目中复用需求和设计组件，从而相对容易地提高生产率和质量。

但是，在我们能够决定一组标准的方法和工具之前，仍然有许多悬而未决的问题需要解决。不同的工具和技术处理的是问题的不同方面，我们需要标识建模原语，以便用一种技术就能获取问题的所有重要的方面。或者我们需要开发一种供所有方法使用的表示技术，当然可能需要某种形式的剪裁。

3. 用户界面原型化

原型化（prototyping）意味着构建一个系统的小版本，通常只有有限的功能，它可用于：帮助用户或客户标识系统的关键需求，证明设计或方法的可行性。

通常，原型化过程是迭代的：首先构建原型，然后对原型进行评估（利用用户和客户的反馈），考虑如何改进产品或设计，之后再构建另外一个原型。当我们和客户认为手头问题的解决方案令人满意时，迭代过程就终止了。

原型化通常用来设计一个良好的用户界面（user interface），即系统与用户交互的部分。但是，在其他场合也可以使用原型，甚至是在嵌入式系统（embedded system）（即其中的软件功能不是明确地对用户可见的系统）中。原型能够向用户展示系统将会有什么样的功能，而不管它们是用硬件还是用软件实现的。因为从某种意义上讲，用户界面是应用领域和软件开发团队之间的桥梁，所以，原型化可以把使用其他需求分析方法不能明确的问题和假设表面化。

4. 软件体系结构

系统的整个体系结构不仅对实现和测试的方便性很重要，而且对维护和修改系统的速度和有效性也是很重要的。体系结构的质量可能成就一个系统，也可能损害一

个系统。

系统的体系结构根据一组体系结构单元以及单元之间的相互关系来描述系统。单元越独立，体系结构越模块化，就越容易分别设计和开发不同的部分。

Wasserman 指出，至少有 5 种方法可以将系统划分为单元。

(1) 模块化分解：基于指派到模块的功能。

(2) 面向数据的分解：基于外部数据结构。

(3) 面向事件的分解：基于系统必须处理的事件。

(4) 由外到内的设计：基于系统的用户输入。

(5) 面向对象的设计：基于标识的对象的类以及它们之间的相互关系。

这些方法并不是相互排斥的。例如，可以用面向事件的分解设计用户界面，同时，使用面向对象或面向数据的方法来设计数据库。这些方法之所以重要，是因为它们体现了我们的设计经验，并通过复用已经做过的和所学到的，充分利用过去的项目。

5.软件过程

自从 20 世纪 80 年代后期以来，很多软件工程师已经在密切留意开发软件的过程以及由此产生的产品。活动中的组织和规范对软件的质量和软件开发速度的积极作用已经得到承认。然而，Wasserman 指出：

不同应用类型和组织文化之间的巨大差异使得对过程本身进行预先规定是不可能的。因此，软件过程不可能以抽象和模块化的方式作为软件工程的基础。(Wasserman 1996)

相反，他提出，不同的软件类型需要不同的过程。尤其是，Wasserman 指出企业范围的应用程序需要大量的控制，而单个的或部门级的应用程序可以利用快速应用程序开发，如图 3 所示。

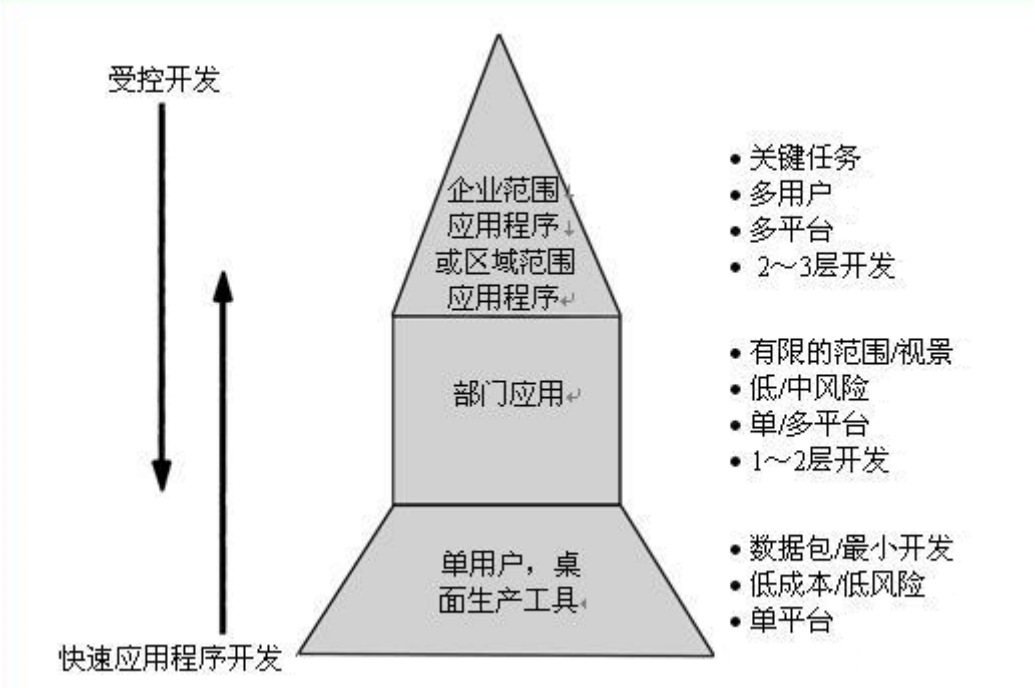


图 3 不同开发中的差别

利用目前的工具，很多中小规模的系统可以由一两个开发人员来完成，其中每个开发人员必须担任多个角色。这样的工具可能包含文本编辑器、编程环境、测试支持工具，还可能包含一个获取关于产品和过程的关键数据元素的小型数据库。因为项

目的风险相对较低，所以需要很少的管理支持或评审。

但是，大型、复杂的系统需要更多的结构、检查和平衡。这些系统通常涉及很多客户和用户，并且开发会持续很长时间。再者，因为某些关键子系统可能由他人提供或用硬件实现，开发人员并不总是能够控制整个过程。这种类型的高风险系统需要分析和设计工具、项目管理、配置管理、更复杂的测试工具以及对系统更严格的评审和因果分析。

6.复用

在软件开发和维护中，通过复用以前开发项目中的项来利用应用程序之间的共性。例如，在不同的开发项目中，我们使用同样的操作系统和数据库管理系统，而不是每次都构建一个新的。类似地，当我们构建一个与以前做过的项目类似但有所不同的系统时，可以复用需求集、部分设计以及测试脚本或数据。

有时候，构建一个小的构件比在可复用构件库中搜索这样一个构件要更快。

要使一个构件足够通用、可以在将来被其他开发人员很容易地复用，则可能需要花费格外多的时间。

由于难以对做过的质量保证和测试的程度进行文档化，可能会导致一个潜在的复用人员认为构件的质量是令人满意的。

如果某个复用的构件失效或需要进行更新，不清楚谁应该对此负责。

理解和复用一个由他人编写的构件，其代价可能是高昂的，也可能是很耗时的。在通用性和专业性之间通常存在冲突。

7.测度

改进是软件工程研究的驱动力：通过改进过程、资源和方法，我们可以生产和维护更好的产品。但是，我们有时只能概况地表示改进目标，原因是没有量化地描述我们做了什么以及我们的目标是什么。正因为如此，软件测度已经成为好的软件工程实践的一个关键方面。通过量化我们做了什么以及我们的目标是什么，就可以用通用数学语言来描述我们的行动和结果，从而使我们能够评估我们的进展。另外，量化的方法允许我们比较不同项目的进展。

8.工具和集成环境

多年以来，厂商一直推荐使用 CASE（计算机辅助软件工程）工具，其中的标准化的集成开发环境将增强软件开发。但是，我们已经看到，不同的开发人员是如何使用不同的过程、方法和资源的。因此，一个统一的方法说起来容易，做起来就难了。

另一方面，研究人员已经提出了几个框架，使我们能对已有的环境和打算构建的环境进行比较和对照。这些框架还允许我们检验每个软件工程环境提供的服务，决定哪一个环境最适合于给定的问题或应用程序的开发。

对工具进行比较主要的难点之一是厂商很少针对整个开发生命周期。相反，他们集中于小的活动集，例如设计或测试等，并且由用户把选择的工具集成到一个完整的开发环境中。Wasserman 指出了在任何工具集成中必须处理的下列 5 个问题（Wasserman 1990）。

(1) 平台集成：工具在异构型网络中的互操作能力。

(2) 表示集成：用户界面的共性。

(3) 过程集成：工具和开发过程之间的链接。

(4) 数据集成：工具共享数据的方式。

(5) 控制集成：一个工具通知和启动另一个工具中的动作的能力。

1.3.4 软件质量实现

软件已经成为人们生活中不可缺少的一部分,正是由于软件越来越重要,人们对其质量的要求也就越来越高。人们都希望开发高质量的软件，但是由于受到市场因素的牵制，不可能达到“完美”，因此，软件工程的目标，不是实现“完美”，而是达到“足够好”，那么，什么是好的软件呢？质量是一个复杂多面的概念，不同的人从各自的视角，会有不同的理解和要求。对于用户来说，他们更关心的是系统的功能质量，比如说是否满足了自己的需求，是否存在影响使用的缺陷，软件性能如何以及是否容易使用等。开发人员更多关心的是系统的结构质量，主要包括代码的可读性、可测试性、可维护性以及效率和安全等方面的因素。对于投资者来说，软件开发的过程质量更为重要，他们主要关心的是项目是否可以在规定的时间和预算内交付，最终产品的交付质量是否可以保证。

通常来说，软件质量应该涵盖软件过程、软件产品和产品效用三个方面。

软件过程用过程质量来衡量，软件产品则包括内在质量和外在质量两部分，产品的效用用使用质量来衡量。

过程的质量会影响到软件产品内在的代码质量，而代码质量的好坏决定了产品的外在质量，外在质量最终影响到用户的使用质量，因此，我们必须用有效的方法来检验整个开发过程、程序代码和最终产品，并对用户的使用质量进行监测，质量大师温伯格认为，质量就是软件产品对于某个（或某些）人的价值，这里的某个或某些人通常指的就是用户。在这一句话中包含了两个层次的质量含义，即“正确的软件”和“软件运行正确”。

所谓“正确的软件”是说一个软件要能够满足用户的需求，为用户创造价值。比如说带来工作的便利，创造了利润或者减少成本。而“软件运行正确”说的是软件没有或很少有 Bug，扩展性强、性能良好、易用性高等。

我们开发的软件既要是正确的也要是运行正确的，这是曾经引发全球热潮的。谷歌眼镜（Google glass）从 2015 年 1 月 19 日开始不再接受订单，与此同时，谷歌还关闭了“探索者”这个软件开发项目，除去销售策略欠佳方面的原因，一个关键的问题在于谷歌自己也不清楚，谷歌眼镜存在的目的，它是让用户更便捷地获得电子邮件等通知信息，还是更迅速地拍照，由于需求调研不足，产品过于超前，再加上价格昂贵等因素，使得这款看起来很酷的产品，很快就丧失了对用户的吸引力。

前面我们提到过微软的 Vista 系统，这是一个典型的“运行不正确”的软件，Vista 系统应该说是用户非常期待的，也是满足用户需求的，但是由于莫名其妙的死机，以及很差的运行效率等一系列缺陷，最终导致许多用户弃用，甚至有的用户选择退回到 XP 版本，所以微软不得不在两年之后用 Win 7 取代了 Vista。

上面两个事例很好地说明了一个好的软件既要是“正确的”也要是“运行正确的”，这两者是相辅相成的，前者关系到软件的成败，后者关系到软件的好坏。对于开发团队来说尤其是偏重技术的开发团队,往往过分注重后者,也就是说“运行正确”,经常会陷入到在软件开发过程中的技术细节中，而忽略了前者，也就是软件需要符合用户的需求，这样开发出的软件经常是能用，但不是用户满意的产品。从软件产品的角度来看,高质量意味着做了用户想要它做的事情,能够正确有效地使用计算机资源,易于用户学习和使用，系统设计良好，代码编写规范，而且易于测试和维护。

在理解了质量的基本含义之后，我们要考虑一下应该如何评价质量。

产品的质量可以从多个维度来进行评价的,David Garvin 提出了一种多维度的质量评价模型，具体包括性能质量、产品特色、可靠性、符合性、耐久性、可服务性以及外观性和感知性等 8 个维度。不同的属性反映了产品质量的不同方面，我们可以通过改善产品的各个质量属性，来提高整个产品的质量。

我们用一个汽车的例子来说明一下 Garvin 的 8 个质量维度，汽车首先是要满足人们出行的驾驶需要，所以在功能和性能上符合要求，为最终用户提供价值，除了主要的产品特性之外，如果还有一些诸如自动泊车、未关门报警、在遥控器锁车的时候自动检测，并关闭敞开的车窗等特性，这些特性会给用户带来意外的惊喜。当然，汽车要可靠安全，要符合国家或行业的相关标准，日常维修不能损坏和降低车辆的性能，维修时间是在可接受的范围之内，除了这些主要的产品特性之外，人们对产品的审美也是必不可少的，当然这个审美是各有不同是非常主观的，也许有的人欣赏某种优雅、稳重的外形，有人则喜欢醒目动感的外在，但是对于产品来说，审美这个性质是非常重要的。

有时候一些偏见，也会影响人们对质量的感知，例如企业或者品牌的声誉会让人对产品质量产生相应的感觉，Gravin 的质量维度是一个通用的产品评价体系，并不是专门为软件制定的，但是它也同样适用于软件产品。

ISO9126 模型是一种评价软件质量的通用模型，它定义了软件的 6 个质量属性，即功能性、可靠性、易用性、效率或性能、以及可维护性和可移植性，每一个属性又细分成一系列子属性。

下面我们简单地解释一下每个质量属性的含义。

功能性：包括适合性、准确性、互操作性和安全性四个子属性。适合性就是说软件提供了用户所需要的功能，同时软件提供的功能也是用户所需要的。准确性是指软件提供用户功能的精确度达到要求，比如说运算结果的准确，数字没有偏差等等。互操作性是软件和其它系统进行交互的能力，当然软件还要有保护信息和数据的安全能力，比如说，我们在京东网站上购买商品，像关键字搜索、商品质量浏览、选购商品、生成产品订单等，都是用户需要的主要功能，订单费用的结算，也必须是运行准确的，同时这个系统还会和银行卡支付系统进行交互，系统也要保证用户账号的安全。

可靠性：指的是系统是否能够在一个稳定的状态下满足用户的使用，那么这就意味着软件要有代码出错的处理能力，对外部出现错误时，软件仍然能够保持正常的运行状态，当然软件不可能绝对不出现问题，但是在失效发生的情况下系统应该能够重新恢复到正常的运行，同时恢复受直接影响的数据。

易用性：是指在规定的条件下使用时，软件产品被理解、学习、使用和吸引用户的能力，软件的易用性是用户在使用过程中，所实际感受到的系统质量，它会直接影响到用户对产品的满意度，性能也是影响产品质量的一个重要因素，通常从时间特性和资源使用两个方面来衡量系统的性能水平。比如说我们打开一个商品网页，浏览商品的时候，希望系统对用户请求具有快速的响应和处理，同时消耗的系统资源和网络带宽比较低。

对于开发人员来说，可维护性和可移植性是非常重要的，可维护性用于衡量软件产品被修改的时候，需要花费多大的努力。可移植性是指软件从一种环境迁移到另一种环境的难易程度。

现在大家考虑一个问题，软件质量是如何实现的，或者说如何才能有效地提高软件质量，软件开发过程包括了分析、设计、实现、测试等一系列活动，测试是检验软件产品的一个重要的手段，但是质量是能够测出来的吗？

实际上到测试的阶段，软件质量的问题如果发生已经很难进行纠正，所以说质量并不是测出来的，而是在开发过程中逐渐的构建起来的。构建高质量代码是每一个软件工程师义不容辞的责任，当然，测试也是开发过程中不可缺少的一个重要环节，因此，对于软件开发来说，高质量的设计、规范的编码以及有效的测试是保证。软件产品质量的三个重要方面也是提高软件质量的必要手段，现在问题又来了，软件质量

的重要性是无容置疑的，那么是不是质量越高就越好，或者是说软件产品是不是应该追求“零缺陷”。

下面我们来看两种不同的使用环境，对软件质量的不同要求。

（1）对于航天软件来说，如果在发射时软件出现问题，就会造成巨大的损失，所以说在发射之前，只要发现任何异常就会立即取消发射指令，直到异常被消除为止，在这种情况下，软件的质量越高越好，软件产品追求的是“零缺陷”。

（2）像许多互联网软件，例如新浪微博、百度导航等，在产品还存在一定缺陷的情况下就发布上线，之后再不断地更新版本，修复已有的缺陷，似乎在这种情况下，用户也是可以接受，一个有缺陷的软件产品的。那么，为什么这种系统不像航天系统一样，需要在发布之前修复所发现的任何缺陷呢。显然，我们不能抛开商业目标来谈论产品质量。企业的根本目标是要获得尽可能多的利润。为了提高用户对产品的满意度，企业必须提高产品质量，但是也不可能为了追求完美的质量，而不惜一切代价，质量是有成本的。当企业为提高质量所付出的代价超过了产品收益时，这个产品也就没有商业价值了。因此，企业必须权衡质量、效率和成本三个因素。产品质量太低或太高都不利于企业的长远发展，理想的质量目标不是“零缺陷”，而是恰好让用户满意，并且将提高质量所付出的代价控制在预算之内。

1.3.5 业界人士谈软件工程

软件工程的本质特性：

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；

——许多软件的复杂性主要不是由问题的内在复杂性造成的，而是由必须处理的大量细节造成的。

3、软件经常化；

4、开发软件的效率非常重要；

5、和谐地合作是开发软件的关键；

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替具有另一种文化背景的人创造产品。

软件工程的基本原理：

1、用分阶段的生命周期计划严格管理；

2、坚持进行阶段评审；

3、实行严格的产品控制；

4、采用现代程序设计的技术；

5、结果应能清楚地审查；

6、开发小组的人员应该少而精；

——人数为N时，可能的通信路径有N(N-1)/2条。

7、承认不断改进软件工程实践的必要性。

1.3.6 软件危机

软件危机是指不断增加的为大系统制造可靠软件的困难度。软件系统的规模和复杂性不断增长，对软件的需求增长超过了供应（开发、演化和维护）能力。软件开发人员极度短缺、开发效率和软件质量不能满足用户的需求，最终延缓经济和社会的发展。

其主要表现如下：

（1）开发的软件产品不能满足用户的需求，即产品的功能和特性与需求不符。

（2）与硬件相比，软件代价过高。

（3）软件的质量难以保证。

（4）软件开发、维护的成本和费用难以准确估计。

（5）开发进度超期。

（6）风险难以控制。

（7）软件维护困难。

（8）文档不完整，与软件产品不一致。

1.4 讨论题

1、什么是软件？什么是软件工程？你认为软件工程专业主要是做什么的？

学习软件工程专业是否就是学习“编程序”“做码农”呢？

就你所了解的，你认为软件工程专业与计算机专业的区别是什么？

请分享一下你对软件工程、软件工程专业的认识。

2、你是如何理解软件危机的？

3、请阅读下面的事件描述，并回答问题：

（1）导致事故发生的原因是什么？

（2）在软件开发过程中应该强调什么事项，以便更好地防止类似的问题发生？

实例：泛美航空公司飞机失事的事件描述：

达拉斯 8 月 23 日电——航空公司今天声称，去年 12 月在哥伦比亚失事的泛美航空公司喷气式飞机的机长输入了一条错误的单字母计算机指令，正是这条指令使飞机撞倒了山上。这次失事致使机上 163 人中除 4 人生还外，其余全部丧生。

美国调查人员总结说，显然这架波音 757 飞机的机长认为他已经输入了目的地 Cali 的坐标。但是，在大多数南美洲的航空图上，Cali 单字母编码与波哥大（Bogota）的编码相同，而波哥大位于相反方向的 132 英里处。

据泛美航空公司的首席飞行员和飞行副总裁 Cecil Ewell 的一封信中说，波哥大的坐标引导飞机撞到了山上。Ewell 说，在大多数计算机数据库中，波哥大和 Cali 的编码是不同的。

泛美航空公司的发言人 John Hotard 确认，Ewell 的信首先是在《达拉斯早间新闻》中报道，本周交到了所有航空飞行员的手中以警告他们这种编码的问题。泛美航空公司的发现也促使联邦航空局向所有的航空公司发布公告，警告他们有些计算机的数据库与航空图存在不一致。

计算机错误还不是引起这次失事原因的最终结论，哥伦比亚调查人员也在检查飞行员训练和航空交通管制的因素。

Ewell 谈到，当他们把喷气式飞机的导航计算机与失事计算机的信息相比较时，泛美航空公司的调查人员发现了计算机错误。数据表明，错误持续了 66 秒钟未被检测到，而同时机组人员匆忙遵守交通管制的指令采取更直接的途径到达 Cali 机场。3 分钟后，当飞机仍在下降而机组人员设法解决飞机为什么已经转向时，飞经坠毁了。

Ewell 说这次失事告诉了飞行员两个重要的教训：“首先，不管你去过南美或任何其他地方多少次，比如落基山区，你绝对不能假设任何情况。其次，飞行员必须明白他们不能让自动驾驶设备承担飞行的责任。”

1.5 测试题

单选题 (15 满分)

1.下面的（ ）说法是正确的。
A.由于软件是产品，因此可以应用其他工程制品所用的技术进行生产
B.购买大多数计算机系统所需的硬件比软件更昂贵
C.大多数软件系统是不容易修改的，除非它们在设计时考虑了变化
D.一般来说，软件只有在其行为与开发者的目标一致的情况下才能成功

2.造成大型软件开发困难的根本原因在于（ ）。
A.开发人员缺乏足够的开发经验
B.对软件开发的资金投入不足
C.项目开发进度不合理
D.软件系统的复杂性

3.软件会逐渐退化而不会磨损，其原因在于（ ）。
A.软件通常暴露在恶劣的环境下
B.软件错误在经常使用之后会逐渐增加
C.不断的变更使组件接口之间引起错误
D.软件备件很难订购

4.“软件工程”术语是在（ ）被首次提出。
A.FredBrooks 的《没有银弹：软件工程中的根本和次要问题》
B.1968 年 NATO 会议
C.IEEE 的软件工程知识体系指南（SWEBOK）
D.美国卡内基·梅隆大学的软件工程研究所

5.Ariane 5 火箭发射失败的事例告诉我们（ ）。
A.系统环境的变化可能影响软件采集数据的精度、范围和对系统的控制
B.软件后备系统可以通过复制生成
C.软件重用必须重新进行系统论证和系统测试
D.选项 A 和 C E.选项 A、B 和 C

6.下面的（ ）说法是正确的。
A.软件危机在 20 世纪 70 年代末期全面爆发
B.软件开发面临的主要挑战是在软件产品中存在一系列的质量问题
C.敏捷开发方法已经成为当今软件开发领域的主流
D.当前先进的软件工程方法已经解决了软件开发遇到的问题

7.（ ）是将系统的、规范的、可定量的方法应用于软件的开发、运行和维护。

A.软件过程
B.软件工程
C.软件产品
D.软件测试

8.软件工程的基本目标是（ ）。

A.开发足够好的软件
B.消除软件固有的复杂性
C.努力发挥开发人员的创造性潜能
D.更好地维护正在使用的软件产品

9.软件工程的基本要素包括方法、工具和（　）。
A.软件系统
B.硬件环境
C.过程
D.人员

10.软件工程方法是（　）。
A.为了获得高质量软件而实施的一系列活动
B.为开发软件提供技术上的解决方法
C.为支持软件开发、维护、管理而研制的计算机程序系统
D.为了理解问题和确定需求而采取的一些技术和方法

11.（　）是利用已有的软件制品，直接组装或者合理修改形成新的软件系统。
A.软件复用
B.分而治之
C.逐步演进
D.优化折中

12.下面的（　）是正确的。
A.运行正确的软件就是高质量的软件
B.软件质量是在开发过程中逐渐构建起来的
C.软件产品质量越高越好，最理想的情况是达到“零缺陷”
D.软件质量是由产品的功能、性能、易用性等外在特性决定的。

13.在 Garvin 多维度模型中，可靠性是指（　）。
A.软件产品提供了让用户产生惊喜的特性
B.软件实现了用户需要的功能和性能
C.软件在规定时间和条件下无故障持续运行
D.软件符合国家或行业的相关标准

14.在 ISO 9126 层次模型中，下面的（　）子属性属于功能质量属性。
A.准确性
B.成熟性
C.易操作性
D.稳定性

15.（　）是软件从一个硬件或软件环境转换到另一环境的容易程度。
A.易用性
B.可维护性
C.可移植性
D.性能

第 2 部分 软件过程

2.1 本章的教学目标

通过本章的学习，使学生初识软件过程和软件的生命周期，掌握软件的生命周期模型及敏捷开发过程等。

2.2 本章的教学方法

本章的教学方法采用“翻转课堂式”教学模式，学生可以通过互联网使用“学堂在线”网上优质教育资源，选择由清华大学刘强副教授主讲的“软件工程”课程的“第 2 章 软件过程”课外观看学习，网址：
http://www.xuetangx.com/courses/course-v1:TsinghuaX+34100325X+sp/about ，通过视频中教师的讲解，回到课堂上教学交流讨论，并相关的完成习题。

通过理论教学、课堂讨论、适当的布置课外习题作业来完成学习。

2.3 本章的教学内容

本章主要介绍什么是软件过程、软件的生命周期、软件的生命周期模型及敏捷开发过程等。

2.3.1 软件过程

过程是软件工程的一个基本要素，过程是一个广义的概念，它是通过一系列相互关联的活动，将输入转化为输出。对于软件开发来说，用户需求是软件开发的基础，也是整个开发过程的一个输入。开发人员通过一系列软件开发活动，最终交付出用户需要的产品。

如何才能保证最终交付的是用户满意的产品，我们可以对最终产品进行质量检验，这个工作肯定是必要的，但是事后的检验，只能验证输出产品的质量，而无法保证产品一定就是符合用户要求的，这一点需要在软件的构建过程中，加以控制和管理。

产品质量保证的关键在于对软件开发过程，进行有效控制，要实现对软件开发过程的有效控制，首先要规定过程实现的方法和步骤，也就是说要把整个过程进行细分，详细地定义出过程里面的每一个环节，以及各个环节之间的执行顺序，其次，还要对过程进行监控，但是这个监控，并不只是对最终产品进行质量检验，而是要对过程的开始，每一个活动的执行，一直到结束，进行全方位地监测，从而保证每一个活动能够，达到应有的质量。过程管理对产品质量的控制，起到至关重要的作用。

例如：麦当劳食品的制作过程，其过程管理是非常重要的。

无论在世界的什么地方，进入麦当劳餐厅，除了少量本地特色食品之外，几乎所有餐品的质量，都没有太大的差别。其对汉堡包的原材料，和制作过程，都做了非常细致的要求，诸如面包的大小，牛肉的品质，肉饼的组成，以及汉堡的烘烤时间等。也就是无论何人，无论何时，无论何地无品质差异，这就是运用过程方法来有效保证最终的产品质量。

过程方法是系统地识别和管理，组织内所使用的过程，保证更有效地获得期望的结果。

过程方法是一个组织和管理，工作活动的有效手段，其目的是更好地为用户创造价值，通常过程包括：实现过程、管理过程和支持过程三种类型。其中实现过程，是提供有价值产品或服务的关键性活动，这些活动负责把输入转化为输出，支持活动负责提供所需要的资源和能力，从而使关键性的实现活动，能够顺利进行。管理活动则是衡量和评价实现过程，和支持过程的效能，建立起组织的质量管理体系。

软件过程是为了获得高质量软件，而实施的一系列活动，它包括：问题定义、需求开发、软件设计、软件构造、软件测试等一系列软件开发的实现活动，而每一项活动，都会产生相应的中间制品，为了保证软件开发过程，能够按照预定的成本，进度质量顺利完成，还需要如项目管理、配置管理、质量保证等一系列开发管理活动，通过建立整个组织的质量管理体系，实现对软件开发，实现活动的有效控制和质量保证，如图 2.1 所示。

任何一个软件产品，都起源于一个实际问题，或者一个创意，当问题或创意提出之后，我们通过开展技术探索和市场调查等活动，来研究系统的可行性和可能的解决方案，从而确定待开发系统的总体目标和范围。

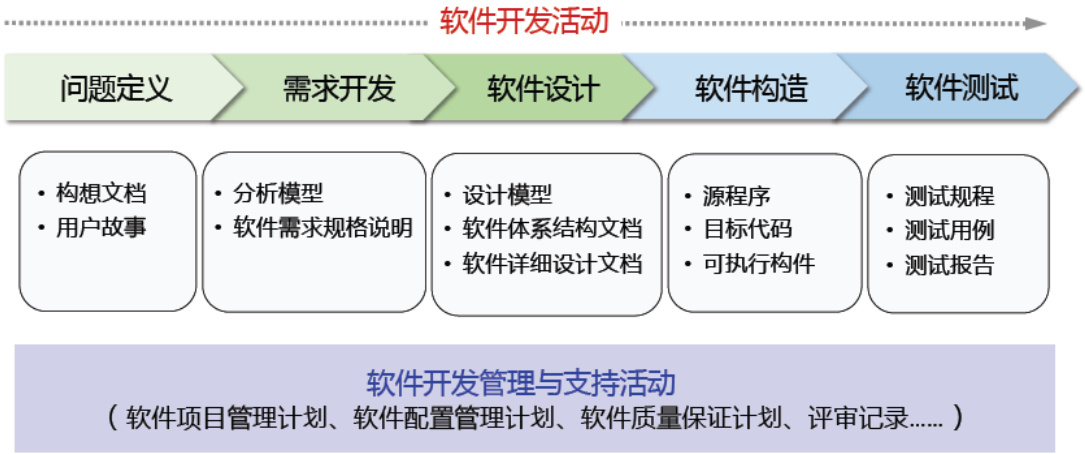


图 2.1 软件过程的一系列活动

以“微信抢票应用”为例，说明一下过程。

过去学生参加学校举办的活动或演出，总是需要排队买票或领票，这个过程是非常烦人的一个过程，解决这个问题，有不同的解决方案，可以开发一个 Web 应用，也可以开发一个独立的手机 APP，还可以做一个微信应用，通过比较这不同方案，我们可以看出微信应用是一个成本最低，开发最快的一个解决方案，而且学生几乎每个人都使用微信，微信应用可以更容易地被学生接受和认可，因此，我们最终确定系统的目标是开发一个微信抢票应用，有了微信抢票应用，再参加学校的活动和演出，就是一个非常愉悦的过程。

在可行性研究之后，还需要进行需求的开发和定义。首先要收集用户的需求，对所收集的需求进行分析、整理和提炼，来理解和建模系统的行为。在这个过程中，可能还要返回去继续收集更多的需求，再对系统的行为进行明确之后，还要用文档的形式把待开发系统的行为定义出来，并且来检查和确认这个文档是不是满足用户的要求。在确认的过程里，还要反复去收集、分析、再补充这样的一个过程，确认通过之后，形成一个正式的软件需求规格说明书，这个需求规格说明就是作为后续开发的一个基础。需求分析过程如图 2.2 所示。

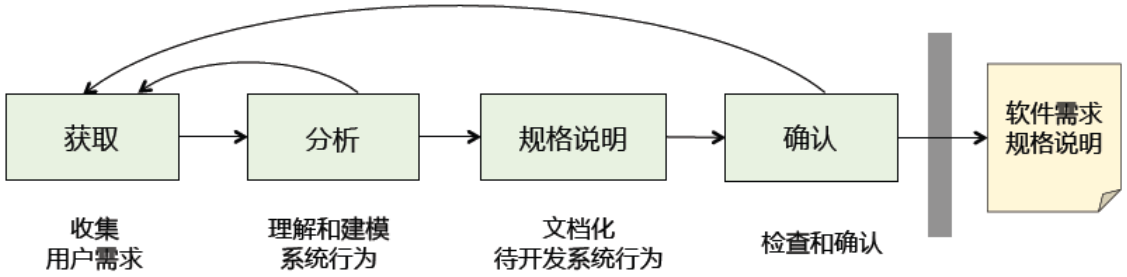


图 2.2 需求分析过程

有了需求规格说明之后，需要对软件进行设计，然后形成软件设计说明书，这个设计的活动包括以下几个方面，首先是要对软件的整体结构进行设计，然后定义出每个模块的接口，并且进一步地设计每一个组件的实现算法和数据结构，同时还要对整个系统的数据库进行设计。软件设计过程如图 2.3 所示。

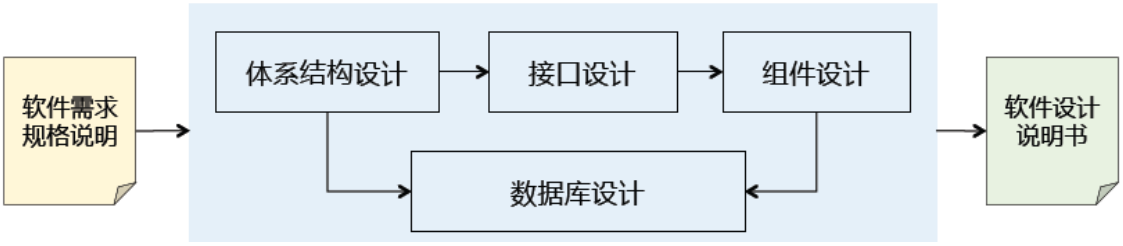


图 2.3 软件设计过程

在设计完成之后，还需要通过构造活动，把设计转化成程序代码，首先我们要理解系统的模型，编写代码，进行代码的审查和单元的测试，还要进行代码优化，最终要构建系统，并且集成联调。这一个过程是一个复杂而迭代的过程。软件构造过程图 2.4 所示。

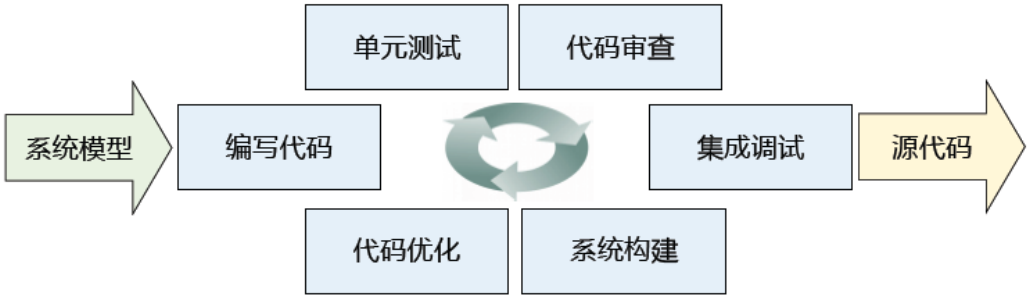


图 2.4 软件构造过程

在软件构造完成之后，还要进行软件产品测试，测试是有不同层次的，包括：单元测试、子系统测试、系统的集成测试和验收测试。在不同的层次上来保证每一个模块、整个系统和最终产品的质量。软件测试过程如图 2.5 所示。

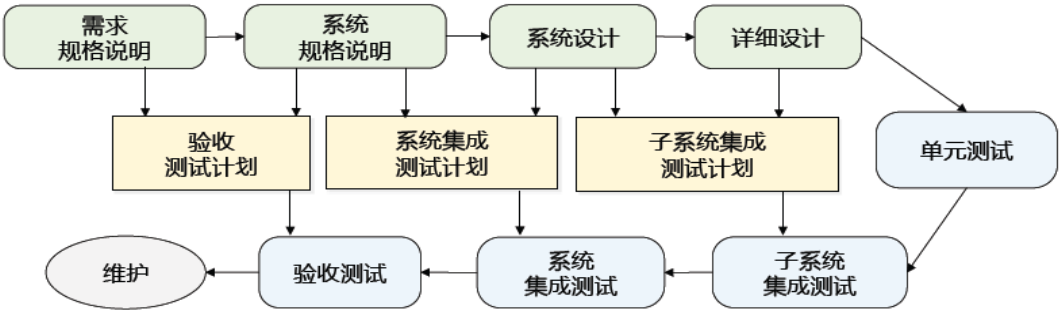


图 2.5 软件测试过程

测试通过以后，产品就可以发布了，但是系统投入使用还会进行不断地修改，来适应不断变化的需求。

应该说完全从头开发的系统是很少的，整个开发和维护是一个连续交叉的过程。软件维护过程如图 2.6 所示。

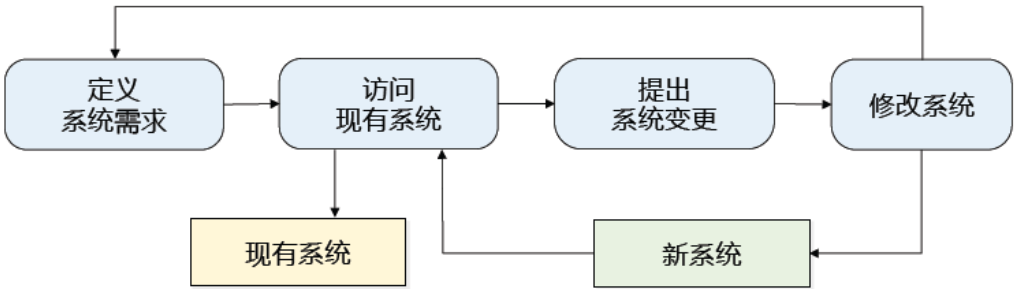


图 2.6 软件维护过程

当新的需求出现之后，要定义这个需求，看现有的系统是不是能够满足当前这个新的需求，如果现有的系统不能满足需求，就要进行进一步地开发，提出系统的变更，针对这个系统的变更，要修改现有的系统，形成一个新的系统，整个的这个过程是一个循环往复的一个过程。

