软件工程复习笔记

徐大鹏

2017 年 6 月 18 日

软件工程大水课！一定要考好！求保佑！

1 软件工程概述

1.1 复习提纲里的考点

1. SE的定义、目的、方法及作用(P2 / P16)
   定义是什么？方法呢？作用呢？不知道。

   - 章前简介：我们的最终目标是，生产出高质量软件，进而找到解决方案，并考虑那些对质量有影响的特性。
   - 1.2节：要写出健壮的、易于理解和维护的并且能以最高效的方式完成工作的代码，必须具备专业软件工程师的技巧和洞察力。因此软件工程的目标就是设计和开发高质量软件。
   - 1.1.2节：软件工程师的角色：软件工程师的精力集中于将计算机作为求解问题的工具，而不是研究硬件设计或者算法的理论证明。

2. 说明**错误**、**缺陷**、**失败**的含义与联系。（请举例说明）（6 页）（44 页习题3）
   当人们在进行软件开发活动的过程中出错（错误）时，就会出现故障。失效是指系统违背了它应有的行为。故障是系统的内部视图，是从开发人员的角度看待系统；失效是系统的外部视图，是从用户的角度。见图1。

3. 软件质量应从哪几个方面来衡量？(P9 – P12)

   产品的质量 用户角度：易于学习、易于使用；故障的数目少，故障类型都是次要的（次要的、主要的、灾难性的）。设计和编写代码的人

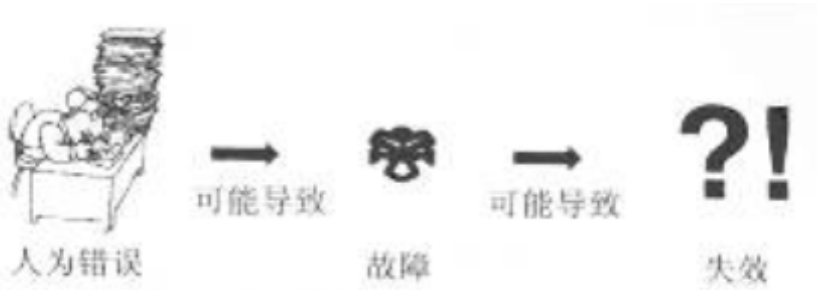

图1-4　人为错误是如何引起失效的

Figure 1: 错误和失效

员、维护该程序的人员：考虑产品的内部特性，把故障的数目和类型看做产品质量的证据。

过程的质量 只要有活动出了差错，产品的质量就会受到影响。提出问题：What? When? Where? How?

商业环境背景下的质量 提供的产品和服务。

4. 现代软件工程大致包含的几个阶段及各个阶段文档(P23–P24)
   1.6.2节。这个问题也是软件过程由哪几个主要部分组成的答案。

   需求分析和定义 与客户会面以确定需求，这些需求是对系统的描述。

   系统设计 系统设计告诉客户，从客户的角度看，系统会是什么样的。然后客户要对设计进行评审。当设计得到批准之后，整个系统设计将被用来生成其中单个程序的设计。

   程序设计

   编写程序

   单元测试 链接之前作为单独的代码段进行测试。

   集成测试 将模块组合到一起，确保他们能够正确运行。

   系统测试 对整个系统的测试，用于确保起初指定的功能和交互得以实现。

   系统交付

   维护 出现任何问题，或者需求发生变化时。

5. 什么是抽象？(P30)

   1.8.2节。抽象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。

6. 什么是软件过程？软件过程的重要性是什么？包含几个阶段？(P32, P45)

   软件过程的定义和重要性在第二章有相应的问题。包含哪几个阶段在上面的问题中提到过。图2仅供参考。

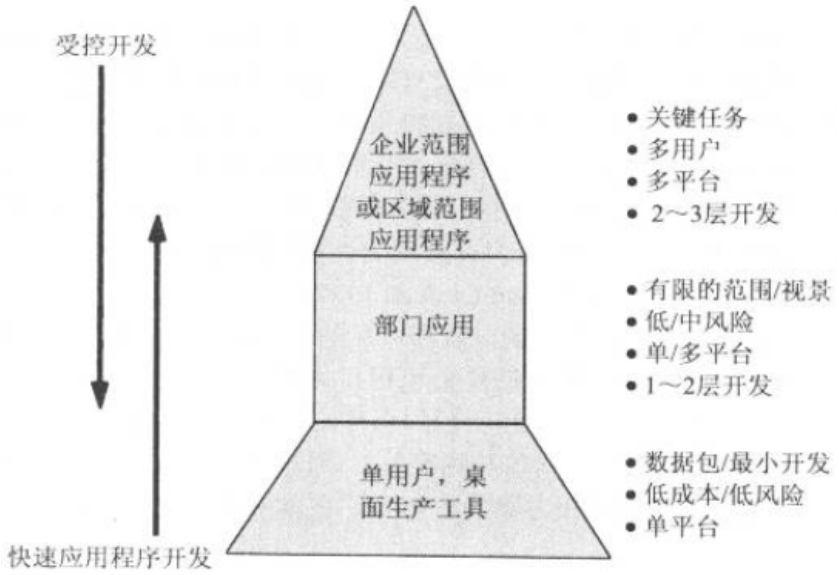

图1-14　不同开发中的差别（Wasserman 1996）

Figure 2: 软件过程的差别

7. 什么是重用等软件工程主要概念？(P34)

   1.8.2节。图3

在软件开发和维护中，通常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。Barnes和Bollinger指出，复用并不是一个新的思想，他们还给出了很多有趣的例子，说明复用的不仅仅是代码（Barnes and Bollinger 1991）。

Prieto-Diaz介绍了这样一种理念：可复用构件是一种商业资产（Prieto-Diaz 1991）。公司和组织机构对那些可复用的项进行投资，而当这些项再次用于后面的项目中的时候，就可以获得巨大的收益。但是，制定一个长期、有效的可复用计划可能是很困难的，因为存在如下这些障碍。

Figure 3: 复用

2 模型化过程和生命周期

2.1 提到的考点

1. 什么是软件过程？软件过程的重要性是什么？(P45-46)
   2.1节。
   软件过程是软件开发活动中产生某种期望结果的一系列有序任务，涉及活动、约束和资源。

   重要性 它强制活动具有一致性和一定的结构。当我们知道如何把事情做好而且希望其他人也能以同样的方式做事时，这些特性就很有用。

   灵活性 当然，可以在过程模型一致的前提下，不同的开发人员或者开发团队选用不同的技术和工具来完成某个开发过程，这体现了灵活性。

   PPT上的描述：

   过程的重要性 一致性和结构性可以使我们知道是否已经做好了工作，还能使别人以同样的方式做工作，因而具有相对通用性。过程有助于保持大量不同人员开发的产品和服务之间的一致性和质量。

2. 瀑布模型及各阶段文档，优缺点？(P49)
   瀑布模型将开发阶段描述为从一个阶段瀑布般地转换到另一个阶段，一个开发阶段必须在另一个开发阶段开始之前完成。

   优点 在帮助开发人员布置他们需要做的工作时，瀑布模型是非常有用的。它的简单性使得开发人员很容易向不熟悉软件开发的客户做出解释。它明确地说明，为了开始下一阶段的开发，那些中间产品是必需的。

   缺点 (1)瀑布模型并不能反应实际的代码开发方式。实际上软件是通过大量的迭代进行开发的。(2)文档转换有困难。

3. 原型的概念(P51)
   原型是一个部分开发的产品，它使客户和开发人员能够对计划开发的系统的相关方面进行检查，以决定它对最终产品是否合适或恰当。

4. 论述分阶段开发模型的含义, 其基本分类及特点是什么？(56 页)
   2.2.6节，阶段化开发：增量和迭代。
   关键概念：循环周期、、。

阶段化开发是指，系统被设计成部分提交, 每次用户只能得到部分功能, 而其他部分处于开发过程中。

产品系统：用户正在使用的版本

开发系统：准备代替现有产品系统的下一个版本

基本分类：

增量开发 系统需求按照功能分成若干子系统，开始建造的版本是规模小的、部分功能的系统，后续版本添加包含新功能的子系统，最后版本是包含全部功能的子系统集。

迭代开发 系统开始就提供了整体功能框架，后续版本陆续增强各个子系统，最后版本使各个子系统的功能达到最强。

5. 螺旋模型四个象限的任务及四重循环的含义？(P58)

螺旋模型将开发活动与风险管理结合起来, 以降低和控制风险。四个象限：计划，确定目标、可选方案和约束，评估可选方案和风险，开发与测试。四次迭代：操作概念、软件需求、软件设计、系统测试与交付。

要看习题：P80–81 页习题2，3。

6. 什么是UP，RUP?