除了基本的实现活动外，软件项目管理和软件配置管理是两个很重要的开发管理活动。软件项目管理是为了软件项目能够按照预定的成本、进度和质量顺利地完成，对人员、进度、质量、成本、风险进行控制和管理的活动。

项目管理主要体现在四个方面：

首先，要明确项目的目标，制定项目的计划，明确项目需要的资源。

其次，要组建开发团队，要明确每一个成员的分工和责任。

然后，在项目这个实施过程中，还要检查和评价项目的总体进展情况，来控制整个项目范围的变更，监控项目进展过程中出现的问题，并及时地纠正这些问题。软件项目管理如图 2.7 所示。

图 2.7 软件项目管理

软件配置管理是通过版本的控制、变更的控制，并且使用合适的配置管理软件来保证整个开发里面的，所有产品配置项，例如代码、文档等的完整性和可跟踪性。软件配置管理如图 2.8 所示。

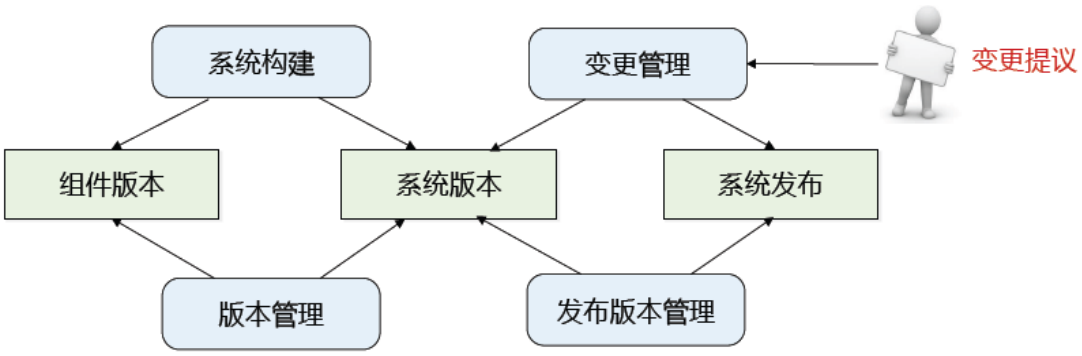


图 2.8 软件配置管理

它主要包括四个基本活动：版本管理是跟踪系统中每一个组件的多个版本，来保证开发者对组件的修改不会产生混乱，系统的构建则是把不同的组件进行编译、链接组成了一个可执行的系统。变更管理是对开发过程中来自用户和开发者的开发请求进行分析和评估，做出适当的这个决策，来决定是否变更和何时变更，当整个开发完成之后，发布版本管理，需要准备发布的软件，并对用户使用的软件进行持续地跟踪。

2.3.2 软件过程模型

软件过程是为了获得高质量软件所需要完成的一系列任务，它定义了完成各项任务的工作步骤，把任务、人员和工具密切地结合在一起。

软件过程模型就是对软件过程的一个抽象描述，常见的软件过程模型，包括瀑布模型、原型化模型、迭代式开发和可转换模型。

瀑布模型将软件开发的基本活动看成是一系列相互独立的阶段，这些活动以线性的方式顺序执行。

原型化模型主要是解决需求不确定问题，原型是一个部分开发的产品，通过原型实现对系统的理解，有助于明确需求和选择可行的设计策略。

迭代式开发是把描述、开发和验证等不同活动交织在一起，通过在开发过程中，建立一系列版本，将系统进行逐步的交付和演化，从而实现软件的快速交付。

可转换模型是利用自动化的手段，通过一系列的转换，把需求规格说明转化为一个可交付使用的系统。

瀑布模型是在 1970 年提出的，直到 20 世纪 80 年代早期，它一直是唯一被广泛采用的软件开发模型。它把软件的生命周期划分为：需求定义与分析、软件设计、软件构造、软件测试和运行维护等若干基本活动，并且规定了这些活动自上而下相互衔接的固定次序，如同瀑布流水一般。

如图 2.9 所示为瀑布模型。

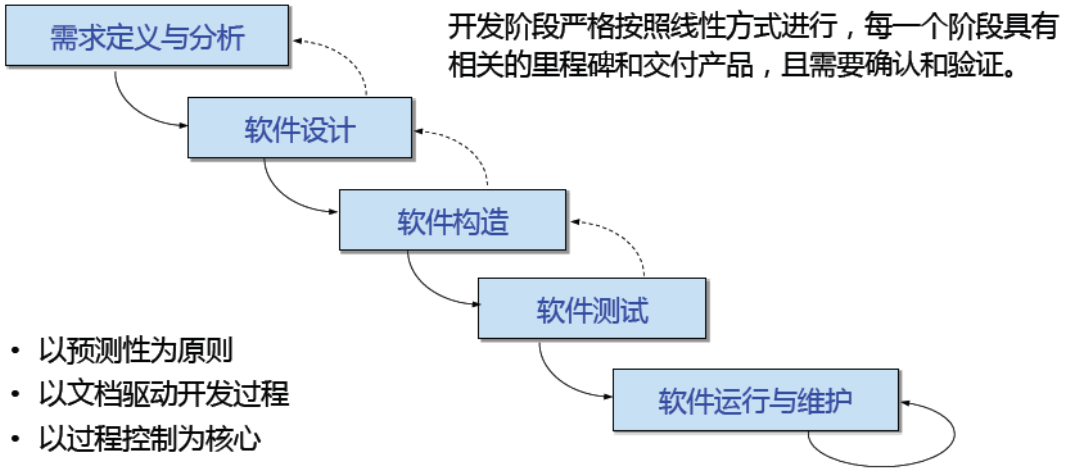


图 2.9 瀑布模型

在瀑布模型中，软件开发的各项活动严格按照线性方式进行，当前活动接受上一项活动的工作结果，实施完成所需的工作内容，当前活动的工作结果再进行验证，如果验证通过就把这个结果作为下一项活动的输入，继续进行下一项活动，否则返回修改。

瀑布模型强调软件文档的作用，要求每个阶段都要仔细地进行验证。

软件的行为只有在运行过程中才能显现出来，因此，瀑布模型只有到测试阶段才能真正地验证和确认软件的功能和性能，但是这个时候所有的代码都已经开发完成，很难返回去纠正需求的问题和设计的缺陷，显然这种模型虽然对各个阶段进行严格控制，但却缺少了对变化的适应。

瀑布模型看似美丽却不现实，目前已经很少在业界使用，它的主要问题在于各个阶段的划分完全固定，阶段之间产生大量的文档，增加了开发工作量。由于开发过程是线性的，用户只有在整个过程结束时，才能看到开发成果，开发过程中间很难响应用户的变更要求，早期的错误也要等到开发后期的测试阶段才能发现，这样会产生严重的后果。因此，瀑布模型仅适用于软件需求在开发初期就可以被完整地确定的项目，而且用户使用的环境也要很稳定，显然这样的要求是不现实的。

软件开发作为一个问题求解过程，一般来说软件需要解决以前，从未解决的问题或者当前的解决方案需要不断更新，以适应业务环境的不断变化，因此，软件开发具有迭代性，需要不断的反复尝试。通过比较和选择不同的设计，最终确定满意的问题解决方案。

从瀑布模型的起源来看，它借鉴了硬件领域的做法，是从制造业的角度看待软件开发。制造业是重复生产某一特定的产品，但是软件开发并不是这样。随着人们对问题的逐步理解，以及对可选方案的评估，软件在不断地演化，因此，软件开发是一个创造的过程，而不是一个制造的过程。在实际的软件开发过程中，最常见的问题是需求的不确定性，用户往往可以很清楚地描述现实中遇到的问题，但是由于所开发的软件在现实中并不存在，因此，用户也说不清楚到底需要什么样的功能，才能解决当前的问题。

在这种情况下，如果用户能够看到软件的操作界面，并且实际操作的话，马上就会明白这些功能是不是自己需要的，所以，原型化模型可以帮助开发人员评价和分析不同方案的实现效果，最终决定更为合适的设计策略。

在上述情况下，我们都实现了产品的一部分，这个部分的产品就称为原型。

原型化模型需要迅速建造一个可运行的软件原型，使用户和开发人员对系统的相关方面进行检查，以决定是否合适和恰当。原型化开发有很多种方法，既可以是可操作的软件界面，也可以是纸上原型，一种纸上原型方法是在纸张和卡片上，手绘或打印界面的元素，再把他们组合拼接，粘贴到一个背景板上，构造成模拟真实产品界面的原型。相比于可操作的软件界面来说，这种方法可以构建得更快，修改更灵活。

今天的商业环境要求软件企业更快速地推出新产品，再加上用户需求存在不确定性和持续变化的特点，传统的瀑布模型无法适应实际需要，必须使用新的过程模型来缩短软件项目的开发周期，迭代式的开发使得软件系统能够逐步地进行交付，开发人员在完成一部分功能之后，形成一个产品版本，然后将其发布给用户使用，当用户使用第一个版本的时候，开发人员继续开发下一个版本，如此迭代循环，这样做不仅可以缩短产品的开发周期，还可以更好地获得用户对产品的反馈，增量模型和迭代模型是迭代式开发的两种形式。

在增量模型中，首先定义一个小的功能系统，然后在每一个新的发布中，逐步增加功能，直到构造出全部的功能。增量模型如图 2.10 所示。

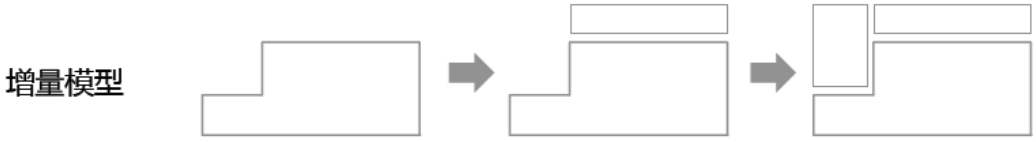


图 2.10 增量模型

在迭代模型中，一开始就提交一个完整的系统，但是每个功能并不完善，然后在后续的发布中再对其补充、完善。迭代模型 2.11 所示。

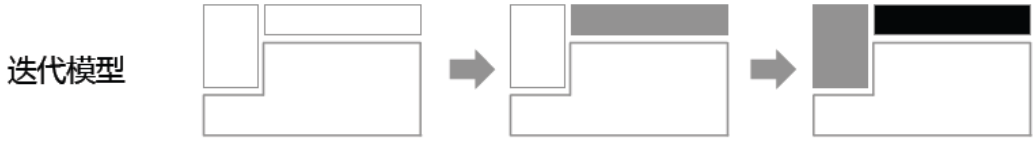


图 2.11 迭代模型

增量模型和迭代模型的区别如图 2.12 所示。

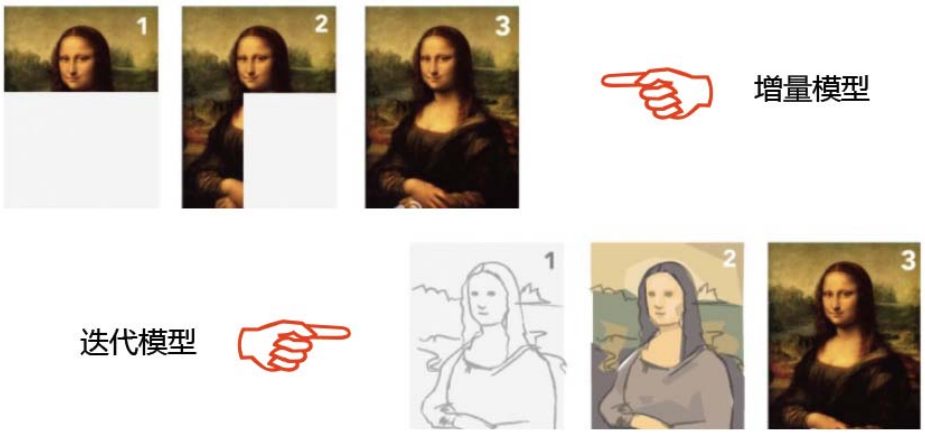


图 2.12 增量模型和迭代模型的区别

这副图很形象地说明了增量模型和迭代模型的区别。

第一种方式是每次完成整个画的一小部分，然后逐步增加新的部分，直到整个画完成，这就是增量的工作方式。

第二种方式是先画出整个画的整体轮廓，然后多次涂色，逐渐形成最终产品的完整效果，这就是迭代的工作方式。

可转换模型是采用形式化的数学方法描述系统，并利用自动化手段，通过一系列转换，将形式化的需求规格说明变为可交付使用的系统。由于数学方法具有严密性和准确性，形式化方法所交付的系统具有较少的缺陷和较高的安全性。这种模型特别适合于那些对安全性　可靠性和保密性要求极高的软件系统。这些系统需要在投入运行前，进行严格的验证，当然，建立形式化的数学描述是一个比较困难的工作。目前这种方法主要还是应用于有限状态的嵌入式系统中。

前面介绍的各种模型各自特点不同，适用的场合也不同，对于实际的软件开发项目来说，应该根据软件系统的特点，选择合适的软件过程模型。

下面分析两个软件系统的例子。

第一个例子是汽车制动防抱系统，也称 ABS。

它是一种具有防滑防锁死等优点的汽车安全控制系统。我们在雨天或雪天开车时，路面非常湿滑，这时突然踩刹车，会造成车轮抱死，使车辆侧滑出现危险，ABS 系统的作用是在汽车制动时，自动地控制制动器的制动力大小，使车轮不被抱死，从而保证车轮和地面的附着力。

这是一个嵌入式控制系统，对安全性和可靠性要求极高，必须在投入运行前进行质量验证。

在我们前面学过的模型中，哪一种适合这种系统的开发，显然，可转换模型是最合适的。

第二个例子是大规模在线公开课程网站。

网站上可以开设大量的课程，教师可以把课程录像、课件和参考资料等公布在上面，学生可以进行自主地学习，这是一种新型的网络教育模式，将教育、娱乐和社交网络有机地结合在一起，这种系统的需求会经常发生变化，业务模式也存在不确定性，而且系统应该易于维护和修改。

在这种情况下应该采用什么类型的过程模型呢？显然迭代式开发是最合适的。

软件过程模型的发展体现了软件工程理论的发展，从早期完全无序的状态，到严格阶段控制的瀑布模型，以及现在流行的应对变化的迭代模型，其目的都是要提高软件开发效率和产品质量，更有效地解决软件开发面临的问题。

2.3.3 敏捷开发过程

随着互联网技术和应用的迅速发展，软件开发面临着需求频繁变化和快速交付的挑战。在这种情况下，人们开始尝试一种新型的敏捷开发方法，敏捷方法采用增量和迭代的开发过程，强调团队紧密的协作。这种方法已经取代了传统的瀑布模型，被众多的软件企业广泛地应用。传统的瀑布模型是最典型的预见性开发方法，它要求需求在开发初期就完全确定，并且在整个过程中很少变化，整个开发过程是计划驱动的，严格按照需求、设计、编码、测试、维护的步骤顺序展开。

那么，软件开发是否可以实现一个完整详尽的计划呢？

软件项目能否预见考虑到所有的风险呢？

由于软件是要解决从未解决的问题，因此，项目中存在难以预知的所有内容和风险，需求是不确定的，如果一开始我们知道一个预想的结果，并且是按照详尽的计划来执行的话，那么最终产生的产品往往不是用户需要的。

软件开发应该是一个逐步认知和明晰的活动，通过不断地逼近，最终产生用户满意的产品。

软件开发到底是要获取一些更有价值的交付产品还是只是按照计划完成工作内容，显然，应该更专注于交付有价值的产品。也就是说高质量的交付产品是最重要的，而这个产品不是一次构建形成，是需要经过迭代演进来形成。

互联网时代是一个快鱼吃慢鱼的时代，快速地推出产品就能够占领市场的先机。在互联网时代，用户的变化和对创新的要求是非常高的，软件的产品要追求创新，要快速地响应用户的变化。敏捷开发就是一种有效应对快速变化需求，快速交付高质量软件的迭代和增量的新型开发方法。它强调更紧密地团队协作，关注可工作的软件产品。这是一种基于实践，而非基于理论的开发方法。

敏捷方法强调适应而非预测，由于软件需求很难预测，那么按照预测产生的结果，往往不是用户需要的产品，所以，软件开发应该是一个自适应的跟踪过程，通过适应和逼近，最终产生用户满意的产品。

2001 年 2 月，17 位软件从业者在美国的一个滑雪圣地举行了为期三天的会议，主要是讨论广大开发人员对软件开发的共同见解。这次会议的主要成果是决定使用敏捷这个术语来表达共同性，相对于瀑布模型的僵硬化的过程，敏捷更加强调灵活和快速，会议制定了敏捷宣言及其 12 项准则。

敏捷宣言只有短短的四句话：（1）个体和交互胜过过程和工具；（2）可以工作的软件胜过面面俱到的文档；（3）客户合作胜过合同谈判；（4）响应变化胜过遵循计划。

（1）个体和交互胜过过程和工具

在 20 世纪 80 年代到 90 年代中期，过程改进的活动非常盛行，当时推出了 CMM 等诸多的质量标准体系，虽然这个体系在一定程度上改善了软件过程的质量，但是也造成了软件开发管理越来越僵化。实际上软件开发是一种创造性的活动，没有优秀的个人和团队的协作，再强大的方法和工具都是摆设。

（2）可以工作的软件胜过面面俱到的文档

传统的瀑布模型强调文档的作用，但是文档对用户来说是不是真的有价值，应该说面面俱到的文档对于客户来说并不重要，用户需要的是一个能够运行起来，能够解决实际问题的可以工作的软件。面面俱到的文档对开发来说也不重要，谁也不愿意去读非常厚的上百页的文档，对开发来讲最好的文档就是代码和团队。

（3）客户合作胜过合同谈判

软件开发的目标是提供给客户满意的软件，只有客户才清楚什么样的产品是满足他的要求的。敏捷开发提倡客户和开发团队，在一起密切地合作中，经常性地提供反馈，通过短期的迭代，尽早地沟通和反馈，把问题及早地暴露出来，从而避免了在后期造成更大的影响。

（4）响应变化胜过遵循计划

计划赶不上变化，敏捷项目承认开发过程中具有不确定性，因此，不会在开发的初期制定长期复杂的完美计划，所有的开发过程都是建立在现实的反馈基础上。敏捷方法依据一些简单的实践和规则，通过经验性的一个过程控制，来处理开发中的复杂问题，通过迭代和不断地响应变化，来消除开发中的不确定性。

敏捷宣言的价值观就是自组织团队与客户紧密协作，通过高度迭代式，增量式的软件开发过程来响应变化，并在每次迭代结束时，交付经过测试的有价值的软件，胜过与客户确定合同后在初期制定并遵循基于活动的完整计划。在重量级过程和工具的指导下，通过完成大量文档，进行知识传递、最后交付需求。

敏捷宣言的 12 个基本原则强调尽早和持续的交付有价值的软件，对需求提出变更，交付的周期越短越好，用户应该和开发人员在一起工作，要善于激励团队成员，要面对面地进行交流，可用的软件是衡量进度的主要指标，要保持恒久稳定的进展速

度，坚持不懈地追求技术卓越和良好的设计，要做到简单，团队应该是自组织的，要定期的反省，并相应地调整其行为。

瀑布模型与敏捷方法的主要区别如图 2.13 所示。

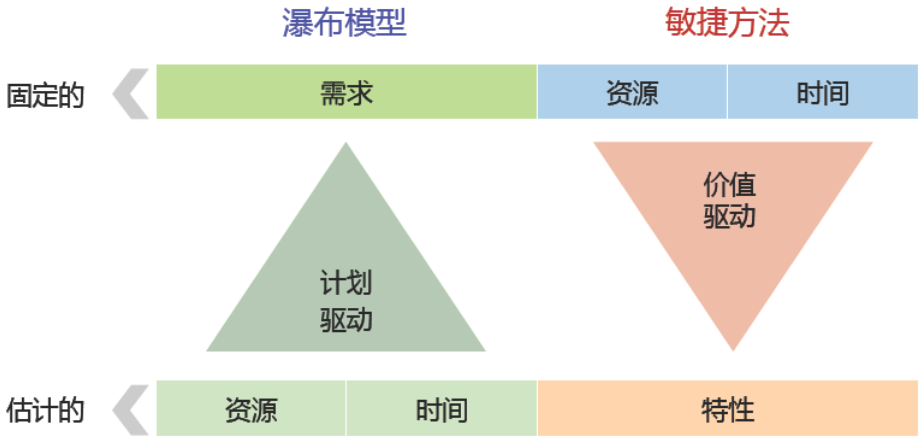


图 2.13 瀑布模型与敏捷方法的区别

瀑布模型假设需求是固定不变的，根据这个需求来估计所需要的资源和时间，制定完美的计划，整个的开发过程是完全按照计划来驱动的。

敏捷方法是认为需求是不确定的，它是在固定的时间和资源范围内来估计出所需要实现的产品特性，通过这个价值的驱动，来实现用户需要的功能，好的产品是开发出来的，不是设计出来的。

传统的开发方法虽然产生了很多设计的文档，但最终交付的产品是很难满足客户需求的，而敏捷开发方法是在开发过程中构建、演化逐渐地逼近最终交付的用户满意的产品。

敏捷方法是一组轻量级开发方法的总称，包含了很多具体的开发过程和方法，如图 2.14 所示，在这里面最有影响的两个方法就是极限编程和 Scrum 开发方法。

图 2.14 敏捷轻量级开发方法

Scrum 方法更加偏重项目管理实践，它规定了产品订单、各种会议以及开发这个人员的角色。极限编程则是偏重变成的一些实践活动，包括重构、测试驱动的开发、结对编程等。

Scrum 是一种应用很广泛的敏捷开发方法，这个单词的英文含义是橄榄球运动中的一个专业术语，表示争球的动作，把一个开发过程命名为 Scrum，就是形容开发团队在开发一个项目的时候，所有的这个团队成员能够像打橄榄球一样迅速富有战斗的激情，你争我抢完成进攻，通过一个逐步逼近的方式取得最后的胜利。

Scrum 是一种兼具计划性与灵活性的敏捷开发过程，它把整个开发过程划分为若

干个更小的迭代，每一个迭代周期称为一个冲刺。

首先产品经理根据用户需求和市场需要，提出一个按照商业价值进行排序的客户需求列表，也就是产品订单。在产品迭代的开始迭代规划会议，要从这些产品订单中挑选出一些优先级最高的，形成迭代的冲刺任务，一般一个冲刺其周期是两到四周。在迭代过程中，会进行每日战略会议，检查每天的进展情况，在迭代结束的时候，就会产生一个可运行的交付版本，由项目的多方人员参加这个版本的演示和回顾的会议，最终决定这个版本是否达到了发布的要求。

Scrum 迭代开发是把整个软件生命周期分成多个小的迭代，每一次迭代就是一个小的瀑布模型，它包括了需求、设计、实现和测试等活动。在每一次迭代的时候，要生成一个稳定的和被验证过的软件的版本。每一个迭代都要建立在稳定的质量基础之上，在一次迭代的过程中，不允许变更交付的产品和交付的日期的，也就是说在一次迭代中，需求是不发生变化的。如果产生了相应的变化，就要进行分解和澄清，如果存在这个很大的争议，就把它看成一个变更，放到下一个产品的订单再进行考虑。

敏捷开发方法已经成功地应用于诸多的软件企业之中，并在 ISO9000 美国军方软件开发标准和 2013 年发布的新版 PMBOK 中，也把敏捷方法作为新增的内容。

敏捷方法正在走向成熟，并在实践的基础上获得进一步的更大的发展。

2.4 讨论题

1、为什么提出软件生命周期，有什么作用解决什么问题？

软件工程的工作是围绕着软件生命周期来展开和进行的。什么是软件生命周期？为什么要提出软件生命周期？提出软件生命周期的初衷是什么，要解决什么问题？

2、有几种软件生命周期模型？它们有什么差异吗？这些差异对软件开发有什么影响？请分享你的见解。

2.5 测试题

单选题 （10 满分）

1.下面的（ ）决策是在需求分析时做出的。

A.自动售票机系统的开发时间预计是 6 个月

B.自动售票机系统由用户界面子系统、价格计算子系统以及与中心计算机通信的网络子系统组成

C.自动售票机系统已经达到交付的要求

D.自动售票机系统将为使用者提供在线帮助

2.下面的（ ）决策是在系统设计时做出的。

A.自动售票机系统的开发时间预计是 6 个月

B.自动售票机系统由用户界面子系统、价格计算子系统以及与中心计算机通信的网络子系统组成

C.自动售票机系统已经达到交付的要求

D.自动售票机系统将为使用者提供在线帮助

3.下面的（ ）是软件构造活动的任务。

A.构建软件组件
B.设计用户界面
C.实施组件的单元测试
D.评估组件的质量
E.选项 A 和 C
F.选项 A、B、C 和 D

4.瀑布模型是（　）。
A.适用于需求被清晰定义的情况
B.一种需要快速构造可运行程序的好方法
C.一种不适用于商业产品的创新模型
D.目前业界最流行的过程模型

5.增量模型是（　）。
A.适用于需求被清晰定义的情况
B.一种需要快速构造核心产品的好方法
C.一种不适用于商业产品的创新模型
D.已不能用于现代环境的过时模型

6.原型化模型是（　）。
A.适用于客户需求被明确定义的情况
B.适用于客户需求难以清楚定义的情况
C.提供一个精确表述的形式化规格说明
D.很难产生有意义产品的一种冒险模型

7.开发一个支持 3D 打印的操作系统最适合采用（　）。
A.瀑布模型
B.原型化模型
C.增量开发
D.可转换模型

8.开发一个铁路信号控制系统最适合采用（　）。
A.瀑布模型
B.原型化模型
C.增量开发
D.可转换模型

9.下面的（　）不是敏捷开发方法的特点。
A.软件开发应该遵循严格受控的过程和详细的项目规划
B.客户应该和开发团队在一起密切地工作
C.通过高度迭代和增量式的软件开发过程响应变化
D.通过频繁地提供可以工作的软件来搜集人们对产品的反馈

10.关于 Scrum 的每一次冲刺（Sprint），下面的（　）是正确的。
A.Sprint 是一个不超过 4 周的迭代，其长度一旦确定，将保持不变。
B.Sprint 的产出是一个可用的、潜在可发布的产品增量。
C.Sprint 在进行过程中，其开发目标、质量验收标准和团队组成不能发生变化。
D.以上所有选项

第 3 部分 需求获取

3.1 本章的教学目标

通过本章的学习，使学生初识软件需求问题、理解软件需求的一种思维方法--结构化思维方法和软件需求的多样性，掌握软件需求工程的过程，包括：需求获取、需求分析、需求规范、需求验证、需求管理等。

3.2 本章的教学方法

本章的教学方法采用“翻转课堂式”教学模式，学生可以通过互联网使用“学堂在线”网上优质教育资源，选择由清华大学刘强副教授主讲的“软件工程”课程的“第 7 章 需求获取” 课外观看学习，网址：
http://www.xuetangx.com/courses/course-v1:TsinghuaX+34100325X+sp/about ，通过视频中教师的讲解，回到课堂上教学交流讨论，并相关的完成习题。

通过理论教学、课堂讨论、适当的布置课外习题作业来完成学习。

3.3 本章的教学内容

本章主要阐述软件需求问题，理解软件需求的一种思维方法--结构化思维方法，介绍软件需求的多样性和软件需求工程的过程（包括：需求获取、需求分析、需求规范、需求验证、需求管理等）。

3.3.1 需求工程师

需求分析是软件工程中的重要步骤，是决定软件项目成败的关键影响因素之一。需求阶段的错误在后期的纠错成本中，将远远高于软件设计和实现阶段的纠错成本，因此，需求工程成为软件工程和系统工程重要的分支领域之一。

在需求工程中，主要关注的是软件和系统需求的获取、建模、分析、验证和管理。

在下面几周的学习过程中将学到：

- 如何产生软件产品的新的创意
- 如何和干系人进行有效地沟通
- 如何基于目标、场景和主体来建模软件系统的需求
- 如何用图形化的方法对软件系统的行为和结构进行分析

注意两点：（1）将问题的分析和解决方案的生成区别开来；

（2）要根据软件项目本身的规模、人员的技能、客观的条件、项目的成本，来选择合适的需求获取技术和建模方法，选择最合理的软件系统设计方案。

下面看一个故事。

有一天软件工程师、计算机科学家和数学家在圆明园相遇，他们同时看到湖面上游来了一只黑天鹅，在看到这一现象之后，软件工程师在他的世界模型中将天鹅建模为黑颜色的。计算机科学家在看到这一现象之后，在他的世界模型中，将天鹅中的一部分建模为黑色的。而数学家在见到这个天鹅之后，得出的结论是在圆明园有一个湖，湖上至少有一只天鹅，它的一面是黑色的。

从这则故事我们可以看出，软件工程师的思维方式是做出尽可能简化问题，则更适用于解决具体领域的工程应用问题。计算机科学家的思维方式是要将眼前的问题看成是一个更具一般性问题的特例，因此，需要找出不失一般性的解决方法，如果更具一般性的问题解决了，那么眼前的问题自然迎刃而解。计算机科学家的思维方式，则追求的是精确性。

作为一位当代的需求工程师：应该具有的是分析问题和解决问题的能力，以及主动参与人际交流和沟通的能力；掌握软件工程师的基本技能和知识；对应用领域的深厚的了解；能够进行书面语言的组织和表达等。

一名优秀的需求工程师，她将以下的内容设为她自己的努力的方向和目标：

（1）识别系统的错误的假设

（2）确保描述的一致性

（3）对标准和规范的依从性

（4）减少组织和个人间的误解

（5）提高支持人员的反应速度和效率

（6）提升客户满意度

（7）撰写优质的需求文档

要值得注意的方面就是要排除一切的干扰，避免沉默、避免过度规约、避免含糊和矛盾的需求描述、避免向前引用以及不切实际和一厢情愿的假设。

3.3.2 需求定义

来自美国科罗拉多州立大学科全分校的 Alan M. Davis 教授给需求的定义是这样的，需求是对外可见的系统特征，它强调的是需求的对外可见性，以及需求关注的是系统的特征描述。

需求管理主要由三项任务组成，首先，是以学习为主的需求获取过程，其次，是对获取到的需求进行优选剪枝的过程，最后，是需求的文档化过程，也就是撰写需求规格说明书。

IEEE 标准对需求的定义是：人们要解决的某个问题或实现某个目标所需的能力和条件，是系统或其组成部分，为满足某种书面合同规定所要具备的能力。

需求将作为系统开发、测试、验收、提交的正式文档依据，从这个定义中我们可以看出，它强调的是合同相关的特性。

另一个经典的需求定位来自 Herber Simon 教授，他给出的描述是：

每一个“人造物”都是一个内部环境与外部环境的“接口”。内部环境是指人造物本身的设计组成，外部环境是指人造物的周边及其作用环境，对这个接口的描述既是需求。

从这个定义我们看出，软件作为一种人造物，它为了很好地满足外部环境，对它的要求就必须要有一个针对内外环境的接口的一个非常精确的描述，只有这样才能够让软件系统的行为和外界的要求合理适配，需求描述的难点和挑战就在于它是连接应用领域现象与机器领域现象的桥梁。

我们要从应用领域的固有性质和用户待解决的需求描述转化为可以用计算机软件，实现的系统行为的描述。

我们知道需求是用户希望系统满足一定的目标，而体现出来的行为，在需求的描述中，要反映出分析师对问题领域的清晰的理解，给出系统使用的上下文和场景，因此，在需求的定义中，要试图回答以下的主要问题：

- 我们为什么要设计这个系统

- 系统将由谁使用
- 系统要做什么
- 涉及哪些信息
- 对解决方案是否有额外的限制
- 如何使用该系统
- 质量指标约束要达到何种程度

通过对这些问题的解答，就能够比较清楚地了解需求的核心内容，系统需求规约是由应用领域和机器领域中共享的事物组成的。

以禁止学生以外的人员参与校内活动抢票，以这一问题的解决为例来说明。

问题的本质是要区分：非法闯入者和合法的学生用户，而这个问题的解决投影到机器领域，就转换为了通过对学生卡，微信用户名和密码的检查，实现身份区别。

这里涉及到三类对象，一类是由人机共享的事物，包括学生卡号、密码、微信号、键盘输入等，它是跨应用领域和机器两个领域的现象，此外，还有专属于应用领域，而机器领域看不到的现象。它们属于应用领域固有的事物，必须要投影到机器可以接受的输入，才能由机器识别到，包括：学生、系统管理员、密码分配过程、学生名册、发给学生的带密码的不干胶贴等。还有专属于机器领域的现象，这些现象是实现系统行为必不可少的，但是却是对用户来说不可见的，这可以包括加密算法、微信认证、内存管理、安全套接字等相关的技术实现模块。

然而，在应用领域和机器领域之间的边界划分，也是可以发生迁移的。比如：在一个电梯控制系统中，原本属于应用领域的乘客等待和乘客在电梯内这两个现象，当我们通过增加传感器、监视器等设备的时候，我们就改变了问题的边界，乘客等待和乘客在电梯内变成了机器可以观测到的状态，于是它们也就进入了二者之间共享现象的集合，系统边界的迁移意味着系统的问题性质发生了改变，从而系统的设计也将随之改变。

在定义需求的过程中，我们要注意将问题与解决方案分开，建立单独的问题描述文档，这样可以深入地剖析待解决的问题，将问题描述与实现描述分开，可以使得干系人之间，可以就待解决问题的本身进行充分地磋商和讨论，对多个候选的设计方案进行优选。问题描述也可以作为测试用例设计的主要信息来源，如图 3.1 所示。

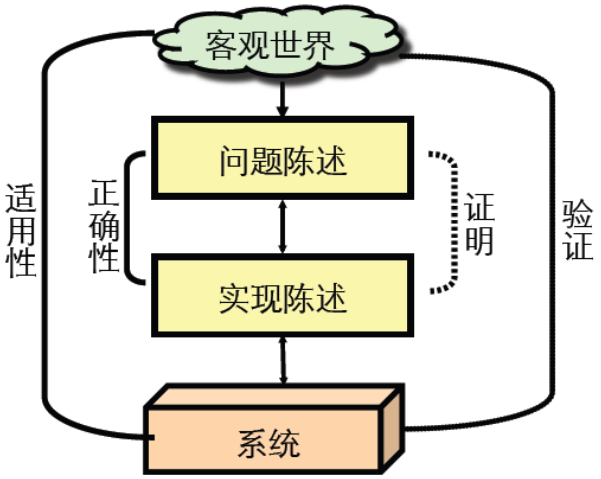


图 3.1

从图 1 我们可以看出：问题陈述和实现陈述之间要进行模型正确性的证明；在客观世界和已经设计出来的系统之间要检查和验证；系统与客观世界的适用性，确保问题陈述与干系人的需求相呼应；领域性质是无论系统存在与否都存在的应用领域性

质。

以抢票系统为例，演出票的数目是不会系统的存在而发生改变的，因此它是应用领域固有的性质，而需求是由于系统的存在而使能的，新的应用领域性质，因为抢票系统是通过微信平台快速公平便捷的，完成了演出票的分配，因此，它的需求就是一种，由于抢票系统存在而带来的新的领域的性质，抢票系统的规约描述，描述的是抢票系统的工作流程，用户身份认证的手段，二维码扫描、验票的手段，那么它实际上是描述系统，为了满足用户的需求，而具有的行为，需求证明是为了确保两件事。