统一软件开发过程（英语：Rational Unified Process，缩写为RUP），一种软件工程方法，为迭代式软件开发流程。最早由Rational Software公司开发，因此冠上公司名称。因此名字上可以直接叫做统一过程(UP)；Rational是公司名，也写作RUP。

统一过程 用例驱动的、以基本架构为中心的、迭代式和增量性的软件开发过程框架。

统一过程将重复一系列生命周期，这些生命期构成了一个系统的寿命。每个生命周期都以向客户推出一个产品版本而结束。统一过程的每个周期包括四个阶段，

构思阶段(inception phase) 包括用户沟通和计划活动两个方面，强调定义和细化用例，并将其作为主要模型。

细化阶段(elaboration phase) 包括用户沟通和建模活动，重点是创建分析和设计模型，强调类的定义和体系结构的表示。

构建阶段(construction phase) 将设计转化为实现，并进行集成和测试。

移交阶段(transition phase) 将产品发布给用户进行测试评价，并收集用户的意见，之后再次进行迭代修改产品使之完善。

每个阶段又可以进一步划分为多次迭代。见图4。

统一过程（UP）定义了下列三个支持工序(discipline)。（感觉看看就行）

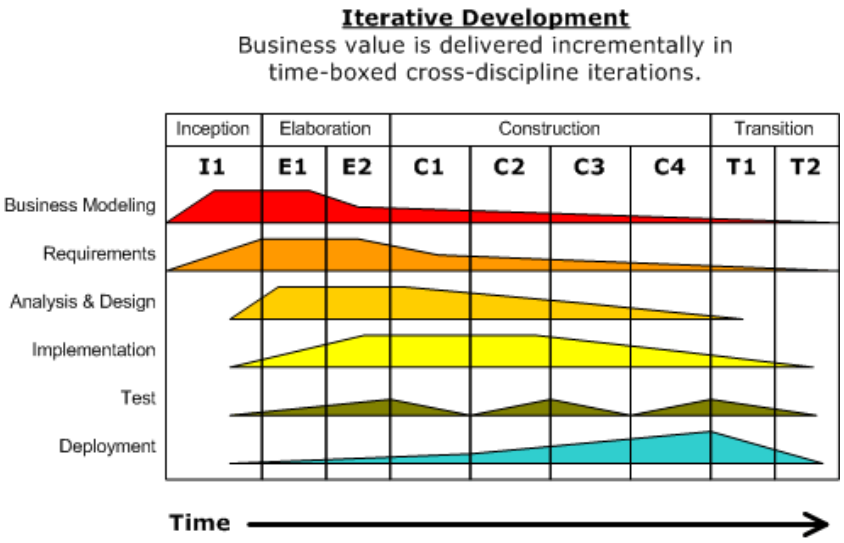

Figure 4: 统一过程

(a) 配置变更管理工序，用来管理系统和需求变更的配置。

(b) 项目管理工序，用来管理项目。

(c) 环境配置工序，用来配置项目的环境，包括所涉及到的过程和工具。

统一过程定义了下列六个核心工序（这个和一般过程相似）

(a) 业务模型工序，通过业务模型获取相关知识以理解需要系统自动完成的业务。

(b) 需求工序，通过用例模型获取相关知识以理解自动完成业务的系统需求。

(c) 分析设计工序，通过分析/设计模型以分析需求，设计系统结构。

(d) 实现工序，基于实现模型实现系统。

(e) 测试工序，通过测试模型进行针对需求的系统测试。

(f) 部署工序，通过部署模型部署系统。

进化式迭代开发（Iterative development）

(a) 迭代开发是统一过程的关键实践

(b) 开发被组织成一系列固定的短期小项目

(c) 每次迭代都产生经过测试、集成并可执行的局部系统

(d) 每次迭代都具有各自的需求分析、设计、实现和测试

(e) 随着时间和一次次迭代，系统增量式完善

3 计划和管理项目

3.1 复习提纲里的考点

1. 什么是项目进度？活动？里程碑？(83页)

   项目进度 通过列举项目的各个阶段，把每个阶段分解成离散的任务或活动，来描述特定项目的软件开发周期。

   活动 是项目的一部分，在一段时间内发生。

   里程碑 是活动完成的时刻。

2. 如何计算软件项目活动图的关键路径？冗余时间？最早和最迟开始时间(习题2, 3) (课堂习题讲解)
   关键路径就是最长路径，即每一个节点的时差都为零的路径。冗余时间也就是时差，满足

   $$\text{时差} = \text{可用时间} - \text{真实时间} = \text{最晚开始时间} - \text{最早开始时间}$$

   计算上，要先求出最长路径，然后沿最长路径回溯，找到每一个活动的最早开始时间和最晚开始时间，然后求出每一个活动的时差。

3. 软件团队人员应该具备的能力是什么？(96 页)
   3.2.1节。见图5。

4. 软件项目组织的基本结构？(101 页)
   主程序员负责制和忘我方法。根据实际情况可以结合这两种极端情况。主程序员负责制的组织结构如何？忘我方法适于那些情况？他们的对比？（对比见图6）