首先是运行在机器上的软件满足规约描述。

其次，是针对给定的领域性质，规约符合用户需求，需求验证的手段，则是要从完备性来入手，首先是不是所有的需求都已经找出，其次，是不是所有有关的应用领域性质都得到了定义，假定待解决的需求是当飞机在跑道上移动时，反推器处于工作状态，我们做出的两条领域性质的假设是当且仅当轮胎转动时产生轮脉冲，当且仅当飞机在跑道上移动时轮胎转动，由这样的两条性质，我们推导出系统的规格说明是，当且仅当有轮脉冲产生时反推器工作，那么这样的一组假设是不是正确。

如果我们所做的领域模型出错时，会有什么样的后果，仔细分析后，你会发现实际上我们做出的第二条领域性质的假设是存在问题的，当飞机在跑道上移动时，轮胎转动，并没有覆盖客观世界的真实情况，在雨天坡道路滑的时候，飞机在跑道上移动，但是轮胎有可能没有转动，此外，当飞机在跑道以外的地方移动的时候，轮胎也是转动的，所以这两种情况是属于没有被我们现在的领域性质覆盖的情况，因此，飞机的行为是与预期不符合的。

我们要注意在需求的描述过程中，避免含糊的、错误的、不完整的、矛盾的和无法测试的需求，这样几种情况，含糊的需求往往是在需求的描述中用了虚指的代词，对这样的情况的处理应该注意明确指出所指代对象是什么，错误的需求描述则往往是忽略了一些明确的事实，比如：如果你说所有的系统将九月作为财政年度的起始时间，无疑是以偏概全的一种假设，要注意需求的完整性，就要考虑到各种可能的情况。比如：错误信息显示在屏幕的第 24 行，那么我们在这里忽略了出错的信息超过一行的情况，因此，这个需求是无法被满足的，要注意对变量定义的，逻辑上的一致性和无矛盾性。

我们不能把 C 首先定义为 A+B 的和稍后又把它定义为 A-B 的差，无法测试的需求往往是针对非功能性需求的。

好的需求描述是可以度量的，它给出项目成功的必要条件，在需求项目的度量指标中，针对单个需求项目的度量，主要包括准确性、正确性、明确性、可行性、可证明性。

针对需求集合的质量，包括它的现实性、精确性、全面性和一致性。

这些需求的质量监测标准，为我们评估自己的需求文档的质量，给出了客观的参照依据。

3.3.3 需求分类

软件需求的分类可以从多个维度上进行，它们可以彼此交叉。

首先，按照软件需求修饰的对象的不同，可以将其分为软件产品的需求和软件过程的需求，产品需求主要约束的是软件产品和软件制品本身的属性。而软件过程需求则是修饰或者限制软件开发过程的要求。

产品需求又可以细分为：功能性需求和非功能性需求。

功能性需求指代软件产品的功能特性，而非功能性需求则是软件产品的质量属

性，是在功能性需求满足的情况下的进一步的要求。

按照软件需求面向对象的不同，以及其抽象层次和详细程度的不同，又可以将需求分为：业务需求、用户需求、系统需求和软件设计规约，其中业务需求主要是针对业务部门的分析人员的，用户需求针对客户方、承包方的管理人员、最终用户、客户工程师和系统架构师。

系统需求则是由最终用户、客户工程师、系统架构师和承包方的程序员来关注的。

软件系统的设计规约则是由客户工程师参考、系统架构师和承包方的程序员重点关注的。

（1）业务需求

企业的业务需求是指那些其达成会影响企业完成，组织目标的需求项。

企业业务需求是关于业务的陈述和需求如何被实现无关，无论是人工手动完成还是通过系统完成，并不影响企业的业务需求，有时我们也把业务需求称为业务目标。

例如：携程旅行这个公司，它的业务需求是销售飞机票，公司的目标是成为当人们想买飞机票时，首先想到的公司，这是对一个企业的业务需求的陈述。

（2）系统需求

系统需求的满足使得系统实现了预期的功能，它从用户的角度描述系统做什么。而与系统由什么样的软硬件实现是无关的。

一个企业选择实现某个软件系统的首要条件就是该系统需求，应该是满足组织的业务需求的。

下面给出了三条描述：

第一条：卖票系统，需要和用户数据库交互

第二条：新的软件，会使汽车的启动速度加倍

第三条：我们新产品制造的低成本，将会让我们有更高的市场份额，并满足销售目标。

在这三条需求中，其实我们可以看到对预期功能进行定义的，只有其中的第二条，它是一个启动加速的软件，而第一条只是说明了卖票系统它要和用户数据库交互这个事实，那么它实际上是一种设计的约束，而第三条我们新产品制造的低成本，这件事情实际上在说低成本，它是一个对过程的一个约束，而并没有说出产品的一个功能是什么，所以它不隶属于系统需求的范围。

（3）软件需求

在一个软硬件结合系统中，软件需求是指关于系统中软件部分的需求，它的满足将有助于实现整体的系统需求。

这里给出了三个围绕软件的需求的描述的实例。

第 1 条：订票系统软件通过标准的网络服务接口和航班信息交互，这是在说明订票系统软件与外部的其它系统之间的交互方式，是通过网络服务接口，因此，它既是一个软件需求，也是一个关于接口的一个描述。

第 2 条：用户接口需要设置关于用户偏好的颜色和字体大小，这也是一个软件系统的功能，就是要设置用户偏好的颜色和字体大小，而同时它又是一种增强用户体验的一条软件需求，是关于质量和非功能性属性的。

第 3 条：系统的 API 需要同时支持 C++和 Java 来让程序员访问系统服务，那这就意味着这是一条软件系统的设计约束，是对编程语言的要求。

（4）用户需求

系统的用户需求是指其满足程度会影响系统的用户接受程度的需求。有时候我

们也把用户需求称为“用户接口需求”。

例如，一个订票系统的用户接口需要提供和系统交互的选项，或者通过控制命令，或者通过 WIMP 接口，客户希望照相机背后有一个液晶屏幕，这样在拍照之后，可以马上看到照片，这些都是关于用户接口的描述，属于用户需求的范畴。

（5）功能性需求

系统的功能性需求是指满足系统需求时，需要提供的功能有时候功能需求，也被称为行为需求。

对一个订票系统来说，两个可能的功能性需求包括：需要提供一个在任何航班上预留座位的功能和一个通过信用卡付费的功能。

（6）非功能性需

非功能性需求定义的是软件系统及软件开发过程中为满足系统的功能需求而要满足的其它的约束条件，具体细分为：质量相关的需求、依从性需求、体系结构约束和设计开发约束等。如图 3.2 所示。

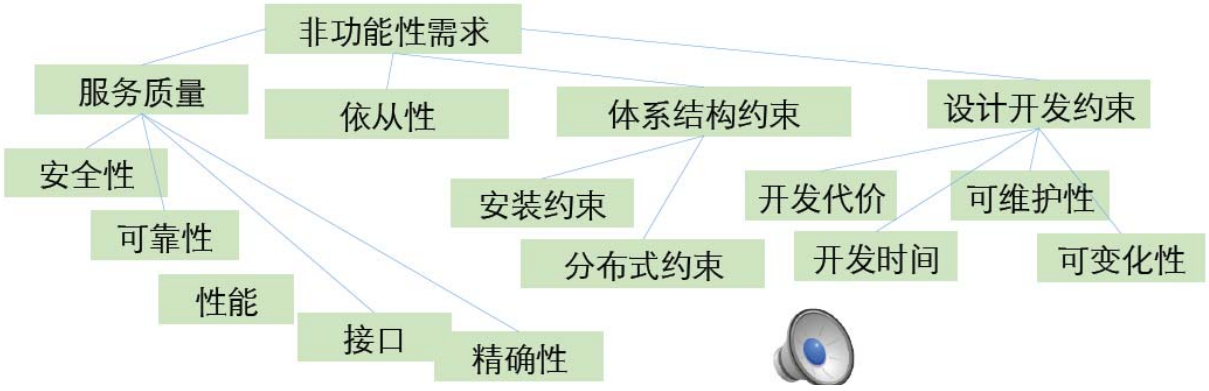


图 3.2

（7）质量需求

质量需求描述软件系统正常工作时需要满足的额外的与质量相关的要求，在软件工程文献中，也称之为“质量属性”，它所应答的是关于“提供的服务好到何种程度”的问题。

例如，在订票系统中，我们要求用户界面要保证用户从选择出发站到终到站的过程，不能超过四次按键，这个是一个关于用户友好性的一条需求，订票系统的订票请求、响应时间要小于 1 分钟，这是一个关于订票系统的响应时间，也就是性能的要求，图书馆的黑名单，应随时响应馆员的查询操作，这是一个对图书馆的黑名单的可用性问题的一个需求。

列车的加速控制软件平均无故障时间是 10 的九次方小时，这是对系统可靠性的一个要求。

（8）依从性需求

依从性需求着重描述软件对国家法律、国际公约社交法则、文化与政治习惯环境标准等约束条件的满足

例如，在列车控制系统中，我们要求两列火车间的最小间距应满足国际铁路运输安全规范中的最坏情况下的停车距离。在会议调度系统中缺省情况下，我们要排除会议所在地的地区的公众假期，这些依从性的需求往往是一个系统它所处的经济、政治上下文要求的一些限制，它的满足是系统正常运行，或者合法运行的一个客观标准。

（9）体系结构设计需求

体系结构设计需求定义的是系统环境，对待设计系统在结构上的约束和限制，

这里主要是从两个方面。

首先是分布式的约束，它要求软件系统的组件满足由于目标组织在地理上的自然分布而导致的对系统设备节点的分布要求，以及数据的分布式存储与处理的要求。比如：在会议调度系统中，它应该与分布在世界各地的参会者的邮件服务系统和电子日程管理系统协同工作，它是由于组织机构本身以及其外部相关系统本身的分布的属性导致的，

另外一类就是安装约束，它要求的是软件系统能够在目标的运行环境下正常的执行。比如：会议调度系统要在微软的 Window8 和 Mac OS10.10 的版本上成功运行，设计开发约束是对软件系统设计过程的约束，具体包括从开发成本、开发时间周期、产品特征的变化性、可维护性、可重用性、可移植性等方面的约束。

例如：清华大学图书馆门户系统的开发成本不应高于 200 万，是对一个项目开发成本的约束。动车控制软件应在两年内投入使用，这是对开发时间周期的约束。会议日程安排系统应根据会议的类型，动态调整会议安排的调度策略，具体的它的变化性包括，对专业会议和内部会议，它的策略应该是不同的，对固定会址和轮换会址的会议类型，它的调度策略也是不同的，对参会人员的重要程度是相同级别还是不同级别，也应该可以动态的设置，所以，这个是对产品的变化性以及适应性的一个约束，这些都隶属于软件的设计开发约束。

（10）需求类型间存在一定重叠

我们需要额外强调的一点是需求的类型之间是存在一定重叠的，比如：功能性需求与非功能性需求间的划分并非是绝对的，可能存在一定的重叠，而顶层的非功能性需求，经过逐步的细化和操作化以后，就转换成了一个系统的功能。

比如：在一个核电站的安全注入系统中，当反应堆正常启动或冷却时，一旦发现冷却剂损失，要将安全注入指示灯打开，这既是一个功能性的需求，又是一条安全性需求。

防火墙管理软件的功能性需求，同时也隶属于安全性需求。

通话系统的来电过滤，常常被看作是功能性的特征，但同时它也和通话者的隐私保护需求相关。

在医疗信息系统中，对患者的病例服务发动拒绝服务攻击，使得医生在手术期间无法访问患者的关键数据，这是关乎数据和患者人身双重安全的需求。

向列车高频发送加速请求，这既是一条安全性需求，又是一条性能需求。

由此可见，需求类型之间它是存在重叠的，这种划分并不是整齐划一的。

（11）需求分类的合理使用

我们要对需求的分类进行合理的使用，我们关注那些特殊的系统需求，关注需求中三类主要的内容。

一是要明确哪些是系统必须支持的行为，这是系统被用户接受的必要条件。

二是要排除那些绝对不可以接受的系统行为，这是导致系统不可接受的那些必要条件，因此，我们一定要在系统行为中，避免这类行为的出现。

三是要明确那些系统最好是支持行为，这是用户对系统的一些偏好的需求，我们要尽量满足，但它又不是必要条件，我们也要对那些适用范围比较有限的关注点和横切许多功能的关注点，区别对待，对二者的处理方法是不同的。

比如 安全性、多语言支持，这类横切关注点，我们可以设计一个通用的解决方案，然后把它和各类系统，能够有机地集成到一起，而对那些适用范围受限的关注点，我们则要个别问题个别分析。

需求的分类，其实主要是为我们抽取需求，提供启发式的规则，使得我们可以

避免忽略了关键类型的需求，而且，对于那些固有的需求类型之间存在矛盾冲突和竞争关系的需求，我们可以在具体的需求条目中去发现这些矛盾，解决相关的冲突。

（12）功能性需求主要关注问题

功能性需求主要关注的是系统的功能和数据两个方面，针对系统的功能需要的提问是系统要做什么，什么时候做，有哪些种操作模式，需要完成何种计算，或者数据格式的转换，对外部刺激的响应行为是什么样的，而针对数据的提问，则主要是围绕输入输出数据的格式，以及数据是否需要持久保存两方面。

（13）设计约束主要关注问题

设计约束主要关注的是系统的物理环境和接口上下文，而过程约束则是关注系统的用户和过程两个方面，引出设计约束的问题，围绕物理环境主要针对设备放在哪儿，节点数量的多少，是否对物理环境，有额外的特殊要求，比如温度、湿度和电磁干扰方面，是否需要更多的加固和保护，系统的规模有什么限制，同时在线用户数是否有要求等。

电源、供热和空调有没有什么限制，是否重用了系统的构建，从而对编程语言，有了特殊的要求，针对接口的提问，主要是针对输入、输出数据，是否有预定义的格式，它们的来源是来自一个还是来自多个，其他系统、哪些系统针对用户的提问主要关注的是谁使用系统,有哪几类用户,对用户的计算机技能及相关技术水平有何要求，对过程的提问则主要围绕，资源、材料、人员及其技能，文档的要求是纸质还是电子版是在线还是本地，本当读者有哪些是否需要依从一些固有的标准来准备文档等。

（14）引出质量需求的问题

引出质量需求的问题主要围绕几个核心的非功能性的属性来完成，包括：系统的性能 可靠性、安全性、可用性、可维护性、精度与精确性、交付时间及成本开销等。

3.3.4 需求过程

在需求工程过程中，最重要的工程活动包括，需求抽取、需求分析、需求规约、需求管理和需求验证。

（1）需求抽取

需求抽取的工作，主要目标是为了主动与干系人协同工作，找出他们的需求，识别潜在的冲突，磋商解决矛盾的冲突，定义系统的范围和边界。

这项活动的实质，就是要了解待解决的问题和它所属的应用领域，我们要确保该问题的解决，是有商业价值的。

如图 3.3 所示的漫画我们可以看到,需求抽取的过程就很像,这个漫画中小男孩，向他妈妈提出的四个问题，妈妈要他打扫房间，他问的第一个问题就是我为什么要打扫房间，这是对需求的顶层目标的一个提问，其次，他又问当前打扫房间的流程是什么样子的，这是对当前业务流程的一个抽取过程。之后，他又问难道做作业，不比打扫房间更重要吗？这是在跟他妈妈征求,两条需求之间哪个优先级更高的问题。最后，他问妈妈的是我怎么知道我的房间算是干净还是没干净,这是在问妈妈需求验收的一个标准。

图 3.3 需求抽取过程的举例

目前，有很多种需求抽取方法供大家使用，根据参与人员的能力、偏好，待了解需求的内容以及应用领域的特性，我们可以选择协同工作、面谈、问卷、观察、原型、文档分析、概念建模、角色扮演、检查列表等形式的抽取方法，可以多种方法混用。

如图 3.4 为需求获取方法应用情况调研结果。

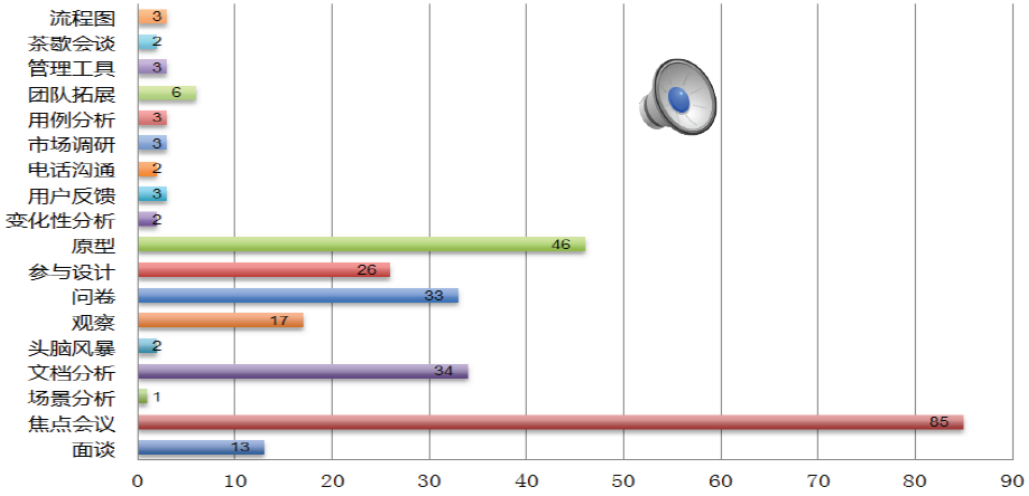


图 3.4 需求获取方法应用情况调研结果

根据我们之前对上百家中国的软件企业，所采用的需求获取方法的应用现状调研结果来看，其中，按照受欢迎程度排序，最受欢迎的几种方法包括：焦点会议的方法、原型法、文档分析法、问卷法、参与设计、被动观察等。

（2）需求分析

抽取需求之后，进入需求的分析阶段，在分析阶段，最主要的目标就是要对产品及其与环境的交互，进行更深入的分析，识别系统需求，设计软件体系结构，建立

需求与体系结构，组件之间的关联。在体系结构设计实现过程中，进一步识别矛盾冲突，并通过干系人之间的磋商、协调，解决这些矛盾。

需求分析的实质问题是要进行概念建模，选择常用的建模语言，并进行功能和信息的建模。需求分析的主要关键的一个问题，就是要实现体系结构设计与需求的适配。通过评估需求的满足度来评价体系结构设计的质量。

我们之前的调研结果反映，目前在工业界应用最广泛的，几种需求建模方法包括：非 UML 的图形建模模型、UML 模型、表结构、用户故事和自然语言文档。

概念建模方法的选择，首先取决于语言的表达能力，其次取决于语言的沟通效率和记忆水平，我们选择需求建模的语言时，必然选择一个学习成本相对较低，而且沟通效率较高的语言。

（3）需求验证

需求验证是对其它需求工程活动的质量保证，它通过数学的形式化工具或者工程化的测试过程，来确保系统满足干系人的要求，常见的需求验证方法包括：评审、原型、模型验证和确认测试等方法。

对刚刚提到的几种需求验证方法，从错误识别、时间开销和工作成本进行对比分析，我们可以发现，其中，原型验证和基于测试的验证，对错误的识别率较高。测试验证的工作成本较少，原型和测试验证，时间开销也相对较低，因此，这二者是首选。

（4）需求管理

需求管理贯穿从需求获取，到软件系统下线的全过程，它涉及软件配置管理，需求追踪 影响分析和版本控制等多个方面。需求追踪主要是描述和追踪，一条需求的来龙去脉，包括向前追踪到软件制品，向后追踪到需求的来源。建立需求的跟踪举证，是管理需求追踪关系的，一个有效的手段，变更请求管理，目前已经成为软件过程管理的，一个必不可少的环节，有系统化的工具和方法学的支持是现在实现广泛应用的，一个需求活动之一。

需求属性管理是对需求项目的细化的管理，其中最主要的属性包括：需求的优先级、需求的状态等。

3.3.5 需求来源

在需求的获取过程中，我们要考虑来自方方面面的限制和约束条件，由 Loucopoulos 和 Karakostas 提出的 40 件模型，为我们提供了一个非常有助于识别来自不同维度的问题的一个模型，该模型将问题领域分为四个世界，包括主题世界、应用世界、系统世界和开发世界。

通过对四个世界的分别的处理，我们能够有效地考虑到，来自四个方面的主要的待解决的问题。在主题世界中我们主要关注的是系统的关注的主题内容，比如，对一个银行信息系统来说，关注的是顾客、账号和各类交易，这是关于主题世界对象的一个探讨。

应用世界则是系统未来的，操作环境，在应用世界中我们要关注的，主要是人和企业的流程，关注经理、职员、顾客、存款、取款等活动，另外一个世界是系统世界，它是指系统在操作环境下的所有的内部操作，包括：系统要将交易纪录，保存在数据库中，要对指定的账号交易进行报告，给出账户的明细等。而开发世界关注的是软件系统的开发过程，项目的团队、人员、进度、安全，性能等质量要求，例如：系统要在 12 个月内，交付使用，这是开发世界要关注的问题。

有了四世界模型，我们可以很好地对问题的本质，以及关注的对象，进行分别

处理，项目开始的时候，我们如何着手开始获取需求，这里有几个主要的出发点，第一步是要确定干系人，然后定义系统的边界，找出系统的目标和情景，分析系统的可行性和风险。

干系人这一说法，来自英文单词 Stakeholders。Stakeholders 本身它的含义就是对一件事有决策权的人，干系人也不例外，他是对我们的软件项目，有话语权、有决策影响的人，找出所有的干系人，对于项目的成功有重要的意义。

我们可以通过四世界模型来分析，它隶属于哪个世界，从而推导出，它关注的是哪些问题。

用户关注的是新系统的功能和它的特征。

设计师关注的是构造正确完美的系统，并且尽量重用已有的代码。

系统分析师他想要获取的是，正确的需求。

培训与用户支持人员，他要确保系统的可用性和易管理。

业务分析师要确保的是我们要比竞争对手做的更好。

技术文档的作者是为系统准备用户手册和其它相关文档的人。

项目经理则希望按时、按预算、按目标完成项目

客户是为新系统买单的人。

从以上我们可以看出，干系人识别最主要的，就是要强调和用户之间的联络关系，要保证系统设计息息相关的人，他们的意愿能够体现到我们的需求中来。

需求定义过程中，客户与干系人的参与是至关重要的，不明确的需求定义，以及随意的需求变更，会产生以下问题。

首先是客户严重不满或产生纠纷，不切实际的估算与承诺，项目延期、超支、项目结束遥遥无期、项目团队精疲力竭等。

在需求过程中，常见的问题包括：产品设计目标不明确、干系人参与不足、干系人之间缺少共识、需求定义画蛇添足、需求快速变化、需求分析不足、需求管理不足等。

需求获取的内容，主要来自干系人、企业的业务过程、组织的规章制度、以及现有的系统。针对这几个来源，我们通过采用的获取手段是不一样的，干系人我们主要是通过沟通和交流的方法，业务过程主要是采用建模和分析的方法，组织规章制度主要是通过文本阅读、文档的处理等。

对现有系统我们可以参照它的用户手册、数据样本、界面描述、报告样本、屏幕截图等。是对现有参照物的一个分析和差异性的比较，干系人识别，是要找出任何和系统有关的人，有可能是资方、客户、系统用户、领域专家、项目研发团队，可见它来自四个世界的，任何一个可能的世界。识别干系人的时候，我们要提的问题包括，产品由谁来使用、输入谁来提供、输出谁要用到、谁监管、影响谁、奖励谁、惩罚谁，逐一召集干系人，或者一组干系人，一起定义系统细节的时候，我们达到了以下三个目的。

首先，我们知道从用户或者客户的角度来看，什么是一个可接受的，高质量的系统，其次，我们可以通过这件事，提高用户的满意度，通过用户和客户的参与，达到了系统教育和培训的目的。

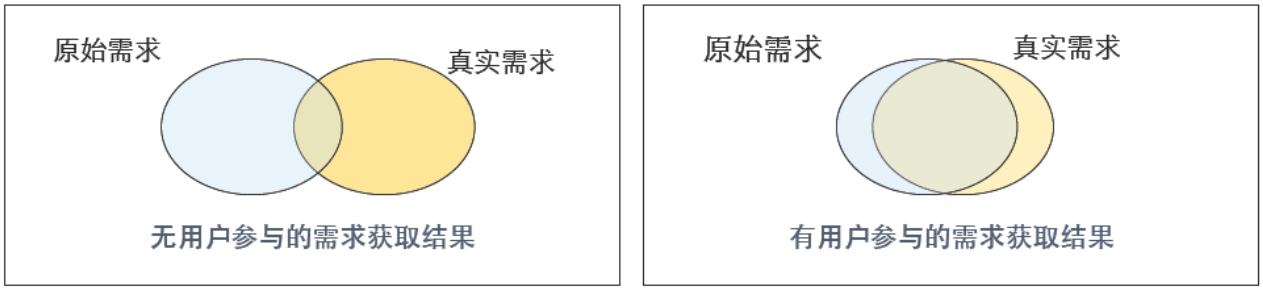


图 3.5 用户参与需求获取的示意图

如图 3.5 所示的两个示意图，我们可以看出，没有用户参与的需求获取和有用户参与的需求获取结果，原始需求和真实需求之间，它的差异就会变得明显的小。

总之，干系人的参与是项目成功的必要因素，他们的参与丰富了我们对系统信息和内容的获取，提升了所获取到的系统信息的质量，提升了系统的生产率，同时加深他们和我们自身对系统的理解，促进我们和客户达成共识，使得客户更希望系统成功，增强成功信心，共同确定了系统的边界，促进了需求质量的提升，提高了团队的整体性。

需求获取的另一个来源，是业务过程，它是对现有业务过程的分析，这样有助于识别业务问题，并加以改进，主要内容包括，首先找出并列举当前业务过程中存在的问题，然后分析问题的本质，是由于遗漏还是不好用，还是有新的需求，之后分析改进的机会和空间，并给出改进的实质，是在完成自动化过程还是在优化和改进现有的流程。

需求的另一个来源，来自组织规章与制度。通常的组织规章制度，定义的是当前该领域的最佳实践，分析规章制度有益于确定业务规则和约束条件，其中业务规则描述的是对业务过程的要求，如系统支撑的业务过程的结构、控制流程、预期效果等。

约束则是对系统开发过程的管理限制，涉及经济、政治、技术、环境四个方面，包括项目资源、时间、目标环境及系统本身，组织规章中往往还涉及过程自动化、工作流、关系、交互等相关方面的内容。

总之，组织规章制度，为我们提供了一个客观的，可参照的依据，分析现有系统有助于了解，未来系统的工作数据，包括数据对象、数据关系、数据库结构与系统结构以及系统报告等。现有系统与未来系统，共享工作环境，共享数据对象，因此，从对现有数据的了解，能够构想未来系统的雏形，并且找出差异的需求，从而更好地对未来系统进行规划。

3.3.6 需求获取技术

Steve McConnell 说过，需求抽取过程中，最困难的不是记录用户需求，而是与用户不断地探讨磋商，发现真正要解决的问题，确定适用的方案。

3.3.6.1 获取需求方法

这里我们列出了几种，最常用的需求获取技术，并给出了它们的适用场景，比如，面谈更适合在一样的时间，一样的地点，由少量人参与，由分析师驱动的获取模式。

问卷调查则是针对不同时间、不同地点，但是有广泛的大量的人参与，由分析师观察的获取模式。

群体诱导技术适合在相同或不同的地点，但在同样的时间，由 20 人左右参与，

由分析师参与的，群体诱导的活动。

参与调查则是在同样的时间，同样的地点，由分析师参与，由被调查者驱动的需求获取活动。

除了这四种需求获取技术以外，其它常见的获取方法还包括，文档分析、头脑风暴、情景分析、原型化方法、建模方法、需求讨论会等。

（1）面谈

面谈说到底就是问问题听答案，那么什么时候安排面谈，这样的抽取方法比较合适，首先就是我们可以见到干系人，其次就是很少的干系人，了解很多内容的时候，而且是干系人是真正的，领域专家的时候，而且干系人不能被聚到一起，进行群体诱导的时候，我们也不需要干系人彼此之间，进行交互，来得到最终解答，那这样的情况，我们就是非常适合安排，用面谈的方法来抽取需求。

面谈按其形式可以分为：有组织面谈和自由面谈两类。

有组织的面谈，通常要事先制订议程，问题可以是开放性的。

自由面谈则没有既定的议程，可以是面谈者和访谈对象，在现场临时安排即兴组织。面谈的优点是，通过面谈可以获得到，大量丰富的数据，有助于发现新观点，了解个人感受，发现目标和事实等。另外一个优点是，面谈的过程中可以深入探讨，根据面谈前一个阶段，获知的内容，动态地调整和补充后续的问题

面谈的缺点就是，面谈获取到的数据，大都是定性的，比较难以分析，而且也难于对多个问题的回答者进行横向比较。面谈的技巧因人而异，也不是很容易掌握。

在面谈过程中我们要注意的是，不要去问无法回答的问题，比如如何系鞋带，基于隐含知识的问题，以及脱离了环境和上下文的问题，另外，面谈者的态度有可能会导致回答者产生一些有偏见的回答。

下面给大家介绍一些简单的面谈技巧，在面谈开始的时候，可以谈一些比较轻松，不太敏感的话题，比如天气、体育比赛、受访者的照片、办公的陈设等。在询问受访者是否同意录音后，将录音笔放在受访者面前，提醒他们可以随时关掉。先问比较容易回答的问题，比如关于个人信息，您在这里工作多久了等。根据对方提出的某些线索，继续提问，关注用户所说的那些能够暗示以现有的方法，可能是错误的这样的情况，你可以说，你能就此多谈谈吗?将自由发挥的开放性的问题，放在最后，比如你还有什么想补充的，您有没有相关的经验等。

在安排基于访谈的用户，需求获取过程的时候，我们要完成以下几个步骤：

首先明确访谈的目的，确定我们通过访谈希望获得，哪些方面的信息，什么情况下我们能够确定目的已经达到。

然而设计访谈提纲，界定访谈的目标人群，邀请访谈用户进行用户访谈，整理访谈数据。

刚刚我们说的几个技巧，主要是在进行访谈这个过程中，要轻松开场，注意抽丝、挖掘，注意把控整个访谈的走向，对访谈数据的处理，我们之后要进行反馈讨论，也要整理、规范化访谈的输出。

在访谈过程中，我们要注意避免问一组固定的问题，要优先关注用户行为背后的潜在的原因，避免让用户成为我们自己系统的设计师，避免讨论技术细节，避免诱导性的问题，鼓励用户讲故事，讲他个人的观点。

（2）问卷调查

基于问卷的调查方法，问卷调查首先是要，定义一组问题，面对广泛的涉众的时候，一般采用基于问卷的方法，问卷的结果收上来之后，采用统计学的分析方法，使问句调查显得更科学，应用问卷的场景通常是有大基数的受访者，而且我们有基于

良好定义的，特定的一组问题和答案，也能够验证，有限次面谈得出的结论是否正确。当我们需要，某一个特定的结果的时候，我们可以采用相应的问卷的设计，来获得我们对既定结果的输入的搜集。

问卷调查的优点，就是我们可以快速获得大量的反馈，可以远程在线执行，可以搜集关于态度、信念、特性的信息。它的缺点也非常明显，由于问卷简单的分类，导致问卷是往往是脱离上下文的，留给用户自由表达它需要的空间相对较小，更多是用选择题的方式。

问卷调查的注意事项包括，在选择样本的时候，要注意尽量减小偏见，自愿的问卷回答者有可能存在一定的偏见，另外，也要避免样本的规模太小，自由发挥的问题，也是难于分析的，对答案有诱导性的提问，也会导致问卷调查本身它的一定偏颇之处，问题设计的妥当性，以及问题的含糊性都会影响问卷调查的数据采集质量。

问卷调查也需要言行化的方法和测试，在设计好问卷以后，可以寻找几个受访对象，对问卷进行测试，看是否其效果是问卷设计者预期的效果。

（3）群体诱导技术

群体诱导技术在执行的时候，往往是在同一个会议地点中，聚集 3-20 个干系人，每个人都将自己的观点大声地说出来，这个时候群体的答案，往往都要比个体提供的方案更全面。那么在什么时候适合采用群体诱导技术，那就是当每个人都只有关于整体的部分知识的时候，而且是人们需要彼此交互来对答案进行优化的时候，也是当我们能够让这些人，在同一时间聚在一起的时候。

如果群体诱导技术我们要保持群体中，每个人的匿名性，我们可以采用一些工具，让他们彼此之间隔离进行，在不同的地方进行，也可以采用一些工具。

群体诱导技术，按其组织形式可以分为两类：一类是专题小组会议，一类是头脑风暴。在群体诱导的过程中，人们的交互比正式的面谈来得轻松自然，人们之间可以互相激发抛砖引玉，实体模型和故事版的应用使群体诱导技术变得更为有趣，成果也更明显。它的缺点是，分组的时候可能会让有些参与者不够适应，就出现了少数服从多数带来的偏见和风险，对技术问题的探讨，往往只是表面化的回答，没有面谈来得深入。群体诱导的成功执行，需要训练有素的协调人引导者。

在群体诱导的过程中，需要注意的几种情况，一就是样本选择的偏见，还有在群体诱导过程中，可能出现统治与屈从的情况，当群体诱导活动，由少数的几个人控制的时候，引导者需要站出来，为每个人分配 5 分钟的规定时长，来陈述他们的观点，以确保每个人都有说话的机会，当发现有些人参与度不够的时候，引导者要站出来给每个人发好主意卷，作为一种激励机制。当出现无理取闹的参与者的时候，要发放恶意中伤卷，惩罚那些发言不够严肃的人。

（4）需求研讨会

需求研讨会是介绍项目的成员和干系人，收集需求列表，在会议的过程中运用头脑风暴、故事版、角色扮演以及现有系统评估等方法获取需求。

它的指导原则是，由主持人组织研讨，给每个人发言的机会，保持研讨话题的相关性，定义每条需求的属性，记录新需求的发现，并在最后总结所获得的需求，并就需求给出大家的决策和结论性的意见。

会议是一种重要的需求获取的途径和方法，可以用于总结、获取成果、取得用户反馈，在每个阶段结束的时候，与干系人安排会议，讨论信息搜集阶段取得的成果，对需求作出结论性的意见。并为某项设计，取得大家的共识，通过会议，我们可以确认所获得的信息讨论新的发现，同时，它也是一种重要的管理手段，用于推进项目开发的进展。

会议的执行，需要首先确定会议执行的目标，要对会议进行仔细的筹划，确保会议是有效的，是有目的的。

以需求获取为目的的会议，主要可能存在以下目标，首先是对需求内容的陈述、介绍，解决遇到的问题，协调矛盾的需求，分析项目进展，搜集和汇总数据，用户培训，对下一轮进行规划。在会议的执行之前和过程中，需要进行仔细的规划安排，首先在会前，要进行日程和设施的安排，确定日程之后，要提前通知给所有与会人。根据会议的目的，决定会议的组织形式，在会议的执行期间，要很好地控制会议进度和程序进展，事后要对会议进行书面总结，给与会者发放会议记录，正式的陈述报告、项目预演和头脑风暴，都有一定的执行方式，供大家参考。

（5）头脑风暴

头脑风暴的目标是要通过群组效应，激发大家对新产品、新系统的新想法，它在需求不完全明确的情况下比较有用。

进行头脑风暴的指导性方针包括：要采用有组织的，研讨会的形式，做到大家百花齐放，不评价、不争论、不批评、不受现实的可行性的限制，新观点的采集和产生多多益善，大家彼此之间抛砖引玉，互相启发，其目的就是为了，获取尽可能多的新观点，是一个发散的过程。

（6）参与观察法

参与观察法是让分析师深入到，干系人的日常工作环境中来，从侧面观察干系人的行为。在这个过程中，分析师的决策越被动，观测的效果越好。

那我们值得注意的一点是，分析师从旁观察可能已经影响到了干系人的行为，也就是说观察到的结果，可能和干系人的日常行为是有一些差异的。观察参与者的时候，要进行相应的动作研究。

那么什么时候采用观察法，那就是当有人或者事物，需要通过观察才能获得相关信息的时候，也是当干系人的知识无以言表，是行为指使的时候，采用观察法是行之有效的。

参与观察法是一种面向纵深的调查方法，由于观察者参与到被观察对象的日常活动中，成为组织中的一员，因此，带来了他的上下文相关的。

其优点，能够发现其它方法，无法发现的一些细节，它的缺点是时间成本，和代价比较大，而且获取到的，过于丰富的细节知识难于分析，难于将无关细节驱除，也无法就改进建议，所带来的结果给出更多的评价。

这里值得注意的是，观察者往往由于参与到环境中，过于深入，有被同化的危险，失去了客观的立场。

（7）亲身实践

亲身实践与参与观察有一定的相似性，但又不尽相同，参与观察的过程中，用户是从旁观看，而且亲身实践，则是通过观察、提问，甚至亲自操作，来更精确地了解工作内容，它建议实时实地的开展实践工作，并获得干系人的及时的反馈。

亲身实践的适用场景是，当用户太忙无法安排专门的时间来参加面谈的时候，可以由工程师到用户的工作场景中去，亲自实践任务的流程。另外一种情况就是人们往往只是在既定的流程中，完成相应的工作，并没有意识到每天的工作实际上是在做什么，也就是说，他是下意识地完成这些任务的时候，通过亲身实践，能够更好地恢复出，工作的流程和步骤。

实践者通过反复的观察干系人，执行该动作的过程，并且学习任务的技能，反过来做给用户看，让用户确认他所做的步骤是否正确，从而获得第一手的实践知识和任务的相关描述，这也有利于与用户和顾客，建立密切的沟通联系，为后续的工作奠

定很好的交流的基础。

（8）文档分析

文档分析是一个非常好的入手点，它的目标是为了理解待解决的问题，可以作为下一步，做业务流程建模和访谈的前期基础，进行文档分析的指导方针包括：要确定当前的业务流程，或者产品的优缺点，找出产品或信息中可以重用的部分，从多个来源抽取用户的需求。

（9）仿真原型

仿真原型是用户最偏爱的需求获取手段之一，它的目标是为了明确那些含糊不确定的需求，简化需求的文档，确认需求，能够尽早地获得用户和客户的反馈，它的适用场景和指导方针是原型主要用于需求确认，能够提供需求评估的数据基础，尤其适合评估不同的用户界面设计方案，帮助用户可视化关键的功能点，是一种直观、有效的沟通交流工具。

（10）情景分析

基于情景的分析方法是课程要重点介绍的方法之一，情景分析的主要目标是要清楚地定义，有用户参与完成的业务实践的步骤，获取对用户来说可见的系统动作，与其它方法相比，它有着得天独厚的优势，因为对用户来说，它几乎不需培训，用户只要直接写出它期望的与系统交互的流程，就已经是情景分析最好的素材。

情景分析的适用场景和指导原则是，它要确定与用户间可能的交互活动，具体包括 定义业务事件，以及感兴趣的领域性质。我们将业务事件细化为具体的业务活动和业务过程，并将其描述出来，形成情景分析的结果

（11）概念建模

概念建模是我们课程要重点探讨的另外一项研究内容，概念建模的目标就是要图形化的手段描述现实世界的问题。建立未来系统的模型，对未来系统进行抽象的表示，在建模过程中，对原始需求进行评估和扩展，得到清楚、完整、正确、一致的需求描述。它的指导原则主要是对系统进行不同角度的抽象，建立不同类型的模型，例如：对系统分而治之，对子系统的结构，进行描述的分解模型，对数据进行描述的 E-R 模型，以及面向对象模型，以描述行为为中心的用例模型、过程模型、活动模型等。

（12）竞争性需求分析

我们在软件产品的设计过程中，其实是有两种创新的程度，一个是演化型的产品的设计，还有一类是全新产品的设计，那么在新产品的设计过程中，我们首先要对它进行系统的分析和评估，这里我们给出一种基于竞争性需求分析的框架，该框架为我们提供了五个思维基点，让我们对新的产品的策划进行可行性的评估。

首先我们要了解的是我们的新的产品的创意是不是解决了用户的需求，解决的用户的哪方面的需求，这个明确需求的过程，就是我们获得市场上用户认可的一个基础

另外一个方面，就是在方法的方面，我们找到了需求以后，下一步该怎么办，我们是不是有独特的方法来解决用户的问题。这些新鲜的方法做法，不仅仅是技术上的，也可以是商业模式上的，是关于这个产品运行的地域、使用人、行业或者是成本方面，只要是在任何一个方面，我们有了独特之处，那么，该产品就是有可能屹立在市场上不被打败的。

此外我们还要评估新产品可能给用户带来的收益，这个时候我们既有了独特的做法，我们又有了用户的需求，那么我们就要了解这个产品给用户带来的是什么真正的好处，如果它已经有了现有的解决方案的话，那我们用什么方法能够让它离开现有

的产品，来使用新设计的产品，这是一个关于用户迁移成本和用户的代价的一个权衡利弊的过程。

其次我们还要评估这个产品的竞争力有多大，市场有多大，有多少竞争者在瓜分这个市场，我们的新产品，如果不是最先进入市场的，那么有没有取得成功的可能性。

这里就涉及到先发优势和后发优势这两个方面，所以竞争性的需求分析，最核心的问题就在于看清新产品的优势在哪里，劣势在哪里。

最后一项就是产品研发完成后，我们如何去交付和推广。交付和推广，实际上在新品的成功中，也扮演着重要的角色，一个好的产品，无法推广到用户的面前，那么它也是很难得到，广泛的应用获得市场的成功的。

（13）A/B 测试的方法

许多互联网公司都在使用 A/B 测试的方法，研究人员也用这种方法进行研究，A/B 测试看起来很简单，如果你的产品已经有用户在用，但是你想对用户界面，做一些改进又不知道用户是否会欢迎这样的改进的时候，我们就可以配置 A/B 测试环境。

首先，选定要实验的是哪两种不同的 UI，确定衡量界面的标准，确定数据搜集的流程，确定试验运行的时间和人数，通常我们选择 5%-10%的用户。

然后通过技术实现 A/B 测试环境的搭建，之后搜集用户的行为数据，分析数据，最后得出我们要得到的实验结论

在 A/B 测试过程中，我们最主要的是要获得系统和用户之间的交互的反馈，比如：有 A、B 两种方案中，我们最终得到 B 方案获得了更多用户的正面反馈，因此，在这个测试的过程中，我们得到的结论是，B 方案优于 A 方案。

互联网平台，目前成为软件运行的基础平台，这使得用户行为数据的采集成为可能，许多软件提供商都在线的将用户的访问数据，进行分析和统计，由此得出对下一步新产品设计的启示。

能够访问到的用户的数据，包括：交互过程数据、眼动数据、页面访问时间、页面访问内容、用户的偏好等。

基于平时的日常交互数据的采集结果，我们可以对操作数据进行统计分析，包括：用户所偏爱的厂商、版本、用户访问的 IP 地址和所来自的地域、产品的日访问量、访问频次、用户的身份、平均访问时长以及用户的打分等。

通过对这些数据的分析，我们能够得出一些关于用户偏好，以及产品现在表现的基本的数据证据，由此推断出下一步对产品的改进方面。

3.3.6.2 获取需求技术选择

我们介绍了众多的需求抽取技术和方法，在选用这些抽取技术和方法的时候，我们要根据获取的需求的具体内容，来有效地选择。

例如：获取界面需求的时候，原型法、情景法和建模法是比较有效的。

在获取业务逻辑的时候，采用观察法、亲身实践、面谈和情景的方法，则有利于业务累计的精确理解和描述。

对未来系统的信息和数据结构，进行获取的时候，面谈与现有系统的分析是最行之有效的，要建立完整精确的需求模型很困难，建立完整的原型也不是很现实，因此，我们要根据需要和客观条件，灵活地搭配组织运用，形成针对一个具体问题的最有效的获取方案。获取需求技术选择如图 3.6 所示。

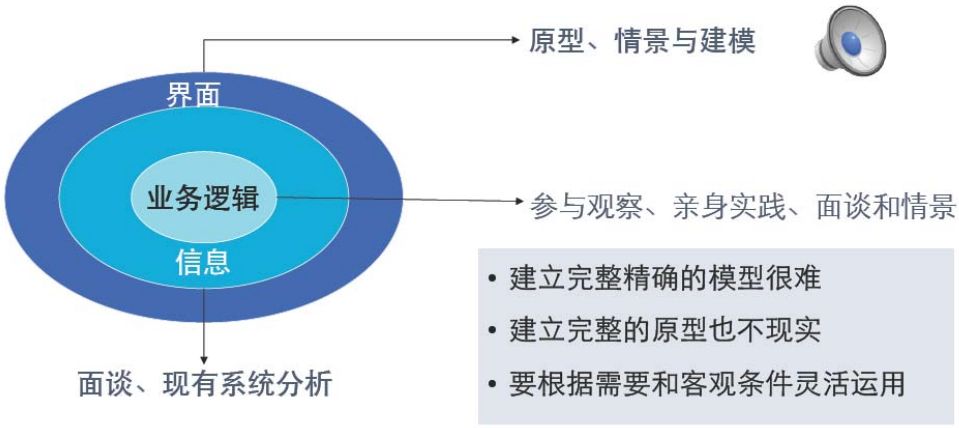


图 3.6 获取需求技术选择

3.3.6.3 获取需求注意事项

用户需求获取是一门倾听的艺术，在需求获取中，我们更要关注的是引导干系人让他们更好地更多地提供有价值的信息。需求分析师的作用就是协助、鼓励干系人表述他们的需求。

由 Alan Davis 教授提出的 Just enough requirements，management，也就是足够好的需求管理，他提到什么是足够好的需求管理，足够好就是要让你完成用户的预期的功能，让用户满意，但是又没有在需求管理过程中，消耗掉所有的项目时间和精力，使得没有足够的时间和精力去做开发，正如足够多的人寿保险，买的保险足够多，使得我们晚上可以睡得安心，不会担心我们所关心的人会得不到足够的照顾。但我们还没有买太多的保险，使得我们会担心明天支付不上保险的预付金。

假如一个客户来找你，跟你说我的电梯太慢了，你是直接跟他说，我不这么认为，我觉得你的电梯有吞吐量问题，而不是速度问题，还是静下来耐心听他说，好的，告诉我为什么你觉得它们是慢的。这两种截然不同的态度，就代表了面对一个用户的时候正确的和错误的态度。在和客户沟通交流的时候，不要尝试向干系人证明，你是更聪明的，要抓住所有的机会，表现出你认为干系人是聪明的，这样他们才更愿意跟你分享他们认为对的事情，以及他们真正的痛点是什么。

实践经验表明，很大一部分需求沟通的错误都是由术语表达存在歧义造成的，确定术语表的时候，我们要针对关键的术语询问，该术语的含义是什么，并把所有已经取得了共识的术语的定义，明确地写出来，这样为后续的阅读者和参照提供依据。

在我所参加的一个海图研发的项目中，对于航行计划这个专业的术语，有三位专家他们的定义都是各不相同的，这导致我们在项目的沟通过程中，花费了一个小时的时间，来明确这个术语的含义，并给出后续的围绕这个术语的软件功能的定义。

在需求获取过程中，只采用一种抽取技术是不够的，技术的选择和项目的参与人相关，与待理解的需求相关，与具体的应用领域相关。

3.4 讨论题

1、软件需求获取方面问题。
如何理解软件需求获取有哪些方法？获取需求注意哪些方面？
对需求的分类和来源进行讨论。

2、软件需求开发过程如何？

3、如何理解软件需求是软件项目成败的关键？

以“盲人摸象”为例说明什么是软件需求？怎样的需求才能被认为是“需求”呢？你是怎样理解软件需求的多样性的？请你就这个问题与大家一起分享你的观点。

这一例子说明：不同的人对不同的事物有不同的见解。每个人都有自己的想法，所以需求是多样的，但是并不是所有需求都会被采纳，应该是多数人都有的需求，才会被解决。

需求定义了系统必须具备的能力，这种能力体现了用户的需要和开发者对用户需要的理解。

例子：“盲人摸象”的故事

古时印度有一个小国，国王名叫镜面王。他信奉释迦牟尼的佛教，每天都拜佛诵经，十分虔诚。可是，国内当时流行着很多神教巫道，多数臣民被它们的说教所迷惑，人心混乱，是非不明，很不利于国家的治理。镜面王很想让其臣民们都归依佛教，于是就想出了一个主意：用盲人摸象的现身说法教育诱导他们。镜面王吩咐侍臣说：“你找一些完全失明的盲人到王城来。”使者很快就凑集了一群盲人，带领他们来到王宫。

使者走进宫殿向镜面王禀报说：“大王，您吩咐找的盲人现已带到殿前。”镜面王说：“你明天一早带领盲人们到象苑去，让他们每人只能触摸大象身体的一个部位，然后马上带他们来王宫前广场。”

第二天上午，镜面王召集所有的大臣和数万平民聚集在王宫前的广场上，沸沸扬扬的人们交头接耳，谁也不知道国王将要宣布什么重大的事情。不一会，使者领着盲人们来到了镜面王的高座前，广场上的人们顿时安静了下来。镜面王向盲人们问道：“你们都摸到大象了吗？”盲人们齐声回答说：“我摸到大象了！”镜面王又说：“你们每个人都讲述一下大象是什么模样的！”摸到大象腿的盲人首先站出来说：“禀告圣明的国君，大象就像一只盛漆的大圆桶。”摸到大象尾巴的盲人说：“大王，大象应该像一把扫帚。”摸到大象腹部的盲人说：“大王，大象确实像大鼓。”随后，摸到大象头部的说大象像大勺子，摸到大象牙的说大象像牛角，摸到大象尾巴后部的说大象像棍杖，摸到大象耳朵的则说大象犹如簸箕。最后，摸到大象鼻子的盲人说：“圣明的大王，大象实在像一根粗绳索。”一群盲人分成了几伙，吵吵嚷嚷，争论不休，都说自己正确而别人说的不对。他们又纷纷到镜面王前争辩说：“大王！大象的模样确实像我说的那样！”这时，在场的臣民见此都大笑不止，镜面王也意味深长地看着众人笑了起来。

3.5 测试题

单选题 （10 满分）

1.下列哪项需求描述属于业务需求描述？

A.“我们的任务是无缝集成有竞争力的软件信息服务来解决商业问题”

B.“我们的目标是让客户将我们的品牌和高质量联系在一起”

C.“我们公司的主营业务是销售飞机票”

D.“公司网站上销售的产品必须满足所有食品药品监管需求”

2. 下面哪项是百货店收银系统的非功能性需求？
   A. “提供新鲜的蔬菜和水果”
   B. “买 10 个或 10 个以下商品的客户可以走特殊通道”
   C. “设有存包处”
   D. “为雇员发工资”

3. 以下哪种需求获取方法最适用于身处多个不同地点的人在各自方便的时间参与，围绕同一个主题表达自己的观点？
   A. “问卷调查”
   B. “面谈”
   C. “群体诱导”
   D. “文档分析”

4. 在一个列车控制软件的需求文档中，我们发现了以下两条需求描述：“列车车门在两个停靠站之间要保持关闭”；“列车发生紧急停车时，要打开车门”。这里出现的需求问题是？
   A. “无法测试的需求”
   B. “不完整的需求”
   C. “含糊的需求”
   D. “矛盾与不一致的需求”

5. 获取软件系统需求不包括以下的哪个来源？
   A. “系统相关领域的法律法规”
   B. “系统的质量控制团队”
   C. “系统的业务流程描述”
   D. “其他类似系统产品”

6. 软件需求工程师的职责不包括以下的哪一项？
   A. “撰写需求规格说明书”
   B. “与用户持续沟通，了解用户对产品的期望”
   C. “控制项目的风险”
   D. “对需求的优先级进行排序”

7. 在选择软件需求获取的技术的时候，以下哪种策略最优？
   A. “考虑尚不了解的那部分需求的特点”
   B. “考虑需求工程师本身对各种获取技术的驾驭能力”
   C. “考虑目前系统所属的行业及应用领域的现状”
   D. “综合考虑上述因素”

8. 以下哪种非功能性需求是面向软件过程的？
   A. 安全性
   B. 可靠性
   C. 可维护性
   D. 用户友好性

9. 以下哪种需求获取方法是面向创新型产品的？
   A. 竞争性需求分析
   B. A/B 测试

C. 用户行为数据采集
D. 可用性分析

10. 对工业界实践情况的调研结果表明，以下哪种需求获取技术在实践中的运用最为广泛？
A. 需求调研会议
B. 系统原型法
C. 文档分析法
D. 参与设计法

第 4 部分 软件体系结构

4.1 本章的教学目标

通过本章的学习，使学生初识软件体系结构的概念和设计原则，掌握软件体系结构风格、案例研究、软件体系结构描述、特定领域的软件体系结构和流行的软件体系结构等。

软件体系结构是软件设计过程中的一个层次，这一层次超越计算过程中的算法设计和数据结构设计。体系结构问题包括总体组织和全局控制、通讯协议、同步、数据存取，给设计元素分配特定功能，设计元素的组织，规模和性能，在各设计方案间进行选择等。

软件体系结构的模型分为：结构模型、框架模型、动态模型、过程模型和功能模型，最常用的是结构模型和动态模型。

4.2 本章的教学方法

本章的教学方法采用“翻转课堂式”教学模式，学生可以通过互联网使用“学堂在线”网上优质教育资源，选择由清华大学刘强副教授主讲的“软件工程”课程的“第 12 章软件体系结构” 课外观看学习，网址：http://www.xuetangx.com/courses/course-v1:TsinghuaX+34100325X+sp/about ，通过视频中教师的讲解，回到课堂上教学交流讨论，并相关的完成习题。

通过理论教学、课堂讨论、适当的布置课外习题作业来完成学习。

4.3 本章的教学内容

软件设计是软件工程生命周期的重要活动，它在软件需求的基础上，进一步产生软件内部结构的描述以作为软件构造的基础。

软件体系结构作为从软件设计抽象出来的学科，目前已经成为软件工程一个重要研究领域。软件设计包含总体（概要）设计和详细设计两个大部分，其中概要设计包括功能（模块）设计、数据概念结构设计、输入输出（接口）设计等；详细设计包含用户界面设计、数据库模式设计、数据结构设计等。

本章主要介绍软件体系结构的概念、作用和发展历史，阐述软件设计原则，并解释体系结构风格、设计模式和框架等。

4.3.1 软件体系结构概念

随着软件技术和应用的不断发展，软件系统的规模越来越大，其复杂度也越来越高。在这种情况下，代码级别的软件复用已经远远不能满足大型软件开发的要求。面对日益复杂的软件系统的设计与构造，软件工程师需要考虑的关键问题是对整个系统的结构和行为进行抽象，一方面是寻求更好的方法，使系统的整体设计更容易理解，另一方面，是寻求构造大型复杂软件系统的有效方法，实现系统级的复用。

我们在前面，做过一个性能分析的实验，这个实验要求编写一个程序，用于统计一个文本文件中，每个英文单词出现的频率，并输出词频最高的 100 个单词。大家在编写这个程序的时候，重点会考虑哪些部分的设计。对于这个程序的设计，我们主

要是考虑，选择合适的分词方法和查询算法，建立可以实现快速查找的数据结构，还会涉及到外部存储文件或数据库的构造。

在代码构造时，可以调用已有的 Python 代码库，也就是说，编写小规模的程序，主要关注的是算法的选择、数据结构的设计和数据库的构造。

如果是开发一个 Web 信息检索系统，又会关注哪些方面。

这个系统要对发布在 Web 上的信息资源进行搜集、整理和组织，形成一个信息资源索引库，并通过检索界面，把最符合用户要求的网站或者是网页提供给用户。

显然，这个系统比前面的词频统计程序要复杂得多，无法简单地用数据结构和算法来进行描述和设计。

我们现在对整个系统的结构和行为进行抽象，一般网页信息检索的过程大概是这样，系统先要从网上抓取网页，然后对网页信息进行处理，建立索引数据库，在用户进行查询的时候，根据输入的查询关键字，在索引数据库中进行搜索，并对搜索结果进行处理和排序，最后返回一系列与用户查询相关的网页信息。

这样整个系统，就可以抽象成三个部分：网页搜集、预处理和检索服务。

对应上面的过程，整个系统可以划分成三个功能模块：信息采集模块实现网页搜集功能。标引处理模块实现预处理功能。信息检索模块实现检索服务的功能。

除了模块分解之外，我们还要考虑使用什么策略，来实现非功能需求的目标。比如说，可以采用分布式的文件系统和计算框架，提高大规模网页搜集和检索的效率等等。

从上面的事例可以看出，对于大规模的复杂系统来说，对系统的全局结构设计和规划，变得比算法的选择和数据结构的设计，明显要重要得多。

在工程领域，体系结构设计被广泛地应用于，处理复杂系统的设计问题，比如说大型的客机，在整体上被划分成：机翼、机身、尾翼、起落装置和动力装置等不同的组成部分，它的总体结构也要根据技术的要求和使用环境，来确定飞机的外形、总体布局、发动机选型等。机体的设计还要满足强度、刚度、疲劳、损伤容限、一些安全性的要求。

体系结构这个词起源于建筑学，它是从系统的宏观层面上，描绘出整个建筑的结构，把一个复杂的建筑体拆分成一些基本的建筑模块，并通过这些基本的模块进行有机地组合，形成整个建筑。建筑的体系结构设计，要满足坚固、适用、赏心悦目的要求，具体包括以下方面，第一，有哪些基本的建筑单元，比如说，砖、瓦、梁、柱、屋顶、外墙、门窗等，第二，如何把这些基本单元进行组合，形成整体的建筑，第三，建筑单元怎么搭配才比较合理，比如说外形是否美观，符合承重和受力的要求等，第四，不同类型的建筑有什么典型的结构，比如说，欧洲古典风格和中国园林风格，就是两种不同的建筑类型，第五，如何快速、节省地进行建造和施工，第六，对于建造完成的建筑，怎么样才能适当地修改，第七，怎么才能保证单元的修改，不会影响整个建筑的质量。

软件体系结构已经在软件工程领域获得了广泛应用，它包括构成系统的设计元素的描述，设计元素之间的交互，设计元素的组合模式，以及在这些模式中的约束。

软件体系结构主要包含五个方面的内容：第一是构件，它代表着一组基本的构成要素，第二是连接件，也就是这些要素之间的连接关系，第三是约束，它是作用于这些要素或者连接关系上的一些限制条件，第四是质量，它是系统的质量属性，像性能、可扩展性、可修改性、可重用性、安全性等，第五是物理分布，它代表着这些要素连接之后形成的拓扑结构，描述了软件到硬件的影射。

简单地说，软件体系结构由构件+连接件+约束组成，它提供了在更宏观的结构

层次上，来理解系统层面问题的一个骨架，它主要关注于，把一个复杂系统从整体到部分的一个最高层次的划分，对于设计模块的一个功能定义，以及如何把这些模块组合成为一个完整的系统。

系统的分解要满足设计目标的要求，反映开发中具有重要影响的设计决策，从而保证系统的质量要求。

在软件体系结构的概念中，构件是具有某种功能的可复用的软件结构单元，它们表示系统中，主要的计算元素和数据存储，任何在系统运行中，承担一定功能，或者发挥一定作用的软件体都可以看作是构件，像函数、模块、对象、类、文件、相关功能的集合等。

构件作为一个封装的实体只能通过它的接口和外部环境进行交互，其内部的结构是被隐藏起来的，构件的功能以服务的形式体现出来，并通过接口向外发布，进而产生与其他构件之间的关联，连接是构件之间建立和维护行为关联和信息传递的一个途径，它需要两个方面的支持，一个是连接发生和维持的机制，另一个是连接的协议。

一般来说，基本的连接机制，包括过程调用、中断、I/O、事件、进程、线程、共享、同步、并发、消息、远程调用、动态连接、API 等，协议是连接的规约，对于过程调用来说，可以是参数的个数和类型、参数排列次序等。

对于消息传送来说，可以是消息的格式，除了连接实现的难易程度之外，同步或者异步是影响连接实现的复杂因素之一。

连接件表示构件之间的交互，并实现构件之间的连接，比如说，管道、过程调用、事件广播、客户机-服务器、数据库连接等都是属于连接件。

连接件也可以看作是一类特殊的构件，它和一般的构件的区别在于，一般构件是软件功能设计和实现的承载体，而连接件是负责完成构件之间信息交换和行为联系的专用构件。

好的开始是成功的一半，初期的总体设计是决定软件产品成败的，一个关键因素往往错误的设计决策，会造成灾难性的后果。

在软件设计过程中，我们可以对一些经过实现证明的体系结构进行复用，从而提高设计的效率和可靠性，降低设计复杂度，另外，我们也可以把一些公共部分抽象提取出来，形成公共类和工具类，为大规模开发提供基础和规范。

体系结构的设计要具备灵活性，在产品开发的演化过程中，可以很方便地增加新的功能，更好的适应用户需求的变化，由于体系结构构建了一个相对小的简单的模型，这样就把复杂的问题简单化，使系统更加易于理解和实现。

软件体系结构突出体现了早期的设计决策，展现出系统满足需求的能力，软件系统规模在迅速增大的同时，软件开发方法也经历了一系列变化。

软件体系结构也从最初的一个模糊概念，发展成一个日趋成熟的技术，从 20 世纪 70 年代开始，出现了面向过程的开发方法，系统被划分成不同的子程序，子程序之间的调用关系就构成了系统的结构，软件结构成为开发的一个明确概念。

到了 80 年代，面向对象方法逐渐兴起和成熟，系统被划分成不同的对象，并采用统一建模语言，来描述系统的结构。

90 年代开始，软件开发强调构件化技术和体系结构技术，体系结构也成为软件工程领域的一个研究热点，出现了体系结构风格、框架和设计模式这样一些概念。

2000 年之后，面向服务的体系结构成为面向对象模型的替代模型，它具备分布式、跨平台、互操作和松散耦合等。这样一些特点致力于解决企业信息化过程中不断变化的需求和异构环境集成这样一些难题。

在这个发展过程中，软件系统的基本模块，它的粒度越来越大，系统的结构越

来越趋向分布和开放，所解决的问题也从技术本身转向商务的过程。

下面我们简单说明一下体系结构风格、设计模式和框架的概念，以及它们的区别。

体系结构风格，描述了某一个特定应用领域中系统组织的惯用模式，它反映了领域中，众多系统所共有的结构和语义特性，比如：MVC 就是一种常见的体系结构风格。

设计模式是描述了软件系统设计过程中常见问题的一些解决方案，一般是从大量的成功实践中总结出来的，而且被广泛公认的一些实践和知识。观察者模式是一种常用的设计模式，它主要用于解决事件处理的问题。

软件框架是由开发人员定制的，应用系统的骨架，它是整个或者部分系统的可重用的设计，一般由一组抽象的构件和构件实例之间的交互方式组成。

像 Django（Django 是一个卓越的新一代 Web 框架）就是一个开放源代码的外部应用框架，是由 python 写成的，包括：面向对象的映射器、基于正则表达式的 URL 分发器、视图系统和模板系统，这样一些核心的构件。

关于框架和体系结构的关系，体系结构是一种设计规约，而框架是一种具体实现，只不过它实现的是应用领域的共性部分，是领域中最终应用的模板。

体系结构的目的是指导软件系统的开发，而框架的目的是设计的复用，确定了框架之后，软件体系结构也随之确定。对于同一种软件体系结构，比如说 Web 开发中的 MVC，可以通过多种框架来实现，像前端有 ANGULAR，BACKBONE，后端有 python 的 django 等框架。

对于框架和设计模式而言，框架给出的是整个应用的体系结构，而设计模式给出的是单一设计问题的解决方案。

比如说一个网络游戏,可以基于网易的 Pomelo 框架开发,这是一个基于 Node.js 的高性能分布式游戏服务器框架。在实现某个动画功能时，可能会使用观察者设计模式来实现自动化的通知更新。

从这个例子可以看出，设计模式的目的是改善代码结构，提高程序的结构质量，框架强调的是设计的重用性和系统的可扩展性，它的目的是缩短开发周期，提高开发质量。

4.3.2 软件设计原则

设计原则是系统分解和模块设计的基本标准，应用这些原则可以使代码更加灵活，易于维护和扩展。虽然不同的程序设计语言可能有一些自己语言特点的原则，但是抽象、封装、模块化、层次化和复用这样一些原则，应该是所有语言都通用的。

下面我们分别介绍它们的含义，抽象是一种思考和解决问题的方法，它只是关注事物中和问题相关的部分，而忽略其他无关的部分。

现在看一个例子，假设一个应用程序，要连接 SQL Server 数据库，那么一种实现方法是应用程序，直接使用 SQLServerConnection 类，来实例化一个对象，然后在程序中调用它，但是如果我们需要把数据库，换成 MySQL，这个时候，程序中所有，SQLServerConnection，就要全部换成 MySQLConnection，显然数据库的更换，对应用程序的修改造成很大影响，现在换一种方法，我们提供一个通用的数据库连接接口 DBConnection，应用程序调用这个接口对数据库进行连接，不同的数据库各自具体实现这个接口，通过这种方式只要接口不变，数据库发生更改时，只会影响具体接口的实现，而不会影响上面的应用程序。

显然抽象是认识复杂事物过程中使用的一种思维工具，它可以起到降低复杂性

和增强扩展能力的作用，封装和信息隐藏是把一个软件单元的实现细节进行隐藏，然后通过外部可见的接口来描述它的特性，需要注意的是，单元接口应该设计的尽可能简单，并把单元对于环境的假设和要求降低到最小。

模块化的概念，在前面的课程中已经提到，它是把一个复杂的系统，分解成多个更小的模块，模块是可以组合、分解和更换的单元。

模块之间通过接口进行联系，这也是分而治之的思想，通过把一个问题分解成多个小的独立，而且相互作用的组件，来降低大型软件开发的复杂度。

对软件系统进行模块化分解的原则，就是高内聚、低耦合。

我们先简单介绍一下内聚和耦合的概念。

所谓内聚，是一个模块或子系统内部的依赖程度，它描述的是模块内的功能联系，高内聚的模块，应该只做而且做好一件事，如果一个模块或子系统的元素，都是彼此相关的，而且都执行类似的任务，那么它的内聚性就比较高。如果含有很多彼此不相关的元素，它的内聚就比较低。

所谓耦合，是指不同模块之间相互连接的程度，如果两个不同模块之间相对独立，或者有很少的连接依赖，那么其中一个发生变化的时候，对另一个产生的影响就很小，反过来，如果它们之间有很强的关联，那么一个发生变化，就可能对另一个产生很大的影响。

下面我们举两个例子，来说明这两个概念。

第一个例子，如图 4.1 所示是一个角色扮演游戏的部分类图。

图 4.1 一个角色扮演游戏的部分类图。

不同角色的人物，可以在一定的区域中移动，当双方相遇之后，会进行打斗。

在第一个方案中，我们把所有的类都放在一个 GameCE 子系统中，但是我们仔细分析一下，就会发现，游戏角色和布局环境这两个部分所实现的任务是完全不同的，而且它们的关联也不是非常紧密，根据高内聚的原则，我们可以把它们分成两部分，其中 Characters 只和游戏角色有关，Environment 只和布局环境有关，而且每一个部分的功能，都是单一职责的，显然第二个方案，内聚性更高。

第二个例子，在一个软件系统中有三个子系统，A、B、C，都要访问同一个关系数据库，这里给出的设计方案是三个子系统直接访问数据库，大家认为这个方案有什么问题吗？

在这个方案中，数据库和 A、B、C 三个子系统形成了高耦合的情况。当数据存储方式发生变化时，三个系统都会发生改变。为了降低这四个部分之间的耦合，我们在 A、B、C 三个子系统和数据库中间增加一个存储子系统 Storage，这样就屏蔽了底层数据库的变化对上层子系统产生的影响。当底层的存储机制发生变化的时候，只需要修改存储子系统。这样做之后，子系统分解的整体耦合度就降低了。

在系统被划分成若干模块之后，我们可以分别在一个模块的内部来处理那些数量已经大量减少的元素，但是为了确保系统的完整和一致，我们同时还必须在系统的

层面来处理这些模块之间的关系。

层次关系是一种常见的系统结构，一个系统的层次分解会产生层次的有序集合，这里的层是指一组提供相关服务的模块单元，通常是通过使用另一层的服务，来实现本层的功能。

层一定是有序组织的，每一层可以访问下面的层，但是不能访问上面的层，最底层是不依赖于任何层的，最顶层也不会被任何层来调用。

在一个封闭式的结构中，每一层只能访问和它相邻的下一层。

在一个开放式的结构中，每一层还可以访问，下面更低的其他层次，当然层的数目不宜过多，还有一种划分是把系统分解成，相互对等的若干模块。

比如说，第二层的 B、C、D 和第三层的 E、F、G，这些模块单元相对独立，每一个模块之间的依赖比较少。

总的来说，一个大的系统可以逐层分解，直到所分解的每一个模块，可以简单到一个开发人员可以独立实现为止，但是由于不同模块单元之间存在接口，所以每一个模块，都增加了处理的开销，过度的划分和分解，会增加额外的复杂性。

这是安卓操作系统层次结构的一个示例，它的最底层是 Linux 内核，它为了安卓提供了启动和管理硬件，以及安卓应用程序最基本的软件。

在内核之上是一系列共享程序库，为开发者和类似终端设备拥有者，提供了必要的核心功能，应用框架支持第三方开发者之间的交互，使他们能够通过抽象的方式，访问所开发的应用程序需要的一些关键资源。最上层是运行在虚拟机上的 JAVA 应用程序。

复用是利用一些已经开发的软件元素来生成新的软件系统，这样做可以提高开发效率和软件质量，软件由不同粒度的复用。

代码复用是一种最常见的形式，构件库中的源代码构件，具有更高的通用性和可靠性。像 Pyhton 社区中就有大量实用的构件库。

软件体系结构复用，是采用已有的软件体系结构，对系统进行设计，通常它支持更高层次，更大粒度的一个系统复用。

框架复用是对特定领域中，存在的一个公共体系结构及其构件进行复用，Python 就拥有大量的框架，比如：django、flask tornado 等。

设计模式是通过为对象协作提供思想和范例来强调方法的复用，应该说设计模式是一种设计思想，但是不同语言的特性会影响设计模式的实现，比如：C++语言实现设计模式是充分利用集成和虚函数这样的机制，但是 Python 语言提供了和 C++完全不同的对象模型，而且它有一些特殊的语法，比如：装饰器，这个本身就应用了设计模式，所以 Python 在运用和实现设计模式上，与其他的一些面向对象语言是不同的。

4.3.3 软件体系结构风格（一）

软件体系结构代表了系统宏观共性的东西。我们从建筑风格谈起，建筑风格等同于建筑体系结构的一种可分类的模式。

它一般是通过向外形 技术和材料这样一些形态上的特征进行区分，这里列出了法式建筑、中式园林建筑和现代高层建筑。三种不同的建筑风格，显然，它们都有自己独特的共性特点，每一种建筑风格的产生，都是人们对居住的思考和尝试，挖掘出的可能性，它是一个长期建筑历史发展的结果。

之所以称为风格，是因为经过长时间的实践，它们已经被证明具有良好的工艺可行性、性能和实用性，并且可以直接用来遵循和模仿。

对于软件系统来说，大部分的设计也是例行设计，有经验的软件开发人员，经常会借鉴现有的一些解决方案，然后按照新的系统的要求，把它改造成一个新的设计。

软件体系结构风格是描述特定系统组织方式的一些惯用范例，它反映的是众多系统所共有的一些结构和语义，使用软件体系结构风格，可以促进设计的重用和大量代码的重用，使系统更加易于理解，而且标准化的风格，也有利于系统的互操作。

卡内其梅龙大学的 Garlan 和 Shaw 教授，是软件体系结构最早的研究者，他们认为体系结构风格是按照结构组织的模式来定义系统，具体地说，就是定义了构件和连接件的语法和连接方法，同时他们也提出了五种通用的体系结构风格。

第一种是独立构件风格，典型的系统有进程通信和事件系统等，其中事件系统，又可以进一步地分成隐式调用和显式调用。

第二种是数据流风格，主要包括批处理和管道过滤器这样一些类型。

第三种，是以数据为中心的风格，像常见的仓库、数据库系统、超文本系统等，都属于这种风格。

第四种是虚拟机风格，主要包括解释器和基于规则的系统。

第五种是调用返回风格，像主程序-子程序、面向对象、层次结构等都属于这种风格。

下面我们来详细介绍其中几种典型的，软件体系结构风格。

主程序-子程序，是结构化程序设计的一种典型风格，它是从功能设计出发，把系统进行模块化的分解，由主程序调用那些所分解的子程序模块，来实现完整的系统功能。这种风格的系统是由一个主程序和一系列子程序组成。