- 完成工作的能力。
- 对工作的兴趣。
- 开发类似应用的经验。
- 使用类似工具或语言的经验。
- 使用类似技术的经验。
- 使用类似开发环境的经验。
- 培训。
- 与其他人交流的能力。
- 与其他人共同承担责任的能力。
- 管理技能。

其中每一种特性都可能影响个人有效完成工作的能力。

Figure 5: 团队人员应该具备的能力

表3-5 组织结构的比较

高度结构化	松散的结构
高度确定性	不确定性
重复	新技术或工艺
大型项目	小型项目

Figure 6: 组织结构的对比

5. 试述COCOMO模型的三个阶段基本工作原理或含义。(111 页)
   3.3.2节。见图7

6. 什么是软件风险？主要风险管理活动？有几种降低风险的策略？(P119, P122)
   易。

7. 找出图3.23和图3.24(P139)的关键路径。
   章末必做练习题。

4 获取需求

4.1 提到的考点

1. 需求的含义是什么?(143 页)

在阶段1，项目通常构建原型以解决包含用户界面、软件和系统交互、性能和技术成熟性等方面在内的高风险问题。这时，人们对正在创建的最终产品的可能规模知之甚少，因此COCOMO II用应用点（其创建者对它的命名）来估算规模。正如我们将看到的，这种技术根据高层的工作量生成器（如屏幕数量和报告数量、第3代语言构件数）来获取项目的规模。

在阶段2（即早期设计阶段），已经决定将项目开发向前推进，但是设计人员必须研究几种可选的体系结构和操作的概念。同样，仍然没有足够的信息支持准确的工作量和工期估算，但是远比第1阶段知道的信息要多。在阶段2，COCOMO II 使用功能点对规模进行测量。功能点是在参考文献IFPUG（1994a and b）中详细讨论的一种技术，估算在需求中获取的功能。因此，与应用点相比，它们提供了更为丰富的系统描述。

在阶段3（后体系结构阶段），开发已经开始，而且已经知道了相当多的信息。在这个阶段，可以根据功能点或代码行来进行规模估算，而且可以较为轻松地估算很多成本因素。

Figure 7: COCOMO模型的三个阶段

需求就是对期望的行为的表达。需求指定客户想要什么行为，而不是如何实现这些行为。

2. 需求作为一个工程，其确定需求的过程是什么？(144 页图4.1)
   图8

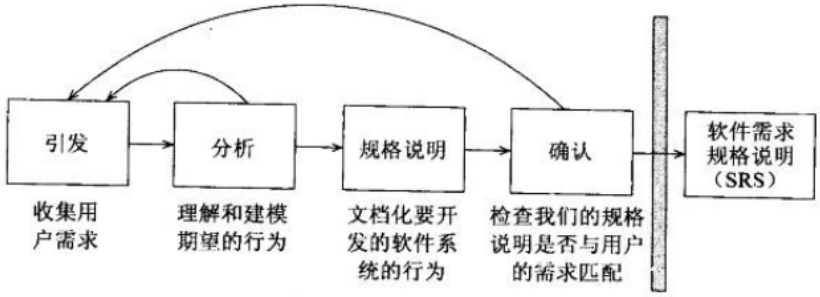

图4-1 获取需求的过程

Figure 8: 获取需求的过程

3. 举例说明获取需求时，若有冲突发生时，如何考虑根据优先级进行需求分类。(152 页)
   4.3.1节。
   一般可以分为三类：必需的、值得要的、可选的。