主程序和子程序，以及子程序之间，通过调用和返回来形成一个层次化的系统结构。大家在学习 C 语言课程时，编写的程序就是典型的主程序-子程序风格。

面向对象风格，是建立在数据抽象和面向对象的基础上，我们把数据和它相应的操作，都封装在一个抽象数据类型或者对象中，整个系统可以看成是一个对象的集合，每一个对象都有自己的功能集合。

这种风格的系统，它的构件是类和对象，对象之间通过函数调用和消息传递，来进行交互，它的特点是具有信息隐藏性，构件之间只通过接口和外部进行交互，它的内部结构是被封装隐藏起来的，像 C++和 Java 程序都是典型的面向对象风格。

管道过滤器风格，是把系统的任务分成一系列连续的处理步骤，这些步骤通过系统的数据流进行连接，一个步骤的输出是下一个步骤的输入。

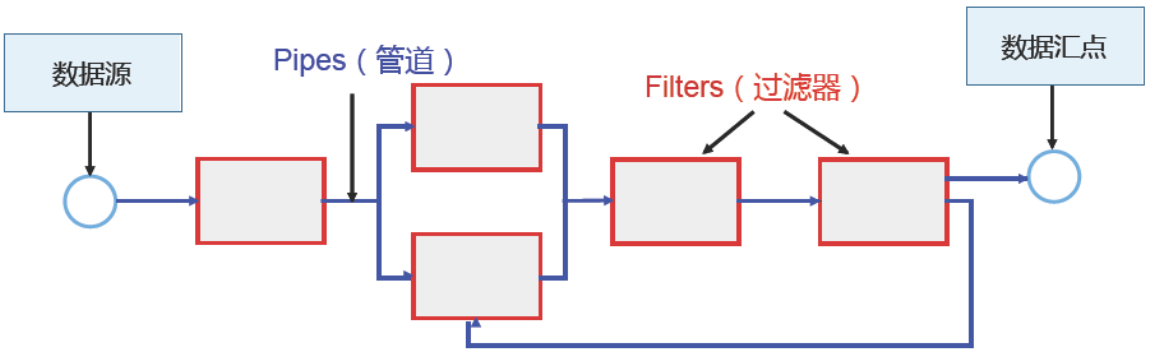


图 4.2 管道过滤器风格

如图 4.2 所示显示了管道过滤器风格的一个直观结构，其中的构件称为过滤器，过滤器对于输入的数据流进行内部处理，然后产生输出数据流，而传送数据的连接件就称为管道。

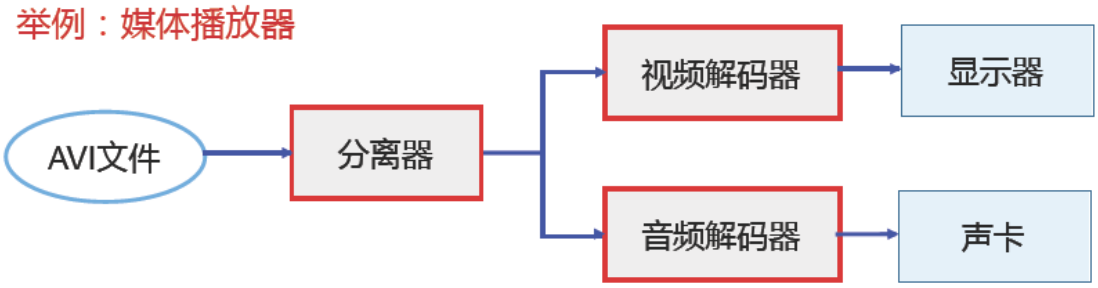


图 4.3 一个媒体播放器

如图 4.3 所示给出的是一个媒体播放器的例子，它的输入是一个 AVI 文件，包括了视频和音频数据，这个文件经过分离器进行处理，把视频和音频数据分离成两个流，其中视频数据传送给视频解码器，音频数据传送给音频解码器，视频解码器和音频解码器分别把压缩的视频和音频数据，解码成原始的图象数据和原始的音频数据，然后再分别输出给显示器和声卡。

在管道过滤器风格中，构件具有良好的隐藏性和高内聚、低耦合的特点，可以很好地支持软件的重用和扩展，但是这种结构不适合交互应用的情况，如果管道过长，或者过滤器过于复杂，系统的性能就会大大降低。

我们经常使用剪贴板，这个程序可以把剪贴的数据进行短时间的存储，然后支持文档和应用之间的数据传递和交换，那么剪贴板和其他应用程序之间应该是一个什么样的结构呢。

剪贴板需要一个公共区域来存储交换的信息，这个区域就像是一个共享的数据仓库，不同的应用程序通过 Copy/Paste 来访问剪贴板，并且使用这个区域来交换格式化的信息

仓库体系结构是一种以数据为中心的体系结构风格，在这种系统中所有的功能模块都访问和修改一个单一的数据存储，这个存储也被称为是一个集中式的仓库，而功能模块之间是相互独立的，他们之间的交互，都是通过这个仓库来完成。

这种体系结构适合于实现那些经常发生改变，而且具有复杂数据处理的任务，只要仓库的定义良好，就可以很方便地增添功能模块，从而实现向系统添加新的服务，但是这种系统的主要问题在于，每个功能模块和仓库之间的耦合非常高，集中式的仓库很有可能，成为系统性能的瓶颈。

这是一个现代程序设计语言编译器的系统结构图，它采用的就是一种仓库形式的体系结构风格，我们从图中可以看出，编译器、调试器和编辑器都是单独的工具。

编译器增量的产生，程序的语法分析数和符号表，代码调试器和编辑器，对它们进行读取和使用。

一般来说，数据库系统也是典型的仓库体系结构风格，它主要包括两种构件，一种是中心数据库，保存当前系统的数据状态，另一种是多个独立的应用，对数据库进行读取，对于复杂的应用系统来说，可能会涉及到多个异构数据库的集成，一个数据库可以被多个应用访问，一个应用也可以访问多个数据库，当然现在有很多，异构数据库的集成技术，在这里不再进行详细讲述。

4.3.4 软件体系结构风格（二）

（1）层次结构

层次化已经成为一种复杂系统设计的普遍性原则，很多复杂软件的设计，从操

作系统到网络系统，再到一般的应用，几乎都是以层次结构来建立的。

在层次结构中，系统被分成若干层次，每一层次由一系列的构件所组成，层次之间存在接口，通过接口形成调用和返回的关系，下层构件向上层构件提供服务，上层构件被看作是下层构件的客户端。

例如：安卓操作系统层次结构，它被划分成 Linux 内核、系统运行库、应用框架和应用四个层次。这是一个网络的分层模型，它包括物理、数据链路、网络、传输、会话、表示和应用等层次，并且严格限定某一层的构件只能和同一层的对等实体，以及和它紧邻的下一层进行交互。

（2）客户机/服务器体系结构

客户机/服务器体系结构是一种分布式的系统模型，它把应用程序的处理分成了两个部分，一个是客户机，负责和用户进行交互，另一个是服务器，它为客户机提供服务。

最早的分布式系统是两层的，一个是客户机，一个是数据库的服务器，后来发展成为三层结构，中间增加了一个应用服务器层，现在已经发展成为一个多层的体系结构。如图 4.4 所示。

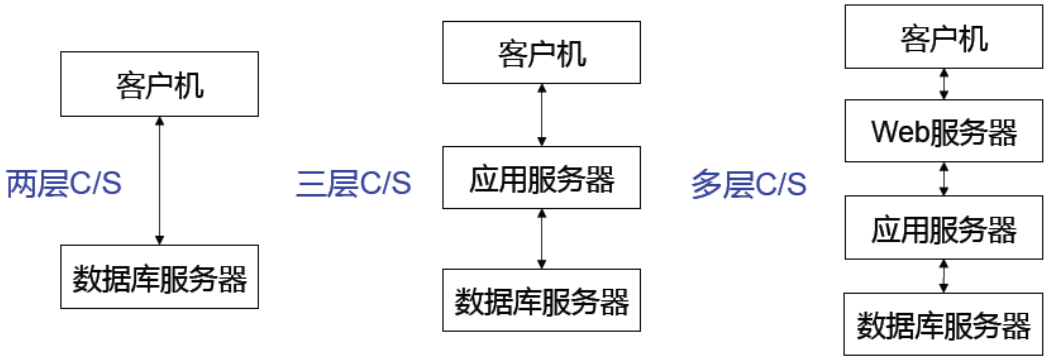


图 4.4 客户机/服务器体系结构

两层的客户机/服务器结构，也称为胖客户端模型。在这种系统中，数据库服务器负责数据的管理，客户机实现应用逻辑，并和用户进行交互。

具体的过程是这样的，用户在客户机的系统界面上输入数据，并发出业务处理请求，客户机上的业务处理程序接受请求，进行处理，在需要进行数据存取的时候，向服务器发出数据存取请求，服务器将执行数据库连接、查找和存取的操作，并把结果进行返回。客户机的业务处理程序处理完用户请求之后，在表示层上进行输出。

两层结构的主要问题在于客户端的负担过重，系统维护和升级比较困难，扩展性也不太好，与胖客户端相对应，还有一种瘦客户端，二者的区别在于业务逻辑到底在客户端多一些，还是在服务器端多一些。

如果客户端执行大部分的数据处理操作，那么就是胖客户端，如果客户端具有很少，或者没有业务逻辑，那就是瘦客户端。

针对两层结构的问题，人们提出了三层客户机/服务器结构，它把应用系统，分成表示层、功能层和数据层三个部分。而且把这三层进行明确的分割，在逻辑上使其独立，其中表示层是应用的用户接口部分，负责实现用户和应用之间的对话，业务逻辑层实现应用程序的处理逻辑和规则，数据层实现对数据库的存储、查询和更新。

在硬件部署时，三层结构的表示层可以配置在客户端，功能层和数据层分别放在不同的服务器上，这样就大大降低了客户端的负荷，也减轻了系统维护和升级的成本和工作量。

在增加新的业务处理时，可以相应的增加装有功能层的服务器，系统的灵活性和伸缩性变得很强，特别是随着系统规模的加大，这种结构的优势就变得更加明显。但是需要注意的是，三层结构中，各层之间的通息效率如果很低，即使分配给各层的硬件很强，系统在整体上也达不到所要求的性能，所以在设计的时候，必须要慎重考虑三层之间的通讯方法、频率和数据量。

（3）浏览器/服务器结构

随着互联网技术的发展，浏览器/服务器结构，也称 B/S 结构逐渐流行，它是三层 C/S 结构的一种实现，它的客户端就是浏览器，表示层负责处理客户端，所需要的展现逻辑应用层负责所有的业务逻辑，数据层负责对数据库的操作。

层次架构的基本思想，是把一个应用分成多个逻辑层，其中每一层都有通用或者特定的角色，这样做可以使应用更易于扩展，在实际部署时，功能层并不一定只驻留在同一台服务器上，数据层也是这样。

（4）集群结构

如果功能层或者数据层分布在多台服务器上，那么就形成了集群式的系统结构，许多 C/S 结构的应用系统，都是从数据库中检索数据，再展现给用户。在用户更改数据之后，系统再把更新的内容，存储到数据库中，具体实现的时候，数据存取逻辑是和用户界面绑在一起的，这样可以提高应用程序的性能，但是这种做法是有问题的，一般来说，用户界面的更改，往往要比数据存取更加频繁，而且除了数据存取之外，在应有程序的业务逻辑中，往往还会包含很多其他的业务逻辑，显然，用户界面一旦更改，就会影响到数据存取逻辑部分，这样是不满足高内聚，低耦合的设计原则。

那么纯粹的 B/S 结构，能不能解决这个问题，由于用户界面仍然需要显式地调用功能层的业务逻辑,所以还是很难避免用户界面修改,导致业务逻辑修改这个问题，所以说，我们需要进一步改进这个结构，以保证各个部分可以单独修改，而不影响其他部分。

我们先来分析一下影响 Web 系统模块化的一些因素，在这种系统中，用户界面的更改，要比业务逻辑更加频繁，如果把两部分的代码都放在用户界面中，那么每一次更改用户界面，就会引起业务逻辑的修改。有些情况下，应用程序会用不同的方式，来显示同一个数据。

比如：用表格、各种图形等等，那么与业务逻辑相比，用户界面的代码，对设备的依赖性就会更大。

设计美观而且有效的用户界面和开发复杂的业务逻辑，往往需要不同的编程技能，可能需要不同的人来编写，而且为用户界面创建自动化的测试，也要比业务逻辑更难、更耗时。

（5）MVC 结构

为了解决上述问题，人们提出了模型-视图-控制器结构，也称 MVC 结构，它是把应用程序的数据模型、业务逻辑和用户界面，分别放在独立的构件中，这种结构，是在传统的 B/S 体系结构基础上，加入了一个新的元素——控制器。由控制器来决定视图和模型之间的依赖关系，这样用户界面的修改，就不会对数据模型，或者业务逻辑造成太大的影响，其中模型用于管理应用系统的行为和数据，它在发生变化时，会通知视图，并且允许视图查询它的当前状态，同时也允许控制器访问在模型中封装的业务功能。视图负责模型内容的展现，控制器则是定义了应用系统的控制行为，负责分派用户的请求，并且针对用户请求选择相应的视图。

现在我们来看一个，Web 应用的执行过程，客户端发送一个 HTTP 请求，这个请求首先会进入到控制器，然后控制器去获取数据，把它封装成模型，最后再把模型传

递到视图中进行展现。

这个过程存在下面一些问题，我们看到每一次请求都要经过控制器、模型、视图这样一个流程，用户才能看到最终展现的界面，整个过程比较复杂，实际上，视图是依赖于模型的，也就是说，如果没有模型，视图是无法展现最终的效果，另外渲染视图的过程是在服务端来完成的，最终呈现给浏览器的是带有模型的视图页面，系统的性能无法得到很好的优化。为了使数据展现过程更直接，而且有良好的用户体验，我们需要对 MVC 模式进行改进。

现在来看一种改进的方案，首先从浏览器发出 AJAX 请求，然后，服务端接受这个请求，返回 JSON 的数据给浏览器，最后在浏览器中，进行界面的渲染。如果我们把浏览器这一端视为前端，服务器视为后端的话，上面改进后的 MVC 模式，可以简化为前后端分离的模式，现在有一种 REST 的机制，可以实现这个模式。

4.3.5 软件体系结构风格（三）

我们用过集成开发环境来编写和调试代码，知道这个软件的体系结构是什么类型吗，我们来看一下，程序调试器的工作过程。

首先，在调试器设置一个断点，编译器、文本编辑器和变量监视器通过注册的方式和断点发生关联，程序运行到断点处的时候，系统就会触发文本编辑器和变量监视器，也就是说，是断点事件隐含地造成它们的调用，这个时候文本编辑器滚动屏幕到断点处，变量监视器刷新变量的当前值，这是一种事件风格的系统。

（1）事件驱动

事件驱动分成显式调用和隐式调用两种类型，顾名思义，在显式调用中，各个构件之间的互动是显式地调用函数，或者程序来实现的，而且调用的过程和次序，也是预先设定不变的。

在隐式调用中，调用过程并不固定，而且事先不可知，各个构件之间是通过事件的方式来进行互动。

在事件风格的系统中，隐式调用是一种常见的形式，系统把应用看成是一个构件的集合，作为事件源的构件，并不是直接调用其他构件，而是触发或者广播一个或多个事件，其他响应事件的构件是作为事件处理器，预先在事件中进行注册，当事件被触发的时候，事件管理器就会调用这些已经注册的构件进行事件的处理。

在前面的程序调试器例子中，事件源是调试器、编辑器和变量监视器是事件的处理器，集成开发环境是事件管理器，编辑器和变量监视器向调试器进行注册，接收断点事件，遇到断点的时候，由调试器来发布事件进而就触发了编辑器和变量监视器。

实现事件处理的第一种方式是发布订阅模型，在这个模型中，有两种角色，分别是发布者和订阅者，订阅者可以订阅一个或多个频道，而发布者可以向指定的频道来发送消息，所有订阅了这个频道的订阅者，就会收到这个消息，但是没有预定的程序，不会收到这个消息。

另一种实现方式是观察者模式，观察者被注入到被观察者中，进行事件的监听，随时对被观察者的变化做出反应，这种方式和发布订阅模式的不同在于，发布订阅模式是在观察者和被观察者中间，增加了一层间隔。二者之间是一种松耦合的关系。

前面我们介绍了几种，常见的体系结构风格，但是在实际应用时，需要借助丰富的经验来进行判断和选择，一般情况下，大部分的实际系统往往是几种体系结构的组合应用，在系统分析和设计过程中，首先要把整个系统，作为一个功能体来进行分析和权衡，得到一个最顶层的体系结构，如果系统中的元素还是比较复杂，那就继续进行分解，再得到某一部分的局部体系结构。也就是说，我们要把焦点集中在系统的

总体结构上，避免过多地考虑实现细节。

在体系结构的选择上，我们需要考虑技术因素和质量因素两个方面。

技术因素包括，使用什么样的构件和连接件，运行的时候构件之间的控制机制，是如何实现，数据如何通讯，数据和控制是如何交互，质量因素包括：可修改性、要考虑到算法、数据表示和系统功能的变化和扩展、性能、以及可复用性等。

（2）软件体系结构的选择

根据实际的开发经验，我们可以总结一些体系结构的选择原则，比如说，一般层次化在任何系统都是要使用的，如果系统的功能可以分解成一系列连续的处理步骤，那么就可以考虑批处理，或者管道-过滤器风格，如果系统的核心问题是数据管理，而且数据是持续存储的，那么可以使用仓库结构风格。

如果任务之间的控制流，是可以预先设定的，可以考虑主程序、子程序，或者面向对象的风格，对于任务需要高度灵活可配置，或者任务是被动的，就可以考虑事件系统或者客户机/服务器的结构。如果设计了一种计算，但是没有机器可以支持它的运行，可以考虑使用虚拟机或者解释器的体系结构。

体系结构的选择和应用，需要丰富的经验，也需要不断的创新，可以在实际的项目开发中，参考已有的成熟方案，在实践的过程中加以运用和掌握。

4.4 讨论题

1、什么是设计？什么是软件设计？什么是软件体系结构？

2、软件设计与软件需求的差异在哪里？软件设计包括哪些方面的设计？

3、有些人说“软件设计就是画画图形、写写文档，既没有什么意思，也没有什么价值”（观点 I），你同意这种观点吗？说说你的理由。也有些人说“软件设计是一回事，而编写程序又是一回事，因此软件设计纯粹是浪费时间和精力，不如把精力用在编写程序上”。（观点 II），你同意这种观点吗？说说你的理由。怎样进行软件设计，才能保证软件设计到位，才能使软件设计“有用”呢？谈谈你的想法。请分享你的观点。

==========

什么是设计？

答：设计通常是指将人们头脑中想象的事物表达成模型或文档的过程，是设计者依据一定的规则将自己的想法用特定的方法或方式表达出来的。

- 什么是软件设计？

答：软件设计是将软件需求的详细功能实现出来成为一个可以使用的软件系统的过程。

- 什么是软件体系结构？

答：软件体系结构是软件设计过程中的一个层次，这一层次超越计算过程中的算法设计和数据结构设计。体系结构问题包括总体组织和全局控制、通讯协议、同步、数据存取，给设计元素分配特定功能，设计元素的组织，规模和性能，在各设计方案间进行选择等。软件体系结构处理算法与数据结构之上关于整体系统结构设计和描述方面的一些问题，如全局组织和全局控制结构、关于通讯、同步与数据存取的协议，设计构件功能定义，物理分布与合成，设计方案的选择、评估与实现等

- 软件设计与软件需求的差异在哪里？

答：软件设计是实现软件需求的过程，软件需求是根据需要整理为希望设计软件的需求详细说明。

- 软件设计包括哪些方面的设计？

答：软件设计包含文档设计，数据库设计，界面设计，组件设计，中间件设计，服务器配置，系统参数配置，代码设计等。

有些人说“软件设计就是画画图形、写写文档，既没有什么意思，也没有什么价值”（观点 I），你同意这种观点吗？说说你的理由。

-不同意。没有完整、准确、清晰的软件设计，则无法完整、快速、正确完成程序编写。

也有些人说“软件设计是一回事，而编写程序又是一回事，因此软件设计纯粹是浪费时间和精力，不如把精力用在编写程序上”（观点 II），你同意这种观点吗？说说你的理由。

-不同意。编写程序建立在软件设计的基础之上。

- 怎样进行软件设计，才能保证软件设计到位，才能使软件设计“有用”呢？

答：在明确的软件需求的基础上，对软件进行建模、分块，并完成撰文、分配等。

在此后的进度中，应严格按照软件设计进行。

4.5 测试题

单选题 (10 满分)

1.随着软件系统的规模和复杂性越来越大，（ ）变得更加重要。

A.算法的选择
B.数据结构的设计
C.数据库的构造
D.系统的全局结构设计

2.下面的说法（ ）是错误的。

A.软件体系结构的最佳表示形式是一个可执行的软件原型
B.软件体系结构描述是不同项目相关人员之间进行沟通的使能器
C.良好的分层体系结构有利于系统的扩展与维护
D.设计模式是从大量成功实践中总结出来且被广泛公认的实践和知识

3.内聚表示一个模块（ ）的程度。

A.可以被更加细化 B.仅关注在一件事情上 C.能够适时地完成其功能 D.联接其他模块和外部世界

4.良好设计的特征是（ ）。

A.模块之间呈现高耦合
B.实现分析模型中的所有需求
C.包括所有组件的测试用例
D.提供软件的完整描述 E.选项 B 和 D F.选项 B、C 和 D

5.程序编译器的体系结构适合使用（ ）。

A.仓库体系结构
B.模型－视图－控制器结构
C.客户机 / 服务器结构

D.以上选项都不是

6.网站系统是一个典型的（　）。

A.仓库体系结构

B.胖客户机/服务器结构

C.瘦客户机/服务器结构

D.以上选项都不是

7.Word、Excel 等应用系统适合采用（　）结构风格。

A.层次系统

B.事件系统

C.解释器

D.管道-过滤器

8.在分层体系结构中，（　）实现与应用程序的处理逻辑和规则。

A.表示层

B.数据层

C.实体层

D.功能层

9.与 C/S 架构的信息系统相比，B/S 架构的信息系统的优势是（　）。

A.具备更高的安全性

B.更容易部署和升级维护

C.具备更强的事务处理能力，易于实现复杂的业务流程

D.用户界面友好，具有更快的响应速度

10.对于观察者模式，下面的（　）说法是错误的。

A.观察者的更新是被动的

B.被观察者可以通知观察者进行更新

C.观察者可以改变被观察者的状态，再由被观察者通知所有观察者

D.以上所有选项

第 5 部分 单元测试

5.1 本章的教学目标

通过本章的学习，使学生初识软件工程技术中的单元测试技术与方法，包括：黑盒测试方法、白盒测试方法和单元测试工具等。

5.2 本章的教学方法

本章的教学方法采用“翻转课堂式”教学模式，学生可以通过互联网使用“学堂在线”网上优质教育资源，选择由清华大学刘强副教授主讲的“软件工程”课程的“第 4 章 单元测试”课外观看学习，网址：
http://www.xuetangx.com/courses/course-v1:TsinghuaX+34100325X+sp/about ，通过视频中教师的讲解，回到课堂上教学交流讨论，并相关的完成习题。

通过理论教学、课堂讨论、适当的布置课外习题作业来完成学习。

5.3 本章的教学内容

本章主要介绍软件工程技术中的单元测试技术与方法，包括：黑盒测试方法、白盒测试方法和单元测试工具等方面。

5.3.1 单元测试概述

在编码完成之后，就迫不及待地进行软件集成工作，表面上看这样做，可以很快地看到实际系统，但是由于缺少应有的单元测试，系统中经常充满了各种 Bug，也会造成故障难以定位，修复困难等问题，这样就会增加额外的时间和成本。

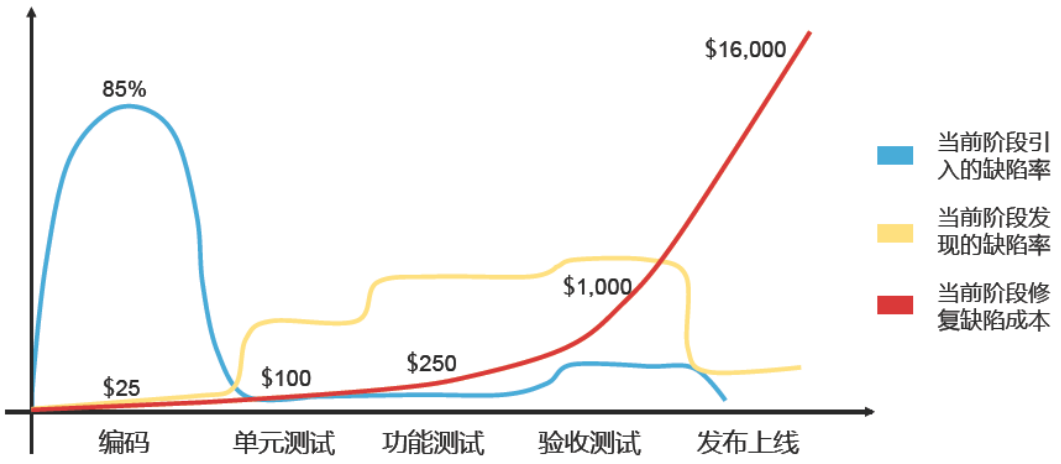


图 5.1 编码和测试的不同阶段

如图 5.1 所示展示了编码和测试的不同阶段，引入和发现缺陷的情况，以及修复它们的成本，开发人员在编写代码的同时，不可避免地会引入缺陷，图中的数据告诉我们，大多数的缺陷，是在编码的初期产生的，修复它们的成本在初期也比较低，越往后拖，增长的幅度就越大。

（1）单元测试

单元测试是软件开发过程中，重要的质量保证活动，它的质量在很大程度上，影响着软件产品的最终质量。

单元是构造系统的基础，这就好比是长城的城墙，如果每块砖的质量不过关，那么建造的城墙就不会坚固。编写代码也是如此，底层代码出现问题，就会牵扯到上层代码，它的修改也可能会导致，其他代码的一连串改动，从而影响整个产品的质量，只有打造坚实的基石，才能保证上层建筑的坚固。

单元测试，是对软件中的最小可测试单元进行检查和验证，它要测试和验证程序代码中，每一项功能的正确性，为后续的开发提供支持。

编写单元测试，可以使开发人员从调用者的角度，进行观察和思考，这样编写的程序，更易于调用和测试，也可以将程序的缺陷降低到最小。

单元测试也是一种文档化的行为，测试用例可以说明函数或者类，是如何使用的，自动化的单元测试，也有助于进行回归测试。

我们在前面讲过，产品的质量是在构建的过程中形成的，开发人员必须对自己的代码负责，单元测试就是对所开发代码质量的，一个基本的承诺。

对于程序员来说，编程不仅仅意味着编写代码，还要完成单元测试。在现实中我们发现，那些代码质量最好，开发速度最快的程序员，也是单位测试做得最好的程序员。

单元测试的内容，主要包括以下方面：

（1）模块接口，主要是检查参数表，调用子模块的参数，全程数据、文件 IO 等内容。

（2）局部数据结构，主要检查数据类型说明，初始化、缺省值等方面的问题，还要查清全程数据对模块的影响。

（3）边界条件，要特别注意数据流，或者控制流中，刚好等于、大于，或者小于边界值的情况，因为这些地方非常容易出错。

（4）独立路径，主要是对模块中重要的执行路径进行测试，通过对基本执行路径和循环进行测试，可以发现大量的路径错误。

（5）出错处理，这一部分，是检查模块的错误处理功能，是否存在错误或者缺陷。

如果对模块运行时间有要求的话，还要专门进行关键路径测试，以确定最坏情况下和平均意义下，影响模块运行时间的因素。

通过单元测试，我们可以保证单元模块的质量，这样为整个的系统打造了坚实的基础。

单元测试包括快速、独立、可重复、自我验证和及时五个原则。

单元测试必须要快，太慢的话就没有人再愿意，频繁地运行它。

单元测试也要相互独立，否则一个测试没有通过，就会导致一连串的失败，很难定位问题。

单元测试是要能够重复执行的，而且结果也是可以重现的。

单元测试执行结束之后，应该由布尔输出，来显示是否通过，而不应该通过人工的方式来确认。

开发人员应该及时编写单元测试代码，最好是在开发单元代码之前来完成。

单元测试过程主要包括以下步骤：

首先，我们要找出程序中潜在的最大问题区，来确定哪些部分需要做单元测试。

其次，针对要做的测试，来编写相应的测试用例。

之后编写单元测试代码，执行单元测试，产生测试结果。

如果测试结果满足了，测试质量的要求，那么整个测试结束。否则还要根据结果和测试的要求，来修改和增加新的测试用例，再进一步地进行测试。

衡量测试质量的指标，一般有测试通过率和测试覆盖率。

测试通过率是在测试过程中，测试用例的通过比例，单元测试一般都是要达到100%。

测试覆盖率是用来衡量测试的完整性，它可以告诉我们测试是否充分以及测试的弱点在哪里，我们可以通过增加测试用例，来提高覆盖率，进而提升测试质量。

单元测试一般用代码覆盖率来衡量，代码覆盖率大致达到 70%到 80%就可以了，过高的覆盖率可能会造成测试成本的大大增加。

代码覆盖率的度量，有多种不同的方式，包括：语句覆盖、条件覆盖和路径覆盖等。

单元测试包括静态测试和动态测试两种类型。

其中静态测试是通过人工分析，或者程序正确性证明的方式，来确认程序的正确性。而动态测试则是通过动态的分析和运行程序的方式来检查和确认，程序中是否存在问题。

对于测试用例设计来说，有黑盒测试和白盒测试两种技术。

（2）黑盒测试

黑盒测试也称功能测试，它是把测试对象看成是一个黑盒子，只是根据需求规格说明，来设计有代表性的测试输入，再通过测试执行的输出结果，来检查程序的功能，是不是符合规格说明。

（3）白盒测试

白盒测试也称结构测试，它允许测试人员根据程序内部的结构，来设计测试用例，对于程序的逻辑路径等进行测试。

一个单元模块并不是独立存在的，模块之间总会存在相互调用的关系。

比如说一个被测模块，通常会被上层的模块来调用，同时它又调用多个下层的模块，那么，在单元测试的时候，如何来保证被测模块是独立的，并且能够构成一个可运行的程序呢。

在这种情况下，我们需要开发驱动模块和桩模块。他们将被测模块进行隔离，并帮助完成单元测试。

其中驱动模块，用于替代上层的调用模块，它去调用被测模块，并且判断被测模块的返回值，是否与测试用例的预期结果相符。

桩模块，则用于替代下层的调用模块，它要模拟地返回所替代模块的各种可能的返回值。

单元测试一般需要借助工具来编写测试代码，目前最流行的是针对不同编程语言的系列 Unit 框架。

（4）单元测试 xUnit 方法

xUnit 测试,一般适合单个函数或类的测试,尤其是纯函数或者接口级别的测试，对于一些复杂场景就很难适用，但是这些场景往往才是测试的重点，也是难点所在。

以分布式系统为例，类和函数级别的单元测试是远远不够的，这种系统主要是测试进程之间的交互，例如，一个进程收到客户请求，应该如何处理，然后转发给其他进程等。

另外分布式系统的测试，通常更关注一些异常的路径，那么这种复杂的场景，应该如何进行单元测试。

（5）单元测试 Mock 方法

Mock 方法，可以帮助我们解决，前面所说的那些复杂场景中遇到的难题，这种测试是在测试过程中，对于某些不容易构造或者不容易获取的对象，用一个虚拟的对象，也就是 Mock 对象来创建，以便进行测试。

这里列出了一些 Mock 测试适合的场景。

现在我们举例说明，Mock 测试的基本方法，很多同学可能都玩过接龙红包，它是通过猜金额，这样一个小游戏的方式来实现朋友之间的互动，并且可以领取春节红包。在这个程序中，GiftMoney 这个类，允许设定红包的金额和上下限，并且将所猜的金额和设定金额进行比较，由于数据是存放在数据库中，所以这一个类，就需要调用数据库存取访问的方法。那么这种情况下，应该如何进行测试。刚才红包的问题可以抽象成，ClaassA 对 ClassB 的访问，在进行 Mock 测试时，需要把程序之间的直接调用，转换成通过接口进行访问。

也就是说，先抽象出一个接口 InterfaceB，ClassA 直接调用它，ClassB 来具体地实现它，然后在使用 Mock 对象，来模拟所调用的对象及其行为，这时被测模块，并不知道它所引用的究竟是真实对象，还是 Mock 对象。

通过这样的方式，可以把被测模块和所依赖的模块进行隔离，从而实现复杂场景下的单元测试。

下面我们再来思考一个，微信抢票应用的例子。

开发人员编写的抢票程序，是通过直接调用系统函数，来获得当前的系统时间，在单元测试的时候，需要设定不同的时间进行抢票测试，如果我们要利用真实的系统时间来测试的话，就只能苦苦等到实际设定的时间，才能进行检验，显然这是一个很笨的做法。那么如果使用 Mock 的方法进行测试，应该如何来重构这个程序呢。

对于这个问题，请大家结合，刚才讲解的 Mock 测试内容，在课下进行思考和讨论。

5.3.2 黑盒测试方法

测试用例设计是测试活动的基础和关键，它可以指导人们系统地进行测试，对于测试的执行起到事半功倍的作用。

测试不可能是完备的，它会受到时间的约束。测试用例可以帮助我们分清先后主次，以便更有效地组织测试的工作，从而提高测试效率，降低测试成本。

在测试的过程中，临时性地发挥也许会有灵感出现，但是大多数情况下，我们会感觉思维混乱，可能有一些功能没有测到，而另一些功能已经重复测过几遍。

测试用例可以帮助我们理清头绪，避免重复和遗漏，测试用例的通过率和软件缺陷的数量，是检验软件质量的量化标。过对测试用例的分析和改进，可以逐步地完善软件的质量，另外测试用例也可以用于衡量测试人员的工作量、进度和效率，从而更有效地管理和规划测试的工作。

什么是测试用例，测试用例是为了一个特定的目标，而设计的一组测试输入、执行条件和预期结果。

它的目的是测试某个程序路径是否正确，或者何时程序是否满足某个特定的需求，测试用例一般包括：测试用例值是测试需要的一些输入值，期望结果是程序满足预期行为，而应当产生的结果。

我们要验证一个程序是否正确，就要事先明确这个程序应该满足的期望行为，如果测试结果和期望结果相符，那么我们就认为它是正确的，另外，测试的时候需要预先设定软件处于测试输入的一个稳定状态，前缀值就是把软件置于合适的状态，来接受测试用例值的，任何必要条件后缀值，就是测试用例值输入以后，为了查看结果或

者终止程序，而需要的任何输入。后缀值又可以再细分为验证值和结束命令两种类型。

比如，我们测试一个移动电话软件，测试用例值可以是多种电话号码，期望的结果就是接通或者未接通，除了这两部分之外，测试的时候我们还需要电话开启，并且处于可拨号的稳定状态，也就是所谓的前缀值，输入电话号码之后，我们还要按下呼叫或者取消按纽，来查看测试结果，这就是后缀值。

测试用例设计应该考虑下面的因素，因为穷举测试是不可行的，所以我们必须要选择少量有代表性的数据，使每一个测试用例，都能覆盖一组范围，或者特定情况。

测试用例要确切地反映系统设计中可能存在的问题，我们要根据需求规格说明，并且结合各种可能情况，来综合考虑测试用例。

测试不能只考虑正确输入情况，还要考虑错误或异常输入，并且分析怎么才能产生。这些错误和异常测试用例要基于用户的实际使用情况，从用户的角度来模拟测试的输入。

黑盒测试是把测试对象看成是一个黑盒子，根据需求说明来检查程序是否符合功能要求，在单元测试部分，我们重点讲解，等价类划分和边界值分析这两种方法。

等价类划分是把输入域划分成尽可能少的若干子域，并且要求这些子域是两两不相交的，那么每一个子域，就称为一个等价类。对于同一个域来说，可能会存在不同的划分方法。

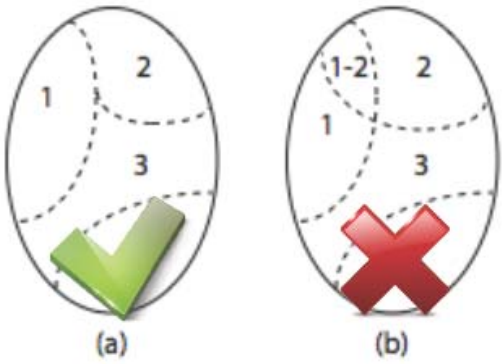


图 5.2 等价类划分两种情况

如图 5.2 所示显示两种划分，那么哪一种划分是等价类划分呢，左边图里面的子域，两两不相交的，右边的 1 和 2 是有交集的，根据上面的定义，左边是等价类划分，但是右边不是。

在等价类划分中，我们用同一个等价类中的任何输入，对软件进行测试的时候，软件都会输出相同的结果，所以只需要从每一个等价类中，选取一个输入作为测试用例就可以构成。

软件一个完整的测试用例集，对于相同的等价类划分，不同的测试人员所选取的测试用例集可能是不同的。那么良好的选择需要经验的积累。

等价类可以分为：有效等价类和无效等价类。

其中有效等价类是有合理的输入数据构成的集合，它能够检验程序是否实现了规格说明中预先规定的功能和性能。

无效等价类是不合理输入数据构成的集合，它可以发现程序异常处理的问题，来检查程序是否具有一定的容错性。

我们设计测试用例的时候，要同时考虑这两种等价类，因为软件不仅要能够接收合理的数据，也要能够经受意外的考验。

下面我们结合一些常见的不同数据类型来简单说明，如何进行等价类的划分，对

于一般的简单数值变量来说，如果输入规定了取值范围，那么范围内是一个有效等价类，范围外是两个无效等价类。

下面来看一个例子，程序的输入参数 X 是小于 100，大于 10 的整数，根据上面的方法，有一个有效等价类，就是 X 大于 10 小于 100，有两个无效等价类，分别是 X 小于等于 10 和 X 大于等于 100。

对于字符串变量，如果规定了输入字符，遵守的规则，那么我们可以确定有一个有效等价类，就是符合规则的字符串和若干无效等价类，它们是从不同角度，违反规则的字符串。

例如：姓名是长度不超过 20 的，非空字符串，只是由字母组成，数字和其他字符都是非法的，显然，我们可以有一个有效等价类，就是满足了上述所有条件的字符串，那么有几个无效等价类。我们可以看长度是一个因素，组成字符串的字符也是一个因素，那么长度可以是 0 或者超过 20，然后其他是非字母的字符，这样我们就得到了三个无效等价类，分别是空字符串、长度超过 20 的字符串包含了数字或者其他字符的字符串，对于枚举类型，如果规定输入数据是一组值，假定 n 个，并且程序要对每一个输入值分别处理，那么就可以确定有 n 个有效等价类和 1 个无效等价类。

比如说，某一个程序根据不同的学历，分别计算岗位工资，其中学历可以是专科、本科、硕士、博士四种类型，那么显然就有四个有效等价类，分别是专科、本科、硕士、博士，还有一个无效等价类，就是其他学历。如果我们把专科、本科、硕士、博士是按一种方式来计算岗位工资，这个时候应该如何划分等价类。

数组是一组具有相同类型元素的集合，那么数组的长度和类型都可以作为等价类划分的依据，假设某一个程序的输入是一个长度为 3 的整数数组，那么根据数组的长度可以划分成一个有效等价类，就是长度是 1 到 3 的所有合法数组，还有两个无效等价类，分别是空数组和所有长度大于 3 的数组，比如：-9、0、12、5，那么这个数组因为长度是 4，所以是一个无效等价类。

如果我们对于数组的元素，还有其他的附加的约束，比如说，数组 oper 的元素，它的取值范围-3 到 3，那么还要适当地来增加相应的等价类，符合数据类型是包含了两个或两个以上，相互独立属性的输入数据。

在进行等价类划分的时候需要考虑输入数据的每一个属性的合法和非法。

比如这个例子，student 这个结构，它是包括了 name、course 和 grade 三个属性，那么在进行等价类划分的时候，先对每一个元素进行划分，然后再把这些等价类进行组合，最终得到了这个 student 结构的一个整个的等价类划分。

一般的测试对象都会有多个输入的参数，我们会把这些参数的等价类进行组合来设计测试用例，这个时候就要考虑如何进行组合以便保证等价类的覆盖率。

首先，所有参数有效等价类的每一个组合，就构成一个有效的测试用例。

其次，任一个参数的无效等价类与其他参数的有效等价类进行组合，构成一个无效的测试用例。

现在来看一个程序的例子，整数 ABC 是它的三个输入，分别代表三角形的三条边，这个程序判断这三条边构成的三角形类型，分别输出等边三角形、等腰三角形、一般三角形或者不构成三角形。这三个整数还满足大于等于 1，小于等于 100 这个条件。

现在使用等价类划分的方法来设计这个程序的测试用例。

首先进行等价类划分，第一种方法，是按照输入的取值进行划分，把 ABC 都划分成 0、大于 0、小于 0、或者 0、1、大于 1，小于 0 这样的一些等价类。

第二种方法是按照输出的几何特性进行划分，划分成等腰三角形、等边三角形、一般三角形和非三角形四个等价类，那么你会选择哪一种方法来划分等价类，显然第

二种方法最合适。

在多数情况下，等价类是根据输入域进行划分的，但并不是说不能从输出来反过来划分，事实上对于三角形这个问题，输出域的划分是最简单的方法。

等价类划分之后，我们就来设计测试用例，根据等价类的性质每一个等价类只需要选取一个有代表性的输入作为测试用例，这样我们就得到了四个有效的测试用例，我们还可以补充一些无效测试用例，根据前面提到的组合原则，每一个测试用例只有一个参数是无效的其余保持有效，这样就形成了健壮等价类的测试用例。

边界值分析，是对输入或输出的边界值进行测试的一种方法，通常也作为等价类划分的一个补充。

首先要确定边界，等价类的边界就是测试的重点，然后选取正好等于、刚好大于或者刚好小于边界的值作为测试数据。

实践表明，大多数的故障往往发生在输入或者输出域的边界上，针对边界情况设计测试用例通常会取得良好的效果。

这里列举了几个边界值分析的例子，第一个例子是一个文本输入区，它允许输入 1 到 255 个字符，那么空串长度为 1、255 和 256 的字符串都是边界条件值。

第二个例子，是要求输入 5 位十进制整数，我们可以使用刚好小于 5 位和大于 5 位的数值作为边界条件。

第三个例子是磁盘空间，这个边界值的选取和整数值方法类似。

边界值分析方法认为，故障往往出现在程序输入的边界值附近，但是它是基于单故障假设的，也就是说两个或两个以上故障同时出现，而导致失效的情况很少发生。

用这种方法设计测试用例的时候，每次只保留一个变量取边界值，其余的变量取正常值。

我们再来看一下，三角形程序的例子，按照前面提到的单故障原则，我们可以设计一些边界值的测试用例，在这里边界值分别取了，1、2、99 和 100。

健壮性测试，是对边界值分析的一个简单扩充，它除了边界内的五个取值之外，还增加了刚好超过边界的两个取值。这种方法可以检查超过极限值的时候系统的情况。

测试经验对于软件测试是非常重要的，人们可以经验或者直觉推测出程序中，可能存在的各种错误，然后有针对性地编写检查这些错误的测试用例，实际上，软件缺陷具有空间聚集性，对于代码的高危多发地段，投入更多的精力进行测试，往往可以发现更多的错误。

5.3.3 白盒测试方法

测试覆盖标准是衡量软件测试质量的一个重要指标，下面我们先了解一下，它的相关概念。

测试需求可以针对不同的软件制品来进行描述，比如：源代码、需求说明、设计文档等，如果我们要测试源代码的所有语句，那么每一条语句就是一个测试需求。

测试标准是把测试需求施加在一个测试集上的一组规则，例如：每一条语句都要执行，就是一个覆盖标准。

测试覆盖，是指对于每一条测试需求，至少要有一个测试用例可以满足这个需求，由于完全测试是不现实的，所以我们使用覆盖程度，或者覆盖率，来说明测试所做的情况，也就是说一个测试用例集，能够满足测试需求的比例。

我们用现实生活中的一个例子，来进一步解释一下上面的概念，假设我们要测试一个装有糖豆的袋子，糖豆有六种口味和四种颜色，那么对于口味来说，就有柠檬、

开心果、梨子、橘子和杏这六种口味的测试需求，对于颜色来说，有黄色、绿色、白色和橙色四种颜色的测试需求。

我们可以选择口味这个标准，也可以选择颜色这个标准，那么用哪一种覆盖标准更好一些，如果我们选择颜色这个标准，那么用每一种颜色各一个糖豆来测试，是可以达到百分之百的覆盖，但是这个时候会有向种口味没有覆盖到，如果选择口味这个标准，那么在覆盖了每一种口味的同时，也就覆盖了颜色的标准，也就是说，口味这个标准包含颜色标准。

在实际的项目里，选择覆盖标准可能要更复杂一些，一般需要考虑三个方面的因素：

第一，是处理测试需求的难易程度。

第二，是生成测试的难易程度。

第三，是用测试发现缺陷的能力。

前面讲过，白盒测试是利用程序内部的逻辑结构，对程序的所有逻辑路径进行测试。

那么为什么要做白盒测试呢？

我们先来看一个例子，假设有一台面包机，从上面倒入面粉和水，开动机器之后，下面出来的就是烤好的面包，这个机器的功能比较单一，接口也很清晰，输入的是面粉和水，输出的是面包。

假设这个机器很多年没有使用，我们现在需要检查一下机器的情况，一种办法，是直接把水和面粉加进去，看它是否可以正常工作，这就是黑盒方法，但是这种情况下，我们没有办法了解机器内部的情况，比如说零件是否生锈，是不是残存了很多脏东西等。

还有一种比较好的办法，是把机器拆开，检查和清洗内部的零件，然后再组装起来，把水和面粉加进去试一试。显然拆开检查，就是白盒方法，再组装回来，就是黑盒方法。

这个例子说明，单纯使用黑盒测试是不够的，还需要进行白盒测试。

白盒测试对代码中，逻辑关系的覆盖更为全面，做得好可以极大地增强产品的健壮性。

我们主要介绍基于控制流的白盒测试技术，控制流图是一个过程或程序的抽象表示，它的节点是语句或语句的一部分，它的边表示语句的控制流。

我们都学过程序流程图，控制流图应该说是流程图的一个简化，它更突出了控制流的结构。

在控制流图里面，我们把所有顺序执行的语句，看成是一个节点，对于判断分支来说，判断条件是一个节点，这个节点根据判断的真和假，产生两个不同的分支，在判断分支执行结束之后，还会有一个合并的节点。

下面我们结合一个例子，来介绍一下如何画出程序的控制流图。

这个程序的功能，是从文件中读取成绩，然后再求出所有成绩的平均值，并打印输出，程序的前三行都是顺序执行语句，在这里，我们把它们看成是一个节点，后面 While 循环，产生一个判断分支，在循环的里面，有一个 if 的判断分支，if 为真时，执行两个顺序语句，if 语句后面是一个读文件的语句，整个 while 循环结束之后，还有一个判断分支，为真执行两个顺序语句，为假执行一个语句，这里 1、4、5、7 和 8，都是可执行语句，也称为基本块。

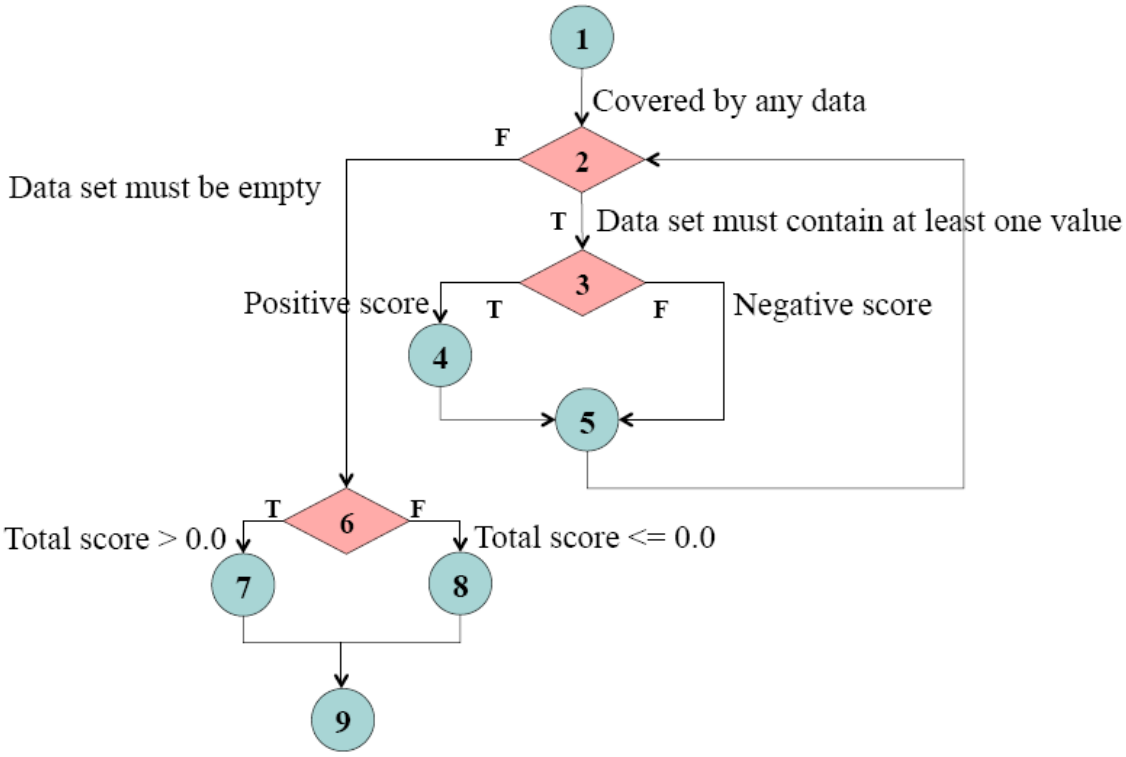


图 5.3 控制流图

根据前面我们对程序的分析，就可以画出如图所示的控制流图如图 5.3 所示，这个图中，三个红色的菱形，是三个分支节点，其余的都是由基本块组成的，基本的计算节点。

基于控制流的白盒测试方法，主要包括以下步骤：

首先、根据程序单元，画出程序的流程图，再用流程图得到控制流图。

然后根据路径选择标准，选择合适的路径，对于所选择的路径，生成相应的测试数据，如果选择的路径可行，那么就得到测试的输入数据，否则继续选择其他路径。

5.3.4 单元测试工具（可选择的）

Python 的单元测试工具和案例，Python 的测试框架 unittest，是受到 unit 启发产生的，并且内置在 Python 中，无需我们安装，unittest 框架，主要有以下几个部分组成，TestCase 类，它是一个最小的测试单元。

我们通过集成这个类，来编写测试用例，一个测试用例，包含若干个以 Test 开头的，相互独立的测试方法，也可以使用 setUp 和 tearDown，来为每一个测试创建销毁。

TestSuite 类，它是测试套件类，包含若干个 TestCase，用于对测试进行分组管理，TestLoader 和 TestResult 类，这两类在编写测试中，我们通常不会直接用到，一些基于 Unittest 工具，可能会用到，main 方法用于启动当前文件中的测试，测试中，我们需要对程序的结果进行判断，来验证结果是否符合我们的预期，及属性断言，例如 assert 关键字就是断言的一种，不过 assert 关键字的功能比较弱，所以 TestCase，它提供了多种强大的断言方法，大家可以在下面的链接中去查阅，这些断言方法，可以在断言的同时，加上一个 message 参数，使断言的意义更加明确，方便我们的用户。

unittest 主要步骤如下：

首先我们需要 import、unittest 包，然后我们继承 unittest 包中的，TestCase 类创建测试用例类，接下来在其中定义 setUp，和 tearDown 方法，在每个测试用例前后，做一些辅助处理之后就可以，开始正式定义测试用例了，所有的测试用例，都是以 test 开头方法，这里需要注意，单元测试和集成测试不一样，它的测试用例，应该只测试一个方面的内容，并且测试目的和测试内容，都应该明确，主要是根据 assertEqoal 等断言，来判断执行结果是否预期，启动测试以后，我们调用 unittest.main 启动测试，运行测试，如果测试未通过，会输出错误提示，如果测试全部通过，则显示 ok，添加-v 参数，可以显示更多详细信息，Python3.3 开始，内置了 Mock 框架，可以方便我们在测试中，用 Mock 对象，替换以下的真实的对象，控制对象的行为，并且记录一代对象，是如何被调用的，它主要有以下几部分组成，Mock 类，用来创建 Mock 对象，当访问 Mock 对象的属性时，Mock 对象会自动创建属性，并且为该属性也创建一个 Mock，MagicMock 类，是 Mock 类的一个子类，区别是它额外定义了操作符，比如大小比较、长度等等。

patch 装饰器，可以把 patch 装饰器，作用在测试方法上，用来限定当前测试中，使用 mock 来替换真实对象，有些用 with 关键字，创建一个作用域，mock 对象提供了一些断言方法，用来我们断言对 mock 项的调用，大家可以在文档中，查看具体的断言方法的说明。

另外我们通常需要从，依赖对象的方法上得到返回值，mock 对象也提供了一些途径，来对返回值进行控制，Python 单元测试覆盖工具，coverage.py，可以通过 pip 来安装，它可以用来记录程序中，每行语句的执行情况，即可以处理覆盖率，覆盖率工具在 PyCharm 上中，可以很方便地使用，主要在运行测试的时候，选择和覆盖率一起来运行，就可以在 PyCharm 项目中看到覆盖率的结果，包括语句的覆盖情况，红色表示没有执行，绿色表示执行，还可以看到语句覆盖率的统计。

下面我们通过一个案例，来学习一下 Python 单元测试，我们首先来分析一下生命游戏，生命游戏它包含四个文件，分别是主程序、输出程序、定时器、还有生命游戏，我们通过 pylint，可以得到以下的依赖关系图，我们可以看到 game_timer，还有 game_map 这两个是，最底层的操作。

在这两个模块当中呢，game_map 又是最核心的模块，它包含了生命游戏的主要逻辑，它有如下的属性方法，rows 和 cols 属性，表示地图的行和列的数量，reset 方法，用一个概率，来设置地图的每个格子的状态，get/set，获得设置某一个格子的状态，get_neighbor_count，获得某一个格子的邻居数量，get_neighbor_count_map，获得每一个格子的邻居数量，set_map 设置整个地图，print_map 打印地图。

下面我们首先来创建测试，我们打开 game_map.py，然后在菜单上选择 NavigateTest，这时会在 game_map 类上，弹出一个小菜单，我们选择 Create、New、Test，这时会弹出创建测试的对话框，我们在对话框中，选择了要测试的方法，这里把所有方法都选中，然后输入测试文件，测试类的名称。

最后点击 ok，这样我们就完成了测试类的创建，在上面它会自动打开测试文件，我们会看到，Test、game_map 这个类，它确实继承了 testCase 类，创建测试之后，我们首先来看一下如何运行测试，我们在 Test、game_map 类的，类名上点击右键，选择运行，test、game_map 中的单元测试。

然后我们就能够在 PyCharm 项目中，看到测试结果，这里一共有 9 个测试，之外的测试用户，这个感叹号在左边的窗口中，来显示，我们可以看到一共失败了 9 个，这是因为新创建的测试方法，它默认调用了，TestCase 的私有方法，而这个方法，

会无条件地使单元测试失败。

下面我们来开始正式编写测试，我们首先来创建 fixture，回顾一下 setUp 方法，它可以用来每个测试，都需要的公共对象，tearDown 方法，用来销毁公共对象，比如数据库，断开连接，关闭文件等等，这里我们只需要 setUp 方法，在其中创建一个，game_map 待测对象，简单起见，这里创建一个，四行三列的 game_map。

然后我们对于 rows 和 cols，进行测试分析出，通过 asserEqual，来判断行为 4、列为 3。

然后我们重新运行测试，可以看到这两个测试通过了，接下来我们来看一下，get 和 set 方法，这两个方法有密切联系，我们把它合并到一个测试当中，这里首先我们断言默认的情况下，每个格子的值应该都是 0，然后我们给 0、0 格子设置为 1，断言 get 应该返回，我们通过 set 设置的值，接下来我们运行测试、通过。

然后是 rest 方法，这个方法它依赖于概率，所以需要我们进行 mock，我们首先看一下，这个方法自己的代码，我们可以发现，它对于每一个行，每一列的一个格子，它都会生成一个随机数，然后判断，是否小于预先给定一个概率值，所以小于的话，就把这个格子设成 1，否则设为 0，我们通过 Patch 方式器，在 TestReset 方法中，把 random 模块的 random 方法，换为 Mock 对象，这个 Mock 对象，我们让它的行为是，返回一个 0.3、0.6，0.9 的循环序列，然后我们调用 Reset 方法，接下来我们断言第一列全部为 1，后两列全部为 0。

然后测试、通过，接下来是，get_neighbor_count 方法，从列表中我们看到，它需要访问 cell 属性，这里我们可以对它进行设置，我们首先将 cell 全部设为 1，这样所有格子的邻居数量，都应该是 8，我们发现测试失败了，对于第一个格子，它的邻居数量返回到 4，而不是 8。

我们检查一下 get_neighbor_count 方法，这里由于 cell 全为 1，所以最后的结果，只和循环次数有关，这样的话循环次数，只跟 DIRECTIONS 有关，我们检查一下 directions，发现这里只写了四个相邻的方向，忽略了角的方向，这就是错了，我们把 directions，修改为正确的值，然后重新运行测试、测试通过，get_neighbor_count_map，它依赖 get_neighbor_count 方法，测试的时候，我们要对依赖方法进行 mock，保持测试的独立性，这样，get_neighbor_count_map 的，正确性就不依赖于，get_neighbor_count 的正确性了，同时我们之后看概率的时候，也不会相互之间有干扰，set_map 它本身比较简单，所以我们这主要测试一下，它对于参数的检查是否完备。

这里我们主要使用了，assertRaises 来判断一个方法，是否抛出了异常，我们看测试通过，我们看到 Print_map 调用了 print，所以我们还是可以通过为 print，进行 mock 来测试，但是这样写本身是不好的，作为一个底层库，这里就要访问一个字符串，我们先设定 cells，然后对 Print 函数进行 mock，print 函数 builtins 包底下，然后我们调用了，assert_has_calls 断言。

这个断言是断言这个方法，是对参数进行了调用，这里的 call 是 mock 包里的一个，代表函数调用的 mock，它的参数表示调用参数，然后我们运行测试、发现通过，到此我们所有的测试都通过了。

在所有的测试通过后，我们就可以进行测试覆盖分析了，我们来运行测试覆盖分析，然后可以看到，84%的行都被覆盖了，查看代码我们发现，没有覆盖的行，全都是这样的类型检查，事实上这些检查，可以通过注解之后，完全通过静态分析工具来检查，所以我们这些不继续测试了。

通过大家有一个误区，就是追求百分之百的覆盖率，但是百分之百的覆盖率，其

实并不能说明非常的正确。所以也不要得到，覆盖率等于正确率，这样的错误结论。

除了以上介绍的工具外，Python，还有很多其他的测试单元工具，其他的测试框架，比如说 Pythonnose，pytest 等等，这里还有一个链接，来比较其他的 Mock 框架，最后这里有一个，Python 测试工具大全的链接，里面几乎包括全部，Python 值的测试框架，Mock 框架、Web 测试工具，GUI 测试工具等等，有兴趣的话可以访问这个网站来自学。

5.4 讨论题

1、黑盒测试和白盒测试问题：

黑盒测试和白盒测试是最主要的两种测试设计技术，请讨论两种测试所针对的测试目标分别是什么？

所发现错误的类型有何差异？

二者各自的优点和缺点是什么？

有了白盒测试为什么还需要黑盒测试？

有了黑盒测试为什么还需要白盒测试？

5.5 测试题

单选题 （10 满分）

1.单元测试内容不包括（ ）。

A.出错处理

B.全局数据结构

C.独立路径

D.模块接口

2.下面的（ ）是错误的。

A.静态测试是不运行被测程序，仅通过检查和阅读等手段来发现程序中的错误

B.动态测试是实际运行被测程序，通过检查运行的结果来发现程序中的错误

C.动态测试可能是黑盒测试，也可能是白盒测试

D.白盒测试是静态测试，黑盒测试是动态测试

3.关于等价类划分，下面的（ ）说法是正确的。

A.等价类划分是将输入域划分成尽可能少的若干子域

B.同一输入域的等价类划分是唯一的

C.用同一等价类中的任意输入对软件进行测试，软件都输出相同的结果

D.对于相同的等价类划分，不同测试人员选取的测试用例集是一样的

4.白盒测试是根据程序的（ ）来设计测试用例。

A.功能

B.性能

C.内部逻辑

D.内部数据

5.关于测试覆盖率，下面的（ ）说法是错误的。

A.测试覆盖率是度量代码质量的一种手段

B.测试覆盖率是度量测试完整性的一种手段
C.测试覆盖率意味着有多少代码经过测试
D.不要盲目地追求100%测试覆盖率

6.在单元测试中，（　）是用来代替被测模块的子模块的。
A.驱动模块
B.桩模块
C.通讯模块
D.代理模块

7.在下面列举的测试覆盖中，（　）是最强的逻辑覆盖准则。
A.语句覆盖
B.条件覆盖
C.判定覆盖
D.条件组合覆盖

8.一个判定中的复合条件表达式为（A>2）or（B≤1），为了达到100%条件覆盖率，至少需要设计（　）测试用例。
A. 1
B.2
C.3
D.4

9.条件覆盖要求（　）。
A.每个判定中每个条件的所有取值至少满足一次
B.每个判定至少取得一次“真”值和一次“假”值
C.每个判定中每个条件的所有可能取值组合至少满足一次
D.每个可执行语句至少执行一次

10.（　）要求每个判定中所有条件的可能取值至少执行一次，而且每个判定的可能结果也至少执行一次。
A.判定覆盖
B.条件覆盖
C.判定条件覆盖
D.条件组合覆盖

第 6 部分 软件系统测试

6.1 本章的教学目标

软件系统测试描述一种用来促进鉴定软件的正确性、完整性、安全性和质量的过程。它是一种实际输出与预期输出间的审核或者比较过程。

通过本章的学习，使学生初识软件测试的基础知识和方法，掌握软件测试概念、测试类型、功能测试和性能测试等内容等。

6.2 本章的教学方法

本章的教学方法采用“翻转课堂式”教学模式，学生可以通过互联网使用“学堂在线”网上优质教育资源，选择由清华大学刘强副教授主讲的“软件工程”课程的“第 15 章 软件系统测试” 课外观看学习，网址：http://www.xuetangx.com/courses/course-v1:TsinghuaX+34100325X+sp/about ，通过视频中教师的讲解，回到课堂上教学交流讨论，并相关的完成习题。

通过理论教学、课堂讨论、适当的布置课外习题作业来完成学习。

6.3 本章的教学内容

软件测试的经典定义是：在规定的条件下对程序进行操作，以发现程序错误，衡量软件质量，并对其是否能满足设计要求进行评估的过程。

本章主要介绍了软件测试的基础知识和方法。通过问题、图表和案例研究，对软件测试问题和技术进行了的研究，主要阐述了软件测试概念、测试类型、功能测试和性能测试等内容等。

6.3.1 软件测试概念

软件质量，一直是人们关注的一个重要问题，在理想的情况下，开发人员编写完代码之后，所有程序都应该能够运行起来，并按照预期的行为进行工作，但是现实往往没有这么美好，即使是那些大公司发布的产品，依然会存在这样那样的问题。

（1）bug 这个词的由来

下面给大家一个例子，1998 年，比尔盖茨在 Win98 新闻发布会上，遭遇了一个情形，在演示 Win98 的 USB 功能时，出现了蓝屏死机，现场颇为尴尬。

软件在投入运行之后，出现问题的事例数不胜数，软件测试作为开发过程的最后一个关键活动，对软件质量保证具有非常重要的作用。

我们经常会听到 bug 这个词，它的原意是臭虫或者虫子，现在经常被人们用于表示隐藏在计算机系统中的一些没有被发现的缺陷或者问题。

这个词的由来，可以追溯到 1945 年，一只飞蛾钻进了计算机电路里，导致系统无法工作。当时格蕾丝·赫柏中尉，把死去的飞蛾贴到在工作日志上，写道就是这个 bug，害得我们今天的工作无法完成，从此，人们把计算机的错误戏称为 Bug，而将发现 Bug，并纠正的过程称为 Debug。

（2）软件缺陷的产生

软件开发是由人来实施的，犯错可以说是人的天性，由于认识不足，关注度和

策略等一系列的问题，人们不可避免地就会在需求分析、总体设计、详细设计和编码调试的过程中出现错误，而这些错误就造成了软件产品的缺陷。

我们来分析一个真实的例子，在 1991 年的海湾战争期间，一枚伊拉克飞毛腿导弹，击中了沙特阿拉伯的一个军营，造成了 28 名美国士兵丧生，事后的调查发现，由于程序的计时用系统时钟，用整数表示，而系统时钟的 1 换算成秒是 1/10 秒这样寄存器存储，并计算导弹的时间，就会产生一个很微小的误差，但是系统缺少有效的防范措施，当爱国者导弹连续工作了 100 小时之后，导弹的时钟，已经偏差了差不多 1/3 秒，这相等于 600 米的距离误差。

由于这个时间的误差，即使是雷达系统侦查到飞毛腿导弹，而且预计了它的弹道，系统也找不到实际上来袭的导弹，最终导致了悲剧的发生。

通过现实中的一些例子，我们已经看到，软件发生错误时，会对人类的生活，造成或大或小的影响，有的甚至会带来灾难性的后果。

（3）软件测试的定义

软件测试可以使这种风险降低，开发人员通过测试，尽可能地发现隐藏在软件中的缺陷，并且及时地修复它们，从而保证最终交付出，经过严格检验的一个完整产品，而不是那些一触就死机的定时炸弹。

现在大家思考一个问题，什么是软件测试，软件测试的概念可以从正向思维和逆向思维两个方面来理解。

从正向思维来看，软件测试就是试图验证软件能够正常工作的

而逆向思维是假定软件是有缺陷的，通过设计和执行一些测试用例，来证明程序执行错误这样的假设。

按照正向思维的模式，测试需要针对系统的所有功能，逐个验证它们的正确性。

测试用例，通常是使用有效的数据、正确的流程和一些多样化的场景，通过执行这些用例，来证明软件是成功的。

在逆向思维中，测试人员要不断思考开发人员理解上的误区，一些不良的编程习惯，各种边界和无效的输入，系统的一些薄弱环节。通过试图破坏，或者摧毁系统的方式，来查找软件中的问题。

（4）软件测试的目的

软件测试，要从发现错误入手，也就是说，它直接的目标是以最少的人力、物力和时间，找出软件中潜在的各种错误和缺陷，通过修正这些错误和缺陷，来提高软件的质量。

但是发现错误并不是测试的最终目的，测试最终还是要对软件质量进行度量和评估，以验证软件质量，满足客户需求的程度。

当然，在测试过程中，通过分析错误产生的原因，可以发现当前开发过程中存在的问题，从而进行软件过程的改进.