4. 需求文档分为哪两类？(153 页)
   4.3.2节。

需求定义 是客户想要的每一件事情的完整列表。

需求规格说明 将需求重新陈述为关于要构建的系统将如何运转的规格说明。

5. 什么是功能性需求和非功能性需求/质量需求？设计约束？过程约束？(149 页)
   4.3节开头。
   功能需求：描述系统内部功能或系统与外部环境的交互作用，涉及系统输入应对，实体状态变化，输出结果，设计约束与过程约束等。
   非功能需求或质量需求：描述软件方案必须具备的某些质量特征。
   设计约束：是已经做出的决策或者限制问题解决方案集的设计决策。物理环境：对环境或设备的限制等（安装及环境要求等）接口：涉及输入输出的限制或约束条件. (输入格式预定等) 用户：使用者的基本情况（限定几种类型的用户）
   过程约束：是对于构建系统的技术和资源的限制。资源：材料、人员技能或其它。文档：类型、数量或其它。（涉及其针对性及要求等）标准：……
   其他：什么原因会导致从工资单列表中删除某雇员？

6. 知道DFD图的构成及画法(172 页)
   4.5.8节

7. 在需求原型化方面，什么是抛弃型原型？什么是演化型原型？(192–193 页)
   4.7节末尾。

5 设计体系结构和设计模块

1. 什么是软件体系结构？设计模式？设计公约？设计？概念设计？技术设计？(223-224 页)
   5.1节开头，5.1.1节。

   体系结构 一种软件解决方案，用于解释如何将系统分解为单元，以及单元如何相互关联，还包括这些单元的所有外部特性。

   设计模式 是一套被反复使用的（多数人知晓并经过分类编目的）代码设计经验的总结，使用设计模式目的是为了可重用代码、让代码更容易被他人理解并且保证软件质量。

   设计公约 一系列设计决策和建议的集合，用于提高系统某方面的设计质量。

设计 是一种创造性过程，它考虑如何实现所有的客户需求。设计所产生的计划也称为设计。

概念设计 告诉客户设计出的系统可以做什么（软件的体系结构和功能，更接近于系统设计）

技术设计 告诉程序员设计出的系统如何做什么（软件功能和接口的实现方法，也是程序员的参考文档，更接近于程序设计）

2. 软件设计过程模型的几个阶段？
   跟第四章第二个提到的考点差不多。图9

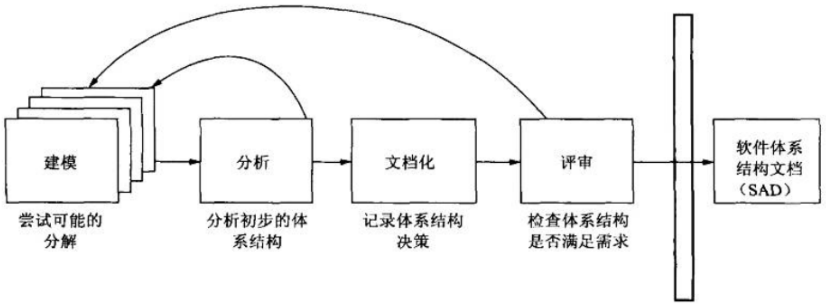

图5-4 软件体系结构开发过程

Figure 9: 软件体系结构开发过程

3. 什么是模块化？什么是抽象？(238 页)
   6.2.1节和6.2.5节。

4. 论述设计用户界面应考虑的问题。(242 页)
   6.6.3节。可以试着从Who? What? When? Where? How? 的角度去考虑，一共6点。

5. 耦合与内聚的概念及各个层次划分？(291—301 页)
   6.2.1节。

   耦合度 模块之间的依赖程度的衡量。

   内容耦合 一个模块修改了另一个模块的内部数据项、代码，或者一个模块内的分支转移到另一个模块。

   公共耦合 从公共数据区存储区来访问数据。所有访问过该数据的模块都会受到影响。

控制耦合 一个模块通过传递参数或返回代码来控制另外一个模块的活动。

标记耦合 使用一个复杂的数据结构来从一个模块到另一个模块传递信息。

数据耦合 两个模块之间有信息传递，但传送的只是数据值，而不是结构数据。

非耦合 完全没有耦合。

内聚度 模块的内部元素（数据、功能、内部模块等）的粘合程度。

巧合内聚 模块的各个部分互不相关。只是由于为了方便或者偶然的原因，不相关的功能、进程或数据处于同一个模块中。

逻辑内聚 模块的各个部分只通过代码的逻辑结构相关联。

时间性内聚 部件各部分要求在同一时间完成。

过程性内聚 各部分有特定次序。

通信内聚 各个部分访问共享数据。

功能内聚 在一个模块中包含了所有必需的元素，并且每一个处理元素对于执行单个功能来说是都是必需的。

信息内聚 在功能内聚的基础上，将其调整为数据抽象化和基于对象的设计。

6. 举例说明耦合与内聚的基本分类。以及各个分类的含义与特征(284 页习题4，5)