（5）测试的局限性

测试是软件质量保证的一种有效手段，但是测试本身也存在一些局限性。

首先，测试是不彻底的，它只能证明错误的存在，而不能证明错误是不存在的，所以，软件经过测试之后，也不能保证没有缺陷和错误。

其次，测试是不完备的，我们不可能使用穷举的方法，把所有输入和程序分支都测试到，所以，设计好的测试用例是非常关键的。

另外，测试并不能直接提高软件质量，软件质量的提高还是要依靠开发，但是没有测试也是万万不行的。

我们通过测试来发现缺陷，进一步地促进修正缺陷，从而间接地起到提高软件

质量的作用。

尽早地介入，是软件测试提倡的一个基本原则，我们在软件开发的各个阶段，都有可能引入错误，而且前期阶段存在的缺陷，会随着软件开发过程的推进，而不断地放大，缺陷发现和修复得越早，所花费的成本也越低。

早期的缺陷，在开发后期或者投入运行之后，再被发现和修复，成本有可能会扩大到几十倍或者上百倍。

（6）缺陷的聚集性

软件错误具有聚集性，它也符合二八原则，即 80%的缺陷，常常存在于 20%的代码中，如果要使软件测试更加有效，就要记住常常光临一些缺陷的高危多发地段，在那里发现缺陷的可能性就会大得多。

软件测试用例具有杀虫剂效应，我们知道给果树喷洒农药时间长了，虫子就会对杀虫剂产生耐药性，测试用例也需要定期评审和修改，同时，还要不断地增加新的测试用例，这样才能发挥更大的效力。

前面说过，软件测试是通过寻找缺陷来提高软件的质量，那么测试需要重点培养，什么样的思考问题的方法，这里列出了测试人员应该具备的不同思维模式。

其中最重要的，就是逆向思维和发散思维

逆向思维，强调的是按照与常规思路，相反的方向进行思考，通常开发人员运用正向的思维去进行软件开发,所以逆向思维可以发现更多,开发人员思维上的漏洞。

发散思维是一种寻求多种答案，最终使问题获得解决的思考方法，这种思维方式可以开阔眼界，活跃思想，用比较独特的视角来看待软件。

不同的思维方式各自有不同的特点，我们在平时也会有意识或无意识地应用这些思维，只有不断地学习、实践和思考，才能做好软件测试的工作。

6.3.2 软件测试类型

软件测试过程通常包括，计划、准备、执行和报告四个部分。

首先，我们要明确测试的任务和方法，制定合理的测试方案和计划，然后，根据任务需要，组织测试团队，并且设计测试用例，还要开发测试工具，或者测试脚本，准备测试的环境和数据，之后，对所提交的软件系统进行测试，记录测试结果，并且跟踪和管理所发现的缺陷，最后，对测试结果进行分析，评价产品的质量，提交最终的测试报告。

软件测试过程中，需要执行不同类型的测试，它们的侧重点各有不同，在执行系统测试之前，开发人员需要通过单元测试来保证单元模块的质量，然后把各个单元模块集成在一起，通过集成测试，得到一个集成后的系统。

系统测试，需要先执行功能测试，来验证系统功是不是正常，功能测试之后，再进行性能测试，以保证系统能够满足规定的非功能需求。

这时我们得到的，是一个可以发布的系统，由于开发人员对软件的需求，可能和客户的实际需求有所不同，所以我们要通过验收测试，来保证所构建的系统，就是客户想要的。

最后 还要在用户的实际环境中，进行安装测试，以保证系统在用户环境下不出现问题。

（1）软件测试类型

软件测试可以从不同的角度，划分成不同的类型，从测试对象来说，包括：单元测试、集成测试、系统测试、验收测试等。

系统测试又分成功能测试和非功能测试。

从测试技术来讲，可以分成黑盒测试和白盒测试两种类型。

从是否运行程序来说，可以分成静态测试和动态测试。

从执行测试的方式来说，又分成手工测试和自动化测试。

下面我们先简单介绍一下，这些测试的基本内容。

单元测试，是对软件系统的基本模块进行测试，以保证单元模块的功能是正确的，一般来说单元测试，是由编写代码的开发人员，自己来执行，正如俗话所说，“一屋不扫，何以扫天下”，如果开发人员，对自己的代码都不做检查，那么更谈不上整个产品的质量保证。

集成测试，是在单元测试的基础上，把所有的模块按照总体设计的要求，组装成为一个子系统或者整个系统所进行的一种测试，一般来说，不同的模块单元是由多个开发人员，并行进行开发，虽然每个模块都通过了单元测试，但是并不能保证这些模块可以正确地组装在一起，对于小规模的子系统，或者系统来说，可以采用一次性的集成方式，就是把所有单元一次性地组装在一起。

对于大型系统来说，通常会采用渐进式的集成方式，就是先组装一部分单元进行测试，之后再增加更多的单元，最终完成所有单元的集成。集成测试的主要目的，是发现不同单元模块之间，接口设计和实现上的问题，

在整个系统集成之后，就可以开始执行系统测试，功能测试和性能测试是系统测试的主要内容。

一般情况下，功能测试是软件测试中工作量最大的一项测试工作，它是从用户的角度，对功能进行验证，以确保每一个功能能够正常的使用。

功能测试主要结合界面、数据、操作、逻辑和接口等方面，来检查系统的功能是不是正确。

比如说 要测试在当当网，搜索并选购图书的功能，我们需要确认每一项功能是符合用户的实际要求，系统的界面清晰、美观、菜单、按钮的操作正常、灵活，除了接受正确的数据输入，对异常数据的输入，要有提示和容错处理，输出结果准确，功能的逻辑清晰，符合使用者的习惯，系统的各种状态按照业务流程而变化，并且保持稳定，要能够配合多种外部的硬件设备和外部其它的应用系统。比如说，支付系统的接口，正确、有效。

性能测试，是在实际或者模拟实际的运行环境下，针对软件的非功能特性，所进行的测试，常见的测试类型，包括容量测试、压力测试、兼容性测试、安全性测试，负载测试等。

容量测试，是在大数据量的环境下对系统进行测试，以便发现其中存在的问题。

压力测试是在强负载，比如说大量并发用户情况下，对系统进行测试来检查系统在峰值使用情况下的操作行为。

兼容性测试，是测试系统在特定的硬件平台上，不同的应用软件之间，不同操作系统平台，以及不同网络环境下，是不是能够正常的运行。

安全性测试，要确保系统满足一些安全性需求。

负载测试，是通过在资源超负荷情况下测试系统的表现，来发现设计上的错误，或者验证系统的负载能力。

（2）验收测试

验收测试，是在软件产品完成系统测试之后，产品发布之前，所进行的测试活动，它的其目的是验证软件的功能和性能，是否能够满足用户的期望。

验收测试一般又包括：α 测试和 β 测试。

α 测试，是软件公司内部的人员来模拟各类用户，对 α 版本的产品进行测试。

β测试，是公司组织一些典型的用户，在日常工作中实际使用β版本，然后用户来报告异常的情况，或提出批评的意见。

（3）安装测试

安装测试，是验收测试之后，把系统在目标环境中进行安装，来检查是不是能够成功安装，并正常地启动使用。

安装测试要考虑目标环境、配置信息以及它对其它应用的影响等一系列的事项。

（4）黑盒测试和白盒测试

黑盒测试和白盒测试是两种不同的测试方法。

黑盒测试，主要是针对界面和功能进行测试，来检查这些功能是不是可以正常使用。

白盒测试，主要是对程序的内部逻辑或者结构进行仔细地检查，包括：逻辑覆盖、基本路径测试等等。

（5）静态测试和动态测试

从测试是否运行程序来区分，又可以分成静态测试和动态测试。

所谓静态测试，就是不实际运行被测软件，而只是对程序代码、界面或者文档进行静态地检查，这种检查可以是非正式的走查，也可以是正式的审查会议。实践证明代码走查是一项非常有效的，查找代码问题的方法

动态测试指的是，实际运行被测程序，通过输入相应的测试数据，来检查实际输出结果和预期结果是否相符。

（6）手工测试和自动化测试

软件测试可以是手工执行，也可以是自动化执行。

在手工方式下，测试人员根据测试大纲中，所描述的测试步骤和方法，手工地输入测试数据，并记录测试结果。

而自动化测试，是使用软件测试工具或者测试脚本，按照测试人员预先设计好的思路，自动地执行。

一般来说手工测试，比较适合业务逻辑测试，特别是一些比较复杂的业务逻辑，而自动化测试，效率更高而且可重复性强，可以把测试人员从重复的劳动中解脱出来，还有一些手工，没有办法执行的测试，比如说，要模拟大量用户，需要大量数据的一些性能测试，都是要进行自动化来执行。

（7）测试文档

在测试之前，需要对测试工作进行计划和准备测试计划是测试工作的，一个指导性文档，它规定测试活动的范围，方法、资源和进度，明确测试内容和任务，以及任务的人员分工等。

测试用例，描述了测试环境、输入数据、执行条件或者步骤以及执行之后的预期结果。

缺陷报告，是记录在测试过程中，发生的任何事件，简单地说 就是记录软件的缺陷。

测试报告，是对测试的过程和结果进行汇总描述，它的核心内容包括两部分，一个是测试结果汇总，要对所测试软件的质量进行评价。

另一个是测试过程的总结，回顾整个测试过程存在的不足，然后进行总结和改进。

测试计划，应该包括的一些主要内容。这是一个测试用例的示例文档，测试的是记事本程序的文件/退出菜单

缺陷报告，一定要详细描述，缺陷发生的基本环境，可重现的执行步骤，错误

现象和简单的问题分析。对于所报告的缺陷，要说明严重性和修复的优先级。

缺陷的严重性，指缺陷对软件产品使用的影响程度，这里把严重性分成四个等级，并且描述了每一个级别，对应的影响程度。

缺陷的优先级，指的是缺陷应该被修复的紧急程度，如图 6.1 所示从高到低列出了四个优先级。

缺陷优先级	描述
立即解决（P1）	缺陷导致系统几乎不能使用或测试不能继续，需要立即修复
高优先级（P2）	缺陷严重，影响测试，需要优先考虑
正常排队（P3）	缺陷需要正常排队等待修复
低优先级（P4）	缺陷可以在开发人员有时间的时候被纠正

图 6.1 从高到低列出了四个优先级。

6.3.3 软件功能测试

功能测试是软件测试中工作量最大的一项工作，它是从用户的角度进行功能验证，以确认每一个功能是否能够正常使用，界面、数据、操作、逻辑和接口这些都是功能测试需要考虑的主要方面。

这里我们简单地列出一些常用的功能测试的检查项目。

功能测试，主要采用黑盒测试方法，它是结合测试内容，对功能进行测试，同时在测试的过程中，对用户需求、设计文档和使用手册进行检查。

设计测试用例，要求有代表性和典型性，尽可能地发现更多的系统弱点，除了正确的输入之外，还要考虑错误或异常的输入，要考虑用户实际使用的各种场景。

测试用例文档主要包括，测试环境、测试步骤、输入数据和预期结果等内容。

前面介绍过等价类分析方法，这是一种典型的黑盒测试方法，它是把程序中所有可能的输入数据，划分成若干个等价类，然后从每个等价类中，选取具有代表性的数据作为测试用例。

测试用例，应该由有效等价类和无效等价类的代表组成。

边界值分析，是对输入或输出的边界值进行测试的一种方法，通常是作为等价类划分法的一个补充，它的测试用例来自于等价类的边界。

场景法，也是一种常用的功能测试方法，主要是根据业务流程，生成一系列业务场景，再对应业务场景设计出相应的测试用例，从而发现需求和实现过程中存在的问题。

现在我们用一个 ATM 取款的例子来说明一下，如何使用场景法设计测试用例。

大家应该对 ATM 取款比较熟悉，正常情况下，用户把银行卡插入 ATM 机中，如果银行卡是合法的，ATM 机就会提示用户，输入银行卡的密码。如果密码正确，而且取款金额符合要求，那么 ATM 机，就点钞并且送出给用户，这样用户就可以成功取款。

设计测试用例：

第一步，是根据需求说明描述出系统的基本流，以及各项备选流，其中基本流，代表正常的业务流程，备选流，代表一些失败或者意外的情况。

在 ATM 取款这个例子中，正常流程就是刚才所说的，成功取款的过程，但是实际

操作时，还有可能出现一些意外情况，比如说，ATM 机器里面没有现金，或者现金不足，银行卡的密码输入有误，以及银行账户的余额不足等，这些都是属于备选流。

第二步，根据基本流和各种备选流，我们可以生成不同的场景，这里列出了 ATM 取款的各种场景，包括成功取款 ATM 没有现金或者现金不足，银行卡的密码有误，一种是还没有达到最大的输入次数，一种是达到了最大的输入次数，账户不存在以及账户余额不足。

第三步，对每一个场景生成相应的测试用例。

最后对生成的所有测试用例进行检查，去除多余的测试用例，在测试用例确定之后，对每一个测试用例选取合适的测试数据，在这里我们给出了 ATM 取款的部分测试用例，如图 6.2 所示。

序号	场景	PIN	账号	取款金额	账面金额	ATM现金	预期结果
1	场景1：成功提款	V	V	V	V	V	成功提款
2	场景2：ATM里没有现金	V	V	V	V	X	提款选项不可用，用例结束
3	场景3：ATM里现金不足	V	V	V	V	X	警告信息，返回基本流相应步骤，重新输入金额
4	场景4：PIN有误（还有不止一次机会）	X	V	n/a	V	V	警告信息，返回基本流相应步骤，重新输入PIN
5	场景4：PIN有误（还有一次机会）	X	V	n/a	V	V	警告信息，返回基本流相应步骤，重新输入PIN
6	场景5：PIN有误（不再有输入机会）	X	V	n/a	V	V	警告信息，卡被没收，用例结束
	.....						

图 6.2 ATM 取款测试用例

错误推测法，也是一种常用的方法，我们可以根据经验或者直觉，推测程序中可能存在的各种错误，从而有针对性地编写检查这些错误的测试用例。

Web 应用系统，涉及到比较复杂的硬件、软件、网络和业务逻辑，相应的测试，也需要考虑更多的环节，一个 Web 应用系统通常需要服务于广大的用户群体，它本身只需要一个服务器端，但是需要通过各种各样的客户端来满足不同用户的要求，它的主要功能是使用 HTML、JavaScript 等语言，来表示文本、图形、音频、视频的内容展现给用户，具有持续演化和及时性的特点，并且要求有良好的感观和一定的安全保障。

Web 应用功能测试，主要是用来检测，Web 应用软件是否实现了预期的功能，对于网页来说，主要包括：内容测试、链接测试、表单测试和 Cookies 测试。

网站测试，是在网页测试的基础上进行，具体包括：特定功能的测试、数据库的测试和设计语言测试等。

下面我们来简单介绍，这些测试的基本含义。

内容测试主要是检测 Web 应用网页提供的信息，具体包括：正确性、准确性和相关性三个方面。

正确性是指信息要是真实可靠，避免出现虚假内容。

准确性，是指网页的文字表述，要符合语法逻辑，没有拼写错误，测试人员需要检查，网页面内容的文字表达是否恰当，样式看起来是否舒服。

相关性，是指能否在当前网页，找到与浏览信息相关的信息列表或者入口，有时网页上的图片，还对应着文字标签，当用户把鼠标移动到图片上的时候，会弹出相关的文字说明。

这是当当网搜索图书的页面，我们进行内容测试时，要检查两个部分。

第一，是对于指定的搜索关键字，比如：软件工程，检查是否列出，含有软件工程名词的图书。

第二是对于图书封面，当鼠标停留或滑过封面的图片时，应该显示相应的商品名称。

网页上的链接，可以使用户从一个页面浏览到另一个页面。

链接测试，主要是检查三个方面的问题。

第一用户点击链接，是否可以顺利打开，所要浏览的内容。

第二所要链接的页面是否存在。

第三要保证系统中，没有孤立的页面，所谓孤立页面，就是没有任何链接指向的页面。

链接测试，可以手动进行，也可以用工具自动检查。

这是某大学网络学堂应用的首页，右下角的两个链接页面就是不存在的。

当用户向 Web 应用系统提交信息的时候，就需要使用表单操作。

表单测试，是模拟表单提交过程，检测其准确性，确保提交信息在整个过程中是正确的。

表单测试主要考虑以下方面：

表单的提交应该模拟用户的提交，以便验证是否完成了预期的功能。

要测试提交操作的完整性，检查提交给服务器的信息，是否正确。

要验证数据的正确性，以及异常处理的能力。

在测试表单时，会涉及到数据校验的问题，需要根据给定的规则，对用户的输入进行校验。

这是一个表单的例子，用户填写个人信息，提交之后，可以申请国航会员，大家可以考虑一下，如何进行测试。

用户在访问网站的时候，Web 服务器会将一小部分信息，存放在客户端的计算机上，Cookie 可以把用户在网站上的输入信息或者是一些选择都记录下来。

当用户再一次访问网站的时候，Web 应用程序就可以直接读取 Cookie 包含的信息。

那么如何对 Cookie 进行测试呢？

这是新浪微博的登录页面，我们在做 cookie 测试时，可能需要检查下面的情况，比如：是否有 Cookie 记录，登陆成功之后，这个记录是不是有问题，Cookie 信息是否进行加密，存放路径是否正确，如果浏览器设置为，不保存 Cookie，那么关闭浏览器之后，检查 Cookie 是否自动删除，要检查 Cookie 的时效，在有效时间里是否自动登陆，到期之后是否自动从本地删除，登陆状态是否自动退出，用户登陆之后，再手动删除 Cookie，或者用户修改密码之后，再次登陆，检查是否退出自动登陆。

测试人员需要对 Web 网站，特定的功能需求进行验证，需要强调的是应该是从用户实际的需要，而不是从产品本身呈现的进行评判。

数据库，是 Web 应用中的重要部分，一般的数据库错误，主要是由于用户提交错误的表单信息，或者程序设计本身的问题引起的。

测试人员，在了解数据库结构和设计内容之后，可以使用破坏性手段或者在并发环境下，来检查数据的完整性和一致性。

设计语言版本的不同，也会引起客户端或服务器端比较严重的问题，所以应用程序要在不同的版本上进行验证。

最后，兼容性测试也非常关键，我们需要测试，不同的操作系统平台、浏览器、显示分辨率，以及打印机等外部设备。

6.3.4 软件性能测试

软件性能，是软件的一种非功能特性，它主要关注的是系统在执行功能时，所表现出来的一些性质，比如：响应是否及时，资源占用的情况如何，是否稳定和安全以及兼容性、可扩展性和可靠性等。

一般来说，我们关注的性能指标，主要包括以下方面：系统对于用户请求的响应时间、容量和数据吞吐量、系统的资源使用率等。

下面我们将分别解释一下这几个指标的具体含义。

性能测试主要是模拟多个用户，使用系统时对性能产生的影响，因此估计用户数是一个基本的要求。

对于 Web 应用来说，用户主要包括：注册用户、在线用户和并发用户三种类型。

注册用户数指的是所有在系统注册的用户数量，但是注册用户有可能并不使用系统。

在线用户数是所有正在访问系统的用户数。

并发用户数是在某一给定时间里，某一个特定时刻进行会话操作的用户数。

通常情况下，我们可以根据网站的统计，得到不同类型用户的数据，如果没有统计数据，也可以通过一定的经验进行预估。

比如：我们大概估计一下可能的注册用户数，然后按照注册数的 20%，估计在线用户数，再按照在线数的 30%，估计并发用户数。

我们以新浪微博的例子来进一步解释一下，用户在注册以后才能够使用，但注册用户，并不是每时每刻都在使用系统。

在线用户在浏览网站的时候，会花很多的时间，阅读网站的信息，所以具体到某一时刻，可能只有一部分在线用户同时向系统发出请求。

由于有些注册用户，可能长时间不登陆网站，所以不能够使用注册用户数，作为性能的指标。一般来说性能测试，更关心的是并发用户数。

响应时间

响应时间是一个重要的性能指标它是从客户端发出请求，到获得响应的整个过程所经历的时间。

这个图显示了从客户端发出请求，到获得响应的整个过程，其中 C1 表示的是用户请求发出前，在客户端需要完成的预处理的时间。N1 是请求由客户端发出，并且达到应用服务器的时间。

A1 是应用服务器对请求进行处理的时间。

如果需要进行数据库的相关操作，N2 是应用服务器把请求发送到数据库服务器，所需要的时间。A2 是数据库服务器对请求进行处理的时间。N3 A3 N4 和 C2 分别是从数据库服务器经过网络、应用服务器到客户端返回处理结果的时间。

对于用户来说，响应时间是上面所有时间的总和。

对于系统来说，响应时间是从应用服务器，接收请求到返回结果的这段时间。

吞吐量和资源利用率是从系统管理的角度，所关心的两个性能指标。

吞吐量是指单位时间里，系统处理客户请求的数量，它反映的是软件系统对性能压力的一个承载能力，一般可以用请求数 / 秒、页面数 / 秒、访问数 / 秒或者处理业务数 / 小时，这样一些单位来衡量。

资源利用率指的是系统资源的使用程度，比如说服务器 CPU 利用率、内存利用率、磁盘利用率、网络带宽利用率等。可以用资源实际使用，占总的资源的比例来表示。

性能测试，是通过自动化的测试工具或手段来模拟多种正常情况、峰值情况以及

异常负载条件，为获取或验证系统性能指标而进行的测试。

负载测试和压力测试，是性能测试的两种主要类型。

负载测试，是通过不断地加载系统负载，比如:，逐渐增加模拟用户的数量，来观察不同负载下，系统的响应时间、数据吞吐量以及系统占用资源的情况，以检验系统的行为和特性，发现系统可能存在的性能问题。

压力测试，是在强负载，比如：大数据量或大量并发用户等条件下，对系统进行测试，来查看系统在峰值使用情况下的操作行为，从而有效地发现系统的某项功能隐患是否具有良好的容错能力和可恢复能力等。

我们来举一个日常生活的例子，来说明二者的区别，如果要测试一个人能够扛多少斤重物。

负载测试，就要确定在 200 斤情况下，这个人能否坚持 5 分钟。

压力测试则是测试，在 200 斤、300 斤、400 斤等不断增重的情况下，这个人的表现，什么时候会失败，失败以后有什么表现，重新再扛 200 斤是否正常。

压力测试也有很多不同的类型。

稳定性测试，是保持在高负载条件下，持续运行 24 小时以上，来查看系统的表现。

破坏性测试，是通过不断地加载，来快速造成系统崩溃，以便让问题尽快暴露出来。

渗入测试，是通过长时间的运行，来发现像内存泄漏，垃圾收集或者一些其他的问题。

峰谷测试，是采用高低突变这样的方式来进行测试，先加载到高水平的负载，然后急剧降低负载，稍微平息一段时间，再加载到高水平的负载，如此反复来查找系统的问题。

另外大数据量测试和疲劳强度测试，也是性能测试的内容。

大数据量测试，主要是检测在数据量比较大的时候系统的性能状况，一般又包括，独立数据量测试和综合数据量测试。

疲劳强度测试，是在系统稳定运行的情况下，通过长时间地运行系统来检测系统无故障稳定运行的能力。

对于一个 Web 应用来说，服务器端、数据库、网络、客户端这些都是影响系统性能的因素，那么应该采用什么样的测试策略来确定系统发生问题的原因。

对于客户端来说，我们可以使用测试工具，在机器上模拟大量的虚拟用户进行并发操作来测试系统。通过综合分析各种指标，最终来评价系统的性能或者发现问题。

对于网络性能，可以使用相应的监控工具，来测试网络的带宽、延时、负载和 TCP 端口变化等因素，对响应时间的影响。

对于服务器性能，测试的关键点是资源占用情况、数据库性能和故障报警。

我们可以使用监控工具，来监测服务器的 CPU 占用、内存消耗和故障情况，检查数据库的读写、缓存、加锁和事务处理等状态。

性能测试，必须要依靠自动化的工具，目前有很多流行的性能测试工具，LoadRunner 是惠普公司开发的一种预测系统行为和性能的负载测试工具，它是使用非常广泛的工业标准级的性能测试工具，可以模拟上千万用户的并发负载来实时监测系统性能，并且确认和查找问题。

jMeter 是目前非常流行的开源压力测试工具，可用于 Web 服务器的性能测试，也可以对数据库、FTP、LDAP 服务器进行测试。

6.4 讨论题

1、软件测试分几种类型？功能测试和性能测试的内容问题。

2、结合网上买过图书，讨论一下场景法测试，试着设计当当网选购图书功能的测试用例。

3、产品质量与软件过程的关系？

有一种观点认为：软件开发中的产品质量与软件过程之间，就像物理学中的“光具有波粒二象性”一样，二者是密不可分，相辅相成的。但也有人认为：强调软件过程管理，会在一定程度上增加管理工作量、降低开发效率、延长开发时间。

请针对上述两类观点给出自己的观点，你认为产品质量和软件过程之间的关系是怎样的？

应该如何辩证的看待软件过程对保证产品质量所起的作用？

6.4 测试题

单选题 （10 满分）

1.软件测试的目的是（ ）。

A.避免软件中出现错误

B.证明软件的正确性

C.解决测试中发现的错误

D.发现软件中潜在的错误

2.下面（ ）说法是错误的。

A.测试应该尽早不断地执行

B.软件错误具有聚集性，对存在错误的部分应重点测试

C.软件测试是提高软件质量的决定性因素

D.测试用例需要定期评审和修改，并且要不断增加新的测试用例

3.下面的（ ）不是集成测试的内容。

A.对软件中最小可测试单元进行检查和验证

B.把各个模块连接在一起时，穿越模块接口的数据是否会丢失

C.一个模块的功能是否会对另一个模块的功能产生不利的影响

D.若干子功能组合在一起是否能产生预期的主功能

4.下面的（ ）是错误的。

A.功能测试是根据需求规格说明验证产品的功能实现是否符合要求

B.压力测试是检测在极限环境中使用系统时施加在用户上的压力

C.安全测试是检测系统中的保护机制是否可以保护系统免受非正常的攻击

D.安装测试是保证应用程序能够被成功地安装

5.（ ）是为了有效地发现软件缺陷而精心设计的少量测试数据。

A.测试计划

B.测试用例

C.缺陷报告
D.测试报告

6.错误推测法是（　）。
A.将输入数据划分成若干个等价类，从中选取有代表性的数据作为测试用例
B.将所有可能的输入数据作为测试用例
C.运用场景对系统的功能点或业务流程进行描述，对应不同的业务场景生成相应的测试用例
D.根据经验或直觉推测程序中可能发生错误的情况，编写检查它们的测试用例

7.Web 链接测试不包括（　）。
A.客户端与服务器端的连接速度
B.无链接指向的页面
C.错误的链接
D.不存在的页面

8.（　）是检测 Web 应用系统提供信息的正确性、准确性和相关性。
A.表单测试
B.链接测试
C.内容测试
D.Cookies 测试

9.下面的（　）不是软件性能的指标。
A.响应时间
B.并发进程数
C.吞吐量
D.资源利用率

10.下面的（　）不是性能测试的目的。
A.达到百分之百的语句覆盖
B.验证软件系统是否能够满足预期的性能要求
C.发现软件系统中存在的性能瓶颈
D.评估软件系统的稳定性和可靠性

第 7 部分 软件交付与维护

7.1 本章的教学目标

通过本章的学习，使学生初识软件维护定义、维护的特点和过程以及软件的可维护性等。掌握软件维护活动是四种类型：纠错性维护（校正性维护）、适应性维护、完善性维护或增强、预防性维护或再工程等。

7.2 本章的教学方法

本章的教学方法采用“翻转课堂式”教学模式，学生可以通过互联网使用“学堂在线”网上优质教育资源，选择由清华大学刘强副教授主讲的“软件工程”课程的“第 16 章 软件交付与维护” 课外观看学习，网址：http://www.xuetangx.com/courses/course-v1:TsinghuaX+34100325X+sp/about ，通过视频中教师的讲解，回到课堂上教学交流讨论，并相关的完成习题。

通过理论教学、课堂讨论、适当的布置课外习题作业来完成学习。

7.3 本章的教学内容

本章主要介绍软件交付和维护的基本概念和方法，主要包括：软件部署与交付和软件演化与维护等。

软件维护（Software maintenance）是一个软件工程名词，是指在软件产品发布后，因修正错误、提升性能或其他属性而进行的软件修改。软件维护主要是指根据需求变化或硬件环境的变化对应用程序进行部分或全部的修改，修改时应充分利用源程序。

7.3.1 软件部署与交付

当两家公司之间进行商业交易时，通常以软件和服务居多。一般需要被委托方公司在一定时间内提交给用户公司软件包或者服务。必须按照用户公司方的要求提交。

当开发进入尾声的时候，项目组的气氛也随着高涨起来，这个时候的主要问题是如何把项目顺利地进行交付。

对于客户委托开发的软件项目，项目的验收和交付工作主要包括：实施、培训和验收三个部分。

验收之后，系统就正式进入维护阶段。

（1）项目实施的任务

项目实施的任务是把软件系统部署到客户的计算机系统上，并且协助客户准备好基础数据，使软件系统能够顺利地上线运行。

首先，要做好系统的测试，保证软件系统的质量符合上线要求。

在实施之前要制定工作计划，确定要发布的代码版本、数据库创建的方式和基础数据的准备。

另外，还要准备好程序代码和相关文档。

在系统部署完成之后，还要组织客户培训，使其掌握软件系统的使用和操作，

培训工作包括选择合适的人员、准备培训内容以及制定培训的计划。

最后，客户对系统进行验收测试，检查用户需求是否实现以及软件的质量是否满足要求。

在验收通过之后，客户将签署验收报告，整个项目正式完成。

（2）软件部署

软件部署是软件开发的一个重要环节，它是通过配置、安装和激活等一系列活动来保证软件制品的投入运行。

部署的技术会影响整个软件的运行效率和投入成本。

软件配置过程也会极大地影响软件部署结果的正确性。

软件部署需要保障软件系统的正常运行，并且使用有效手段提高部署的效率，具体说来，就是要提高软件部署的通用性和灵活性，使其能够适用于更为广泛的软件类型和应用场景，要加强软件部署的可靠性和正确性，实现软件系统的正确配置，优化系统性能，要提高软件部署的自动化程度，尽量减少人工参与，避免人工操作带来的错误。

不同的软件系统具有不同的部署模式,对于像类似 Windows,这种单机软件来说，它的署模式就是一般的安装和更新过程，主要包括：打包、安装、更新、激活。

对于客户基于服务器结构的系统，一种方式是集中式的服务器应用部署，主要适合于小规模用户群的应用环境，一般情况下，这种系统只有一台或几台单独的服务器，每个服务器承担独立的任务，比如： Web 服务器、邮件服务器、数据库服务器等。

另一种方式，是集群式的服务器应用部署，主要适用于并发用户访问量大，而且对系统稳定性和性能要求很高的分布式平台。

在这种模式下，不同的服务器职能是由若干服务器机群来实现，通过负载平衡，把任务分配到集群中的每一台机器上。

在互联网环境下，Web 应用系统的开发，主要是采用敏捷开发方法，强调持续集成和快速交付。

所谓持续集成，就是开发小组经常集成自己的工作，通常每人每天至少集成一次，每一次集成都通过自动化的构建来完成。

持续交付是在持续集成的基础上实现的，它是以自动化或者半自动化的方式，把构建的版本从开发环境推送到接近实际使用的交付准备环境中。

例如像 flickr 系统，大概一周会平均部署几十次，几乎每一个开发人员的每一次修改就会导致一次部署，这样做可以快速地获得用户的反馈，更好地适应需求的变化。

这里显示的是单个产品的构建流水线，从开发人员提交修改到源代码库开始，后续的所有步骤，都是由构建流水线自动来完成。

首先，是 package 打包阶段，它把应用准备好，到能够在实际的环境中进行部署这样的程度，打包之后，可以把产品包部署到一个 staging，也就是预演环境下，准备进行功能测试。

这里要求准备一个干净的预演环境，在环境部署好之后，对环境中的产品运行功能测试脚本，如果测试全部通过，就可以把部署脚本和产品包发布到仓库中。

然后进行 E2E，也就是进行端对端的测试，这里也是要准备好所需要的服务器等设施，然后把集成测试所涉及的所有产品都部署到这个环境中，再运行测试。

如果集成测试也是通过的，那么就可以把产品包部署到实际使用环境中，在传统的瀑布开发过程中，开发人员完成整个代码编写工作之后，测试才正式开始启动。

在测试阶段，开发人员和测试人员一起进行缺陷修复和系统测试，最后通过测试之后，进行上线维护。

在实际开发过程中，会有很多并行项目同时进行，为了避免相互干扰，很多项目都是从主干中拉出分支进行开发。

比如：项目 1 创建一个新的分支进行开发，在开发期间，项目 2 完成开发，合并到主干进行上线，项目 1 完成开发之后，需要先把主干上的程序合并到自己的分支进行联调，联调完成之后进行提测，这个时候测试人员和开发人员一起进行测试和 bug 的修复，测试通过之后，再把项目 1 的分支合并到主干进行上线。

显然，对于大型项目而言，项目的分支可能很多这样整个合并的过程，就非常的复杂，无法做到快速发布。

在敏捷开发过程中，测试和开发从一开始就密切合作，要做到持续地集成和交付，需要对前面的过程进行改进，现在很多互联网公司都是采用主干开发，分支提测的方式，这种方式下所有的开发都在一个主干上进行，然后持续地提交和合并，在达到提测的时候，产生分支进行提测，测试通过之后形成上线的版本，这样改进之后，就可以做到持续快速的进行交付。

7.3.2 软件演化与维护

软件在投入使用之后，由于出现新的需求，商业环境的变化、修复缺陷、软硬件升级以及进一步提升性能等原因，都会导致软件的不断修改，这种变化说明软件具有使用价值，但是修改又会带来副作用，所以关键的问题是需要采取适当的方法，有效地实施和管理这种变化。

IBM 公司的 Lehman 博士，通过对 IBM OS360 系统的研究，提出了关于演化的一系列法则。

系统维护是不可避免的，当系统环境发生改变时，就会产生新的需求，系统就要修改，修改后的系统重新投入使用，又会促进环境的变化，于是进入新一轮的修改循环，随着系统的改变，系统的结构有可能发生退化。

大型系统自身的动态特性，是在开发早期建立的，它决定了系统维护过程中的，一个总趋势和系统变更可能次数的一个极限。

绝大多数大型项目，是在一种饱和状态下进行运作，也就是说资源和人员的变化对系统长期演化的影响是不易察觉的，向系统添加新功能，难免会引入新的缺陷，一个大的功能增量，意味着后面要紧跟着一个版本，来修补系统新引入的缺陷，这个修补版本中，新的功能会比较少，所以一个版本中，如果有大的功能增量，在预算时应该考虑，还需要缺陷的修复。

软件演化包括软件维护和软件再工程两种策略。

软件维护是为了修改软件的缺陷或者增加新的功能而对软件进行修改，这种修改通常只发生在局部，一般不会改变整个结构。

软件再工程是为了避免软件本身的退化，而对软件的一部分进行重新地设计和构造，以便提高软件的可维护性和可靠性。

软件维护，包括改正性维护、适应性维护和完善性维护三种类型。

改正性维护是修改使用之后发现的缺陷。

适应性维护是修改软件，使它适应不同的操作环境。

完善性维护是增加或者修改系统功能，使它适应业务的变化

其中完善性维护的比例最高，大概达到所有维护的 2/3 左右。

软件维护的成本是非常高的，通常是开发成本的 1-4 倍以上。

软件维护工作受到很多因素的影响，大部分情况下负责维护的人员，并不是原来的开发人员，维护人员需要对系统进行理解和掌握，才能做好维护。

开发人员往往不喜欢做软件的维护，感觉没有成就感。

维护人员的技术水平，也会对维护工作产生很大影响。

系统使用越长久也就越难以维护。

比如：很多银行系统，甚至是 20 世纪 60 年代开发的，现在已经很少有人熟悉，当时的编程语言。

一般的软件维护过程，需要分析变更产生的影响，再决定是立即修改，在后续版本修改或者暂时不修改。

变更实现过程也与正常开发过程类似，有时候，会有一些需要紧急处理的情况，比如：出现了严重缺陷需要马上修复，这时就会直接修改源代码进行解决。

当软件的修改造成系统结构受到影响，或者某些部分修改频繁，这个时候，可以考虑用再工程的方法进行维护。

再工程，就是通过再构造和再文档化的方法，使系统更易于维护，它不会改变系统的功能。

再工程需要对现有系统进行理解和转换，具体的方法是首先使用逆向工程，把系统重新文档化，然后进一步修改程序结构，同样也需要对数据进行再工程。

最后得到一个改善了结构的再工程系统。

逆向工程和开发过程相反，它是从源代码得到软件系统的规格说明和设计信息，现在也有很多逆向工程的工具，但是它们在实际应用中存在一些障碍，由于软件缺少形式化的表示方法，加上代码编写不规范或者结构比较混乱，在用工具进行逆向工程的时候，有可能得到的建模信息，没有太大价值，所以，逆向工程仍然是一个在探索的软件工程领域。

7.4 讨论题

1、讨论软件部署和交付的过程。

2、软件维护内容和类型问题讨论。

3、软件再工程的基本思想是什么？需要哪些关键技术？

软件再工程是解决遗留系统存在的问题，对其进行优化和保证其能够继续正常运行的主要手段。那么，软件再工程的主要思想是什么？它同开发新的软件系统在流程、技术和方法等方面的主要差异是什么？其中需要哪些关键技术的支持？请就此进行讨论。

7.5 测试题

单选题 （5 满分）

1.下面的（ ）是错误的。

A.软件交付的主要工作是将程序代码和相关文档交给用户

B.用户培训是帮助用户理解产品并掌握系统的使用和操作

C.软件部署是通过配置、安装和激活等活动保证软件系统的正常运行

D.持续集成是频繁持续地将团队成员的工作进行集成

2.下面的（　）是正确的。

A.只有质量差的软件产品才需要维护

B.软件的维护成本通常比开发成本低

C.软件的不断修改将导致系统结构的恶化

D.重新开发一个新系统通常要比再工程的成本要低

3.（　）是由于计算机软件和硬件环境变化而修改软件的过程。

A.改正性维护

B.适应性维护

C.完善性维护

D.预防性维护

4.下面的（　）不是软件再工程活动。

A.增加新的功能

B.逆向工程

C.程序结构改善

D.数据再工程

5.逆向工程通常用在软件生命周期的（　）阶段，它是从源代码或目标代码中提取设计信息。

A.需求分析

B.软件设计

C.软件测试

D.软件维护

第 8 部分 敏捷开发与配置管理

8.1 本章的教学目标

敏捷开发是针对传统的瀑布开发模式的弊端而产生的一种新的开发模式，目标是提高开发效率和响应能力。除了原则和实践，模式也是很重要的，多研究模式及其应用可以使你更深层次的理解敏捷开发。

配置管理(Configuration Management，CM)是通过技术或行政手段对软件产品及其开发过程和生命周期进行控制、规范的一系列措施。配置管理的目标是记录软件产品的演化过程，确保软件开发者在软件生命周期中各个阶段都能得到精确的产品配置。

通过本章的学习，使学生初识敏捷开发与配置管理的基本知识，了解敏捷开发之 Scrum、用户故事与估算、团队协作工具 Tower、软件配置管理、配置管理工具 Git 等。

8.2 本章的教学方法

本章的教学方法采用“翻转课堂式”教学模式，学生可以通过互联网使用“学堂在线”网上优质教育资源，选择由清华大学刘强副教授主讲的“软件工程”课程的“第 6 章 敏捷开发与配置管理” 课外观看学习，网址：http://www.xuetangx.com/courses/course-v1:TsinghuaX+34100325X+sp/about ，通过视频中教师的讲解，回到课堂上教学交流讨论，并相关的完成习题。

通过理论教学、课堂讨论、适当的布置课外习题作业来完成学习。

8.3 本章的教学内容

本章主要阐述敏捷开发与配置管理的基本知识，包括：敏捷开发之 Scrum、用户故事与估算、团队协作工具 Tower、软件配置管理、配置管理工具 Git 等。

8.3.1 敏捷开发之 Scrum

Scrum 是目前最流行，也是最有效的一种敏捷项目管理方法，已经被许多知名的软件企业所采用。

这个英文单词的意思，是橄榄球运动的一个专业术语，表示争球的动作，用在软件开发上，它象征着开发团队，在开发软件的时候，所有的人都像打橄榄球一样迅速，而富有战斗激情，你争我抢，完成任务，并且通过逐步逼近的方式，取得最后的胜利。

在 Scrum 框架中，整个开发过程是由若干个很短的迭代周期所组成，一个短的迭代周期，称为一个 Sprint。

Scrum 是通过以下过程，来实现产品的迭代开发。

首先产品经理根据用户需求和市场需要，提出一个按照商业价值，进行排序的客户需求列表，在每一个迭代的开始，开发团队要召开迭代计划会议，从这个列表中，挑选出一些优先级最高的条目形成迭代的任务，在迭代开发过程中，要进行每日站立会议，检查每天的进展情况，在迭代结束的时候，就会产生一个可运行的交付版本，由项目的相关人员，参加产品的演示和回顾会议，来决定这个版本是否达到了发布的要求。

一个 Sprint 是指一个 1 到 4 周的迭代，现在很多的团队采用 2 周作为一次迭代，像一些市场变化快，竞争激烈的领域。

比如：互联网或移动互联网领域，有可能使用一周的迭代，Sprint 在整个开发过程中，是保持固定的时间长度，结束的时候，会产生一个完成的可运行的潜在可发布的产品增量。

在一个迭代的执行过程中，它的实现目标、质量和验收标准，都是不允许变化的，除非迭代的目标，失去了价值和意义。

比如：公司的发展方向，或者市场技术等情况发生了变化，这个时候就应该取消迭代，不过由于迭代的周期非常短，这种情况一般也很少发生。

Scrum 框架包括三个部分：开发团队角色、开发制品和开发活动。

首先来看开发团队角色，主要包括产品负责人、Scrum 主管和团队成员三种类型。

产品负责人代表客户，决定产品的需求和优先级，它是开发团队对于需求的唯一诉求人，产品负责人的主要职责包括：确定产品的功能、负责维护产品的需求列表、决定产品的发布日期和发布内容、为产品的投资回报负责、要根据市场价值、确定需求的优先级、在每个迭代开始前调整功能和功能优先级，在迭代结束的时候要验收开发团队的工作成果。

Scrum 主管作为团队领导与产品负责人密切合作，及时为团队成员提供帮助，使得团队免予外界的干扰，它的主要职责是直接管理项目，要知道什么任务已经完成，哪些任务已经开始，哪些新的任务被发现，以及哪些估算可能已经发生变化，要保证开发过程按照计划进行，组织每日站立会议、迭代计划会议、迭代评审会议和迭代回顾会议，要引导团队正确地应用敏捷实践，保证团队资源完全使用和高效运作，要促进团队成员之间的良好合作，解决团队开发中的障碍，作为团队和外部的接口，屏蔽外界对团队成员的干扰。

Scrum 团队负责在每个迭代中，把产品需求转化为潜在的可交付的功能增量，它的特点是，团队的规模控制在 5 到 9 人。

如果小于 5 人，团队的生产力会下降，有可能会受到技能的限制，从而导致无法交付，可发布的产品模块。

如果成员多于 9 人，那么成员之间，就需要太多的协调和沟通，对于大型项目来说，可以采用多个小的 Scrum 团队，每个团队派出代表，进行团队之间的协调和沟通，团队是跨职能的，团队成员必须具备交付产品增量所需要的各种技能，比如说编程、质量控制、业务分析、架构 用户界面设计或数据库设计等。

在 Scrum 团队中，没有头衔的概念，每一个人都必须尽心尽力完成迭代目标，团队成员要全职工作，而且是自我组织和管理的，成员之间协调配合，提高整个开发效率，最终保证每一个迭代的成功。

自组织团队是敏捷软件开发的基本观念，它是指团队被授权自己管理他们的工作过程和进度，并且由团队决定如何完成工作。

具体说来，就是团队自己进行任务分配，自己做技术决策，在确保目标的前提下，制定自己的行为准则，团队要有义务保持过程的透明性，要监督和管理自己的过程和进度。

Scrum 制品主要包括：产品订单，也称产品待办事项，迭代订单和燃尽图。

产品订单是一个从客户价值角度理解的，有优先级顺序的产品功能列表。

产品订单中包含开发和交付成功产品，所需要的所有条件和因素。

产品订单的条目，可以是功能需求，也可以是非功能需求，还可以是主要的技术改进目标，比如：用 java 代替 C++重写系统，以及已知的缺陷。

比如：分析并修复订单处理脚本的错误等等。

迭代订单是团队承诺的，在当前迭代要完成的任务列表，这些任务是通过选取产品订单项，并进一步细化和分解形成的，其目的是将产品订单项转化为潜在的可交付的产品增量。

对于简单情况，可以直接把订单项作为任务分配。

在复杂情况下，就需要进一步把订单项，分解成一系列，更细致的开发任务。

如何选择产品订单项，取决于它的优先级以及开发团队完成开发所需要花费的时间，选择哪些订单项，以及多少项是由开发团队自己来决定。

可工作产品，是迭代开发的产出结果，它是一个可以交付的产品增量，可交付的标准是在迭代初期提前设定的，每一次迭代都应该是一个可运行的版本。

产品负责人负责产品订单的内容、可用性和优先级，产品订单永远都不会是完整的，最初它只是列出一些最基本的和非常明确的需求，这些需求至少要足够一个迭代的开发，随着开发团队，对产品和用户的了解，产品订单在不断演进。

所以产品订单是动态的，它经常发生变化，以保证产品更加合理，更具有竞争性和更有用。

产品订单条目是按照优先级进行排序，优先级主要是由商业价值、风险和必要性来决定，优先级高的条目，需要优先进行开发，优先级越高，条目就越详细，越低就越概括。

使用用户故事，来描述产品订单条目，是一个非常有效的实践。

用户故事是从用户的角度，来描述他所期望得到的功能。

这个表列出了一些，关于网上订购商品的用户故事，这些故事描述了作为顾客、注册、搜索、浏览和购买商品的，一系列功能，以及作为工作人员维护商品信息和查看订单的功能。

对于迭代过程中，所有要完成的任务，可以使用任务板直观地进行展现。

在迭代开发中，任务板是在不断地更新的，如果某个开发人员想到了一个任务，他就可以把这个任务写下来，放在任务板上，在每日站立会议期间或者之后，如果估计发生了变化，任务会根据变化，在任务板上进行相应的调整，任务板有电子和实物两种形式，对于远程开发团队来说，电子版是一个有效的方式，对于集中在一起，面对面工作的团队来说，实物板更利于沟通。

燃尽图是以图形方式显示迭代过程中，累计剩余的工作量，它是一个反映工作量完成状况的趋势图，其中 Y 轴代表的是剩余工作量，X 轴代表的是迭代的工作日。

在迭代开始的时候，开发团队会标识和估计，这一个迭代需要完成的详细任务，所有这一个迭代中需要完成，但是没有完成的任务工作量是累计的工作量。

Scrum 主管会根据进展的情况，每天更新累计工作量，如果在迭代结束时，累计工作量降低到 0，迭代就成功地结束。

现在图上显示了三条曲线，绿色线代表这个团队计划良好，所有的故事都已经完成，接近理想情况。

紫色的线表明该团队，也完成了目标，但是有可能没有主动地，去更新数字，也有可能是前期有点拖延，后期比较赶工。

蓝色线表明该团队执行不太理想，在迭代时间截止时，并没有完成所有任务。

Scrum 是通过以下活动，来管理整个项目开发过程的。

迭代计划会议，是计划即将开始的迭代开发，每日站立会议用来检查迭代过程的工作进展。

迭代评审会议，用来检验所发布的工作成果。

迭代回顾会议是回顾和总结，已经完成的迭代过程。

Scrum 项目包括两级规划，一个是发布规划，它对整个产品发布过程进行展望，定义用户故事，并进行优先级划分，估算开发的规模，以及评估开发团队的速度，制定出整个产品的发布计划。另一个是迭代规划，只对一次迭代进行展望，确定迭代的目标，并且选择用户故事，把故事分解和细化成任务，并进行时间估算。

迭代计划会议，是在每次迭代开始时召开，它的目的，是选择和估算本次迭代的工作项，整个会议分成两个部分。

第一部分，是以需求分析为主，确定到底要做什么，也就是从产品订单中，选择和排序本次迭代需要实现的订单条目。

第二部分，是以设计为主，确定要怎么做，由开发团队决定系统的设计方案和工作内容。

迭代计划会议需要整个团队参加，产品负责人逐条讲解最重要的产品功能，开发团队共同来估算，故事所需要的工作量，直到本次迭代的工作量达到饱和为止。

产品负责人参与讨论，并回答与需求相关的问题，但是并不干涉估算的结果。

在迭代开发过程中，开发团队通过每日站立会议，来确认他们的工作进展情况，评估是否可以实现迭代的目标，这个会议每天在同样的时间和同样的地点召开。

每个团队成员，都需要回答三个问题：

上次例会之后完成了什么，遇到了什么困难，下次例会之前计划要做什么。

每日站立会议中，可能有简要的问题澄清和回答，但是不应该有任何话题的讨论，讨论可以下去单独进行，这个会议并不是向任何人做汇报，而是开发团队内部的一个沟通会议，来帮助大家快速地发现问题，促进团队的自组织和自管理。

每日站立会议要求简短，通常不超过 15 分钟。

迭代评审会议，是在迭代结束的时候召开，开发团队和相关人员一起来评审迭代产出的结果。

一般情况下，开发团队会演示产品的增量，让用户代表尝试使用这些新功能，来听取用户对产品功能的一个反馈，整个小组也会讨论，他们在迭代中观察到了什么，有哪些新的产品想法，产品负责人会对未来，做出一个最终的决定，并适当地调整产品订单条目。

在每次迭代完成之后，还要举行一个迭代总结会，会上所有团队成员，都要反思这个迭代过程，要识别出哪些做得好，哪些做得不好，找出潜在的可以改进的事项，为将来的改进制定计划。

需要注意的是要抓重点问题，每次也就是 1-3 个关键问题，做出可行的解决方案，这里可行的含义，是方法简单、影响范围小、见效快，再一个目标不要太激进，要现实可行、积少成多。

对于参加课程实验项目的团队，希望能够尝试着使用这种方法，来管理自己的项目，体会和掌握敏捷开发管理方法。

8.3.2 用户故事与估算

在敏捷项目中，用户故事，是描述产品需求的一种常见方法，所谓用户故事是从用户的角度，来描述它所需要的功能。

一般情况下，我们是把故事用很简短的语言写在一张卡片上，这里列出了一种常见的故事表达格式，它包括三个部分。

第一是角色，谁要使用这个功能。

第二是活动，需要完成什么样的功能。

第三是价值，为什么需要这个功能，这个功能可以给用户带来什么价值或者好处。

一个好的用户故事，应该具备独立性、可协商、有价值、可估算、短小和可测试六个方面的特点。

故事之间要保持独立性，尽可能地避免存在依赖，否则的话，就会使制定计划、确定优先级和工作量估算都变得很困难，故事的内容是可以协商的，一个故事卡片上，只是对故事的一个简短的描述，不应该包括太多的细节，具体的细节应该是在沟通的过程中产生的。

每一个故事必须对客户是有价值的，没有价值的故事，就不应该纳入开发范围中。

开发团队需要对故事的规模进行估计，以便确定开发的工作量，安排开发计划。

故事的开发工作量要尽量的短小，最好不要超过 10 个理想人天，至少要保证是在一个迭代中，能够完成。

故事必须也是要可测试的，这样才可以确认它是可以完成的。

我们在故事卡片的正面，写上故事的内容，比如：顾客可以使用信用卡，购买购物车中的商品，并且注明所接受的三种信用卡，在卡片的反面可以列出，对这个故事的一些测试项，包括三种信用卡、借记卡、某种会员卡、各种卡号、失效的卡以及不同的限额等。

用户故事是要面向价值，进行编写的，例如：这是一个网络游戏，排行榜功能的用户故事，作为一个玩家可以通过显示排名，以便让自己在服务器中的地位，获得认可。

表面上看，这个故事好像没有什么问题，但是仔细分析可以发现，虽然这个功能可以激发玩家的斗志，鼓励购买道具，但是实现起来却有技术的问题。

如果玩家太多，实时查看排名是很不现实的，另一个问题是，小的玩家其实对自己的排名并不关心，即使关心，也不会为了提升排名去购买道具，只有少数的顶级玩家，才会真正地受此诱惑。

所以这个功能，就被改为系统每周重新排名一次，而且只显示前多少名玩家，那么我们把故事就写成，作为一个排名靠前的付费玩家，可以通过显示排名，以便让自己在服务器中的地位获得认可，以刺激消费。

通常情况下，我们主要使用故事来描述产品功能，这里有三个故事的例子。

第一个是维基用户，希望上传文件到维基网站。

第二个，是客服代表希望记录客户问题。

第三个，是网站的管理员需要统计，访问网站的人数。

这三个故事，都采用了前面的格式进行描述，而且明确地给出了具体的用户角色，需要特别说明的是，我们不要把故事的角色总是写成，作为一个用户之类的含糊的说法，而是要把用户区别对待，这样才能更好地理解，他们使用什么功能，如何使用以及为何使用。

另外故事除了用于描述功能之外，有时候也用于描述非功能需求，技术增强和缺陷修复等。

比如：这个故事描述的就是系统，对浏览器支持的非功能需求。

开发人员有时也会创建一个技术增强的故事。

比如：这里给出的这个例子，就是开发团队希望评估分析，两种可行的过滤引擎的技术架构，并且建议要进行性能测试、规模测试和类型测试，然后用一个简单的备忘录来说明所进行的实验取得的结果和下一步开发的建议等。这个故事则是描述了，需要修复的缺陷信息。

产品订单，是一系列用户故事的列表，这些故事描述的是一些粗粒度的功能，它

是由产品负责人负责维护，并根据市场价值来决定故事的优先级。

不过这个列表并不是一成不变的，需要在每次迭代之前，对功能和功能的优先级进行调整。

在每次迭代开始，我们通过迭代计划会议，挑选出需要开发的订单故事，再进一步细化成开发任务，最后由开发团队进行实现。

这个表是一个关于网上购物的产品订单示例，每个条目是一个用户故事，描述了用户需要的功能，所有的条目按照优先级进行排列。

这里是某一个产品的需求列表，那么需要多长时间才能完成版本 1 的开发，要回答这个问题，首先需要估算出版本 1 的工作量，具体的做法是。

估算每一个故事的工作量，然后把所有的估算值相加，得到一个总的估算值。

接着需要给出团队的开发速度，我们可以使用历史值，或者做一个猜测，也可以试着做一轮进行测算。

这个图显示了前 5 次迭代的开发速度，通过计算就可以得到一个平均速率，测算的时候，只是统计每一个迭代，所有已经完成的故事，没有完成的不计算在内。

最后我们就可以估计出，版本 1 的迭代次数。

敏捷估算有两种度量单位，用以表达用户故事、功能或者其他工作的总体规模，一个是故事点，它是一个相对度量单位，使用的时候，我们给每个故事分配一个点值，点值本身并不重要，重要的是不同故事点值的相对大小。

另一个是理想时间，它是一个绝对度量单位，可以是天、小时等。

理想时间是指某件事，在剔除所有外围活动以后，所需要的时间。

一般按照一天有效工作时间的 60%-80%计算比较合理。

理想时间的估算方法，是团队成员分析和讨论，故事的细节和复杂性，然后估算出完成故事开发，所需要的理想时间。

这种方法是我们平时习惯使用的，容易理解和掌握，但是做绝对的估计，是人们天生不擅长的，而且每个人的估算不同，很难达成共识。

另外，每个人的理想时间也是不一样的，显然，无法把估算的结果进行相加，所以这样产生的计划，肯定是不准确的。

故事点的基本做法，是先找出一些标准的故事，设定一个标准点数，形成比较基线，其它故事和标准故事进行比较，给出一个相对的比例，从而得到这个故事的一个估计值。

这种方法的难点在于，故事点的产品特征很明显，没有办法在不同团队之间进行比较，如果没有历史数据，很难设定标准故事。

对于网上购物这个订单来说，我们首先选择一个，简单的标准故事、注册帐户，把它的规模设定为 1，然后其它故事和标准故事进行比较，可以得到相应的故事点数。

敏捷估算是由开发团队共同完成，产品负责人和 Scrum 主管并不参与实际的估算，产品负责人只是阐述和澄清用户故事。

Scrum 主管是指导和引导，整个估算的过程，估算不是承诺，如果我们用估算的结果，来评价一个成员的工作是否按时完成，那么它就会在原有估算的基础上，加入一个安全量，从而人为地放大估算值，估算应该准确，但是没有必要过于精确，我们应该投入刚好够用的工作量，得到一个大致正确，足够好的估算，过于精确的估算也是浪费。

由于人们更擅长相对的估算，所以应该使用相对大小，而不是绝对大小进行估算。

我们可能对于一个玻璃杯到底能放入多少饮料没有什么概念，但是说出一个玻璃杯，相对于另一个多大比较容易。

开发团队可以使用敏捷估算扑克来进行估算，它是一种基于共识的，估算工作量的技术。

在扑克牌上，印有一些估算值，一般是有三种形式给出。

一个是自然序列，一个是斐波纳契数列，还有一个是不连续的自然数。

比如：2 的幂等，选用哪一种形式，是由开发团队来决定的。

用敏捷估算扑克进行估算，主要包括以下步骤。

首先是分牌，敏捷扑克和普遍游戏扑克一样，都有 54 张牌，拥有 4 种花色，每一种各 13 张，每一名参与估算的成员会得到相同花色的一组牌，扑克牌的正面，印有估算的数字，以斐波纳契数为例，0 代表故事已经完成，或者太小没有估算的意义，1/2 代表微小的故事，1、2 和 3 代表小的故事，5、8 和 13 代表中等大小的故事，20 和 40 代表大的故事，100 代表非常大的故事，问号代表估算的成员对故事不理解，或者不知道怎么来进行估算。

接下来产品负责人，从订单中选择一个故事，为大家进行详细讲解，团队成员讨论并且提问，产品负责人解答大家的问题。

当团队成员确认已经对故事完全了解，而且没有重大问题之后，大家开始进行估算，所有成员都选出代表自己估算值的纸牌，然后扣在桌面上，选完之后大家同时亮牌，如果每张牌的估算值存在很大差异，代表大家对这个故事，没有达到共识，就需要对评估结果进行讨论。

在对故事进一步了解之后，再重新进行估算，直到团队成员之间的估算值，达到一致。

一般情况下，最多三轮就可以达到统一，如果三轮之后，依然没有达到统一的意见，那么 Scrum 主管就要立即中断这个估算，或者取平均值，或者取一个大家共同接受的值，作为估算的结果。

8.3.3 软件配置管理

软件开发过程中，经常需要修改代码，大家是否遇到过下面的问题.

有的时候代码改乱了，想要返回到之前的某个版本，但是却没有保存.

调试了半天代码，最后发现问题的原因，是代码的版本不对.

在没有经过允许的情况下，开发人员擅自修改了代码，结果造成系统出现新的问题，由于代码管理比较混乱，在人员流动时，发现有的代码并没有交接.

在维护过程中，可能需要重新编译某个历史版本，但是因为缺少原有的开发工具，或者运行环境，造成无法重新编译。

在协同工作过程中，代码修改混乱，比如说，自己新改好的代码，又被别人覆盖为旧的版本。

软件项目开发过程面临的一个主要问题，就是持续不断地变化，变化可以导致开发的混乱。

软件配置管理，就是用于管理和控制，变化的有效手段。

软件配置管理是一种标识、组织和控制修改的技术，它的目的是使错误达到最小，并且能够有效地提高开发效率。

软件配置管理的作用，是记录软件产品的演化过程，确保开发人员在软件生命周期的每个阶段都可以获得精确地产品配置，最终保证软件产品的完整性、一致性和可追溯性。

下面我们来介绍，软件配置管理的相关概念。

软件配置项，是软件配置管理的对象，它是可以作为单独实体来处理的，一个工

作产品或者软件。

常见的软件配置项，包括：文档数据、源代码和目标代码，还有一些构造软件的工具和运行环境，也常常列入配置管理的范围。

版本是在明确定义的时间点上，某个配置项的状态。

版本管理，是软件配置管理的核心内容，它是对系统的不同版本进行标识和跟踪的过程，从而保证，软件技术状态的一致性。

一个配置项，比如：某一个源程序文件，可以有一个主干版本，像 1.0、1.1、1.2、2.0 等，也可以从某一个版本的状态，产生若干个分支。分支版本也可以合并到主干上面。

基线，是软件配置项的一个稳定版本，它是进一步开发的基础，成为基线的配置项，只有通过正式地变更控制过程，才能修改，在迭代开发过程中，一次迭代包括了计划，需求、设计、实现、测试等不同的阶段。

基线一般是标志着，一个开发阶段的结束，比如说，需求文档经过正式评审之后，就成为一个基线。

形成基线的文档，需要经过变更申请和批准之后，才能修改。

在一次迭代结束后，就形成一个软件开发的里程碑。

在软件开发过程中，程序员修改代码可能会出现两种情况。

第一种情况，每个程序员各自负责不同的模块，但是他们对源程序文件的修改，完全不存在交集，这个时候每一个人都可以直接从代码库读取文件，修改之后再存入代码库中。

第二种情况，可能有两个程序员同时修改同一个代码文件，这个时候就会出现代码覆盖的问题，比如说 Joe 和 Jane，从代码库中都读取了同一个文件，然后分别进行修改，Joe 先修改完，并且把修改的程序文件存入代码库中，等 Jane 修改完存入的时候，如果不加控制就会意外地覆盖了 Joe 原来改过的部分。

解决这种问题，有两种不同的方法。

第一种是独占工作模式，就是一个程序员在修改代码时，将独自占有代码文件，不允许其他程序员修改，当 Joe 读取文件之后，系统会对文件进行锁定，这个时候，如果 Jane 想修改文件，系统就不允许锁定的文件再被修改。

在 Joe 修改完成之后，把文件写入到代码库中，系统会对其进行解锁，这个时候，Jane 才可以从代码库中，获取最新版本进行修改，系统会再进行加锁控制。

第二种方法是并行工作模式，它支持多个人共同修改同一个文件。

首先，Joe 和 Jane 同时读取一个文件，然后分别开始编辑文件，Joe 先修改完之后，把文件写入代码库中，这个时候 Jane 再写入文件时，系统就会提示，文件的版本已经被更新。

Jane 需要将代码库里的最新文件取出来和自己的文件进行比较，然后把自己的修改合并进去，Jane 将合并后的文件写入代码库中，之后再从代码库读取的文件，就是合并后的新的版本。

代码分支，是支持软件并行开发的常用机制，它实现了一个项目的文件树及其发展历史。

首先，代码库应该有一个而且只有一个主分支，所有提供给用户使用的正式版本，都在主分支上发布，主分支只用来发布重大的版本，日常的开发应该在另外的分支上完成。

如果开发分支上的文件，需要正式对外发布，就在主分支上对开发的分支进行合并，除了主干和日常开发分支之外，还有一些临时性的分支，用于应对一些特定目的

的版本开发，比如说，功能分支或者缺陷修复的分支等。

需要注意的是每一个分支的目的和用途都要明确，运用分支可以实现，多人并行开发一个新的系统，同步地更改多个并行版本的错误，以及同时集成和发布多个版本。

软件配置管理，有很多成熟的工具，像 IBM 公司的 ClearCase，微软公司的 Sourcesafe，还有一些开源的工具，其中 SVN 和 Git 是两种流行的开源版本控制工具。

Git 是一个开源的分布式版本控制系统，具有处理速度快，分支和合并表现出色的特点，Github 是一个基于 Git 的开源项目托管库。目前已经成为全球最大的，开源社交编程以及代码托管网站，在这门课中，我们推荐大家学习和掌握 Git 工具，学会在 Github 上管理自己的代码。

8.4 测试题

单选题 (10 满分)

1.敏捷开发方法通过（ ）管理不可预测性。

A.非常仔细地收集和定义需求
B.制定详细的开发计划
C.软件增量必须在较短周期内发布
D.软件过程必须逐渐适应变化
E.选项 A 和 B
F.选项 C 和 D

2.关于 Sprint，下面的（ ）是错误的。

A.一个 Sprint 通常是一个 1-4 周的迭代
B.Sprint 长度在开发过程中是可以调整的
C.需求在一个 Sprint 中是不允许变化的
D.Sprint 的产出是“完成”的、可用的、潜在可发布的产品增量

3.在每日站立会议上，下面（ ）不是每个团队成员需要回答的主要问题。

A.从上次 Scrum 站立会议后你做了什么？
B.你遇到哪些障碍或困难？
C.你所遇到问题的原因是什么？
D.你打算到下次 Scrum 站立会议完成什么？

4.下面的（ ）不属于产品负责人（Product Owner）的职责范围。

A.组织每日站立会议
B.定义产品需求
C.确定需求优先级
D.验收迭代结果
E.负责产品的投资回报

5.在敏捷开发方法中，用户故事（User Story）的作用是（ ）。

A.定义需要发布给最终用户的软件特性和功能
B.确定发布每一次增量的日程表
C.用于代替详细的活动计划
D.用于估算构建当前增量所需要的努力
E.选项 A 和 C

F.选项 A 和 D

6.下面的（ ）是正确的。

A.故事点是一个绝对度量单位

B.故事点估算一定要做到非常精确

C.故事点表示开发一个用户故事或特性的复杂度

D.故事点表示开发一个用户故事或特性所要付出的工作量

7.软件配置管理的目的是（ ）。

A.降低开发成本

B.控制软件修改

C.减少混乱

D.提高软件开发效率

E.提高正确率

8.下面的（ ）是有效的软件配置项。

A.软件工具

B.文档

C.可执行程序

D.测试数据

E.以上所有选项

9.在使用 Git 进行代码文件提交时，如果提示提交内容为空、不能提交，则最为合适的处理方式是（ ）。

A.执行 git status 查看状态，再执行 git add 命令选择要提交的文件，然后提交。

B.执行 git commit --allow-empty，允许空提交。

C.执行 git commit -a ，提交所有改动。

D.执行 git commit --amend 进行修补提交。

10.如果项目中文件 hello.c 的内容被破坏，执行（ ）使其还原至原始版本。

A.git reset -- hello.c

B.git checkout HEAD -- hello.c

C.git revert hello.c

D.git update hello.c

第 9 部分 微信抢票应用案例

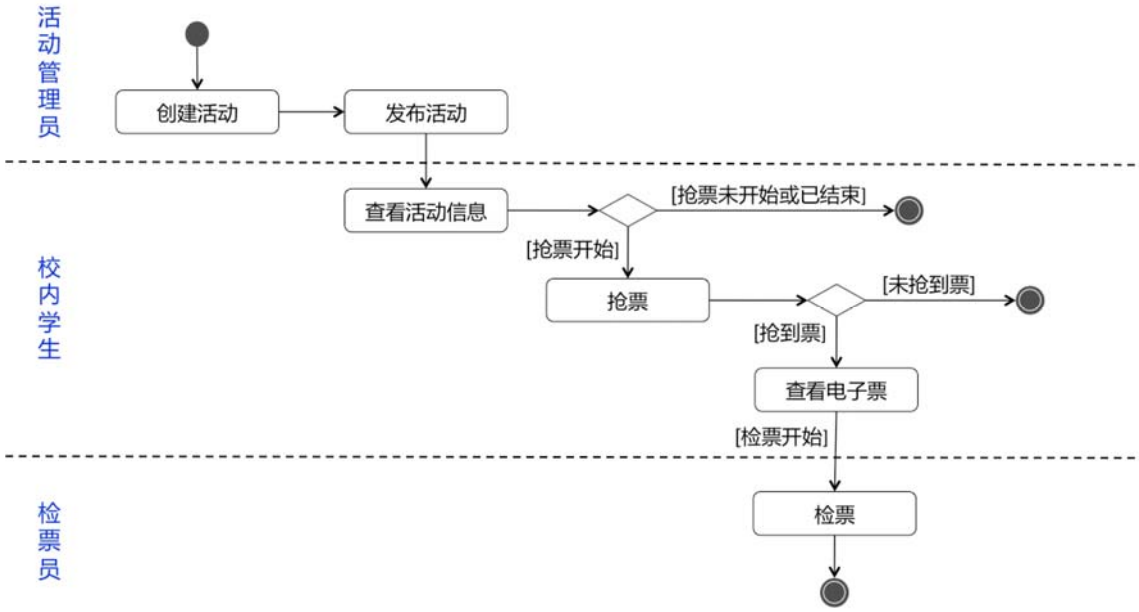
9.1 本章的教学内容

9.1.1 问题背景与系统需求

某学校团委经常组织一些活动，让组织人员非常头疼的一个问题是活动票的发放。尤其是一些受学生欢迎的热门活动，因为原来采取排队领票或购票形式，经常出现的场景就是在活动票发放前两三个小时门外就排起长长的队伍。为了改善学生排长队领票的不便，校团委希望开发一款微信抢票应用，为学生们提供热门活动推送、校园活动抢票等服务。

整个抢票过程包括创建和发布活动、抢票和检票三个阶段，具体过程如下图所示。校团委相关部门负责创建和发布活动，校内学生可以查看活动的详细信息。在抢票开始时，学生可以参与指定活动的抢票。如果学生成功抢到活动票，系统将自动生成一个带有二维码的电子票，学生可以查看自己的电子票；如果没有抢到票，系统将未抢到票的消息反馈给学生。

在活动开始时，校团委工作人员在活动入场处进行检票，学生可以持电子票检票参加活动。对于学生持有的电子票，工作人员使用二维码扫描枪进行扫描，验票成功即可入场，验票成功的条件是电子票有效且未被使用；学生也可以持自己的学生证，由工作人员通过学号查询电子票，再手动确认检票。



开发团队根据上述业务流程和要求，确定了微信抢票应用的系统需求。

1. 功能需求

- 活动管理员可以发布和维护最新的校园活动信息，包括活动名称、活动详情介绍、活动时间、活动地点、抢票数量、抢票时间等。

- 本校学生使用自己在学校信息门户的账号和密码实现微信号与校园账号的绑定。
- 学生可以查看校园活动的详细信息。
- 学生可以在活动抢票时段进行微信抢票，目前规定一个账号一次只能抢一张票。
- 学生在抢票成功之后可以获得系统生成的一张二维码电子票。
- 抢到票的学生在抢票未结束时可以退票。
- 抢到票的学生在活动开始时可以使用电子票通过检票进入活动现场。

2. 非功能需求

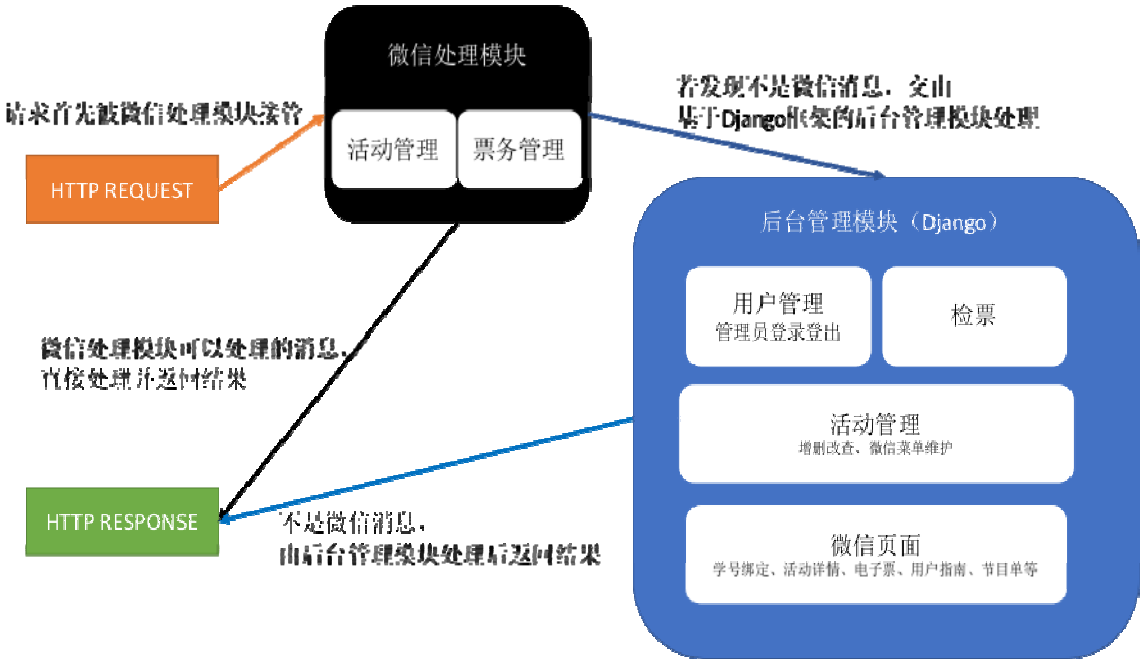
- 系统应能够支持 500 个用户并发访问。
- 系统应当支持 iOS 和 Andriod 两种主流手机操作系统。
- 在正常网络环境下，系统的响应速度应该控制在 5 秒以内。
- 所交付的系统源代码要求格式规范、风格统一，易于阅读和维护。
- 系统应该具有良好的架构设计，可扩展性强。
- 系统应具有良好的用户体验，并充分体现微信的交互特点。
- 系统应该保证安全可靠。

考虑到热门活动的门票可能比较紧俏，个别大学生可能抱有好奇心用非法手段钻系统的漏洞，开发团队认真讨论了可能出现的安全性问题，并提出了可能的应对方案。

- 问题 1：个别学生可能会利用刷票工具或者自己编写程序进行抢票，这样会造成遵守规定的学生无法抢到热门活动票。
- 解决方法：如果直接从活动详情页面点击进行抢票，就需要在应用程序中自己处理刷票的问题，这样势必会增加实现的难度。如果从微信中直接发送消息，那么抢票命令只能由操作人从微信公众号手动发出，刷票程序很难替代，因此最终采取抢票和退票均由微信命令进行操作的方式。
- 问题 2：个别学生可能复制别人抢到的电子票，并使用复制票进行检票入场。
- 解决方法：由于抢票活动是限制在校内学生参加的，所以可以把学号与电子票进行关联，一个学号只能对应一张电子票。另外，如果入场时由于手机忘带或者没电等造成无法出示电子票，也可以直接使用学生证通过学号进行检票。

9.1.2 技术方案选择

根据上述活动管理与抢票系统的需求，一个由六位同学组成的开发小组提出了自己的系统方案。整个设计的关键在于微信消息处理和后台管理两个部分，下图简要概括了一个 HTTP REQUEST 是如何在所设计系统的各个模块协作下完成处理的。



在 Web 开发框架上，开发小组选择了 Python 的 Django 框架，因为 Django 有非常好用的数据库 ORM。在数据库选型上，开发小组选择了自己熟悉的关系型数据库 MySQL。此外，部署通过 nginx+uwsgi 来实现多进程后端，以此提高性能。在抢票的正确性方面，由于对票进行修改需要同时修改活动的剩余票数，因此通过数据库的 atomic 原子锁来保证，当生成一张新票时会将活动的表锁定并修改剩余票数然后再解锁。

当服务器程序收到 HTTP 请求时，会先由微信处理模块进行处理。若是微信发来的消息，就根据消息的要求进行处理并返回结果即可；若不是，则将请求交给基于 Django 框架的后台管理模块进行处理并返回结果。

后台管理模块还需要管理前端页面，包括管理员使用的后台管理页面以及用户直接在微信上可以打开的相应页面。由于这些页面都比较简单，因此前端的 HTML 放在后端 template 中进行渲染。

9.1.3 学生开发作品

根据前面给出的系统求和架构设计，开发小组实现了微信抢票应用 V1.0 版本，并在 github 上发布了源代码，具体地址是 https://github.com/chenhuarong/tsinghuatuan，相关开发文档放在 https://github.com/chenhuarong/tsinghuatuan/tree/master/docs。

实际上，前面给出的抢票应用技术方案方面还是存在不少问题，你发现了吗？尽管如此，上述系统还是成功地上线使用一年多时间，并且获得校内学生的欢迎。由此可见，系统架构是否最优并不是一个项目投入实际使用的决定性因素，但是要使这个系统的生命力更强，应该在适当的时候对系统架构进行一些优化。

随着抢票活动以及参与抢票人数的增加，校团委升级了服务器，同时也升级了公众号，现在的公众号具有向用户主动推送消息等高级功能。在此基础上，校团委希望对现有抢票系统进行一个大版本的升级改造。具体要求如下：

（1）现有系统只支持单人抢单张票，而且是自动分配座位，新系统希望在一次活动中用户可以抢不超过设定最大票数的任意张票，而且可以为每张票选择座位。

（2）当管理员创建活动后，希望之前参加过同类活动的用户能收到该活动的推送。当用户抢到票后，活动开始当天早 9 点和活动开始前 30 分钟，用户都能收到一条推送消息提醒及时检票入场。

（3）原有系统在个别热门活动的抢票中出现过系统不响应的问题，希望新的系统能够改进性能，能够至少支持 1000 个用户的并发访问。

9.2 讨论题

你可以对新系统提出更好的架构方案吗？想想看，提出更好的架构方案是为了适应新需求，还是为了让系统获得更好的开发效率和运行性能？为什么？

9.3 作业题

请提交一个 pdf 文档，在其中详细说明你设计的符合以上要求的系统改进方案。在文档中请仔细分析新系统的每一个需求，并提出完整的系统架构方案，并针对每个需求分析在你的架构设计中该需求如何实现，如果在你的架构设计中该需求的实现方式与旧系统不同，请说明你这样设计的理由和优势。

此外，请在文档中回答问题：提出更好的架构方案是为了适应新需求，还是为了让系统获得更好的开发效率和运行性能？为什么？

提示：请从 MVC、RESTful API、数据库选择、异步任务队列、缓存等角度考虑新系统的架构方案。