   章末必做练习题。

7. 面向对象设计的基本原则？

   单一职责原则 一个类只负责一个功能领域中的相应职责。

   重用原则

   开闭原则

   替换原则

   依赖倒置原则

   接口隔离原则

   德米特法则 一个软件实体应当尽可能少地与其他实体发生相互作用。(例如，在两个或多个组件之间引入一个中间类，将网状结构转变为中心式结构，避免多个组件之间的直接交互)

8. 面向对象开发有何优势？(291 页)
   (a) 语言的一致性。使用相同语义的概念来描述问题和解决方案：类，对象，接口，方法，属性，行为。
   (b) 过程的一致性。从需求分析、顶层设计、底层设计、编码到测试，所有的过程都使用相同语义的概念。
9. 面向对象开发过程有几个步骤？(292 页)
   (a) 面向对象需求分析
   (b) 面向对象顶层设计
   (c) 面向对象底层设计
   (d) 面向对象编程
   (e) 面向对象测试
10. 熟悉用例图的组成和画法，用例的几个要素的含义，掌握用例图的实例解析方法(294页)
11. 用例图、类图等对面向对象的项目开发的意义是什么？
12. 熟悉类图中各个类之间的基本关系分类(303-305)
    关联、组合、聚合、泛化、依赖、导航。
13. 熟悉类图等的组成和画法(300-308 页)
14. 知道UML 其他图示结构的基本用途。

6 编写程序

6.1 提到的考点

1. 在编写程序内部文档时，除了HCB外，还应添加什么注释信息？(352-354 页)
   7.3.1节。
2. 什么是极限编程(XP)? 以及结对编程？(357 页) 7.4.2、7.4.3节。

7 测试程序

7.1 提到的考点

1. 几种主要的缺陷类型？(367-368 页)

   算法缺陷 算法的某些处理步骤或逻辑有问题，以至于软件部件对给定的输入数据无法产生正确的输出。

   计算和精度缺陷 算法或公式在编程实现时出现错误或最终结果达不到精度要求。

   文档缺陷 文档与程序实际做的事情不一致。

   过载缺陷 当填充数据结构时如果超过了他们规定的能力。

   能力缺陷 系统活动达到指定的极限时，系统性能会变得不可接受。

   时序性缺陷 在实时系统中，几个同时执行的或按仔细定义的顺序执行的进程出现混乱。

   性能缺陷 系统不能以需求规定的速度执行。（同能力缺陷的区别：这里指正常条件下性能需求不能满足）

   恢复性缺陷 系统不能从失效中恢复。

   硬件和系统软件缺陷 提供的硬件和系统软件没有按照文档中的操作条件和步骤运行。

   标准和过程缺陷 代码没有遵循组织机构的标准和过程。

2. 什么是正交缺陷分类？(369 页)
   被分类的任何一项缺陷都只属于一个类别，那么这种缺陷分类方案是正交的。

3. 测试的各个阶段及其任务？(372 页图8.3)
   8.2.1节。

4. 掌握测试的方法—黑盒、白盒的概念？(374)
   8.2.4节。

5. 什么是单元测试？什么是走查和检查？(376 页)
   单元测试的概念：8.2.1节。
   代码走查和代码审查：8.3.1节。

6. 黑盒白盒方法各自的分类？各种覆盖方法等。测试用例的设计和给出方法(课件和补充课件)

黑盒测试方法：

等价分类法 每一个测试用例都代表了一类与它等价的其它例子。如果用这个例子未能发现程序的错误，则与它等价的其它例子一般也不会发现程序的错误。

边界值分析法 在等价分类法中，代表一个类的测试数据可以在这个类的允许范围内任意选择。但如果把测试值选在等价类的边界上，往往有更好的效果，这就是边界值分析法的主要思想。边界值分析也适用于考察程序的输出值边界。

错误猜测法 所谓猜错，就是猜测被测程序中哪些地方容易出错，并据此设计测试用例。如果说等价法和边界值法均有线索可寻，则猜错法将更多地依赖于测试人员的直觉与经验。所以在通常情况下，这种方法仅用作辅助手段，即首先用其它方法设计测试用例，再用猜错法补充一些例子。

因果图法 因果图法是借助图形来设计测试用例的一种系统方法，特别适用于被测程序具有多种输入条件，程序的输出又依赖于输入条件的各种组合的情况。

白盒测试方法：

逻辑覆盖法 逻辑覆盖是一组覆盖方法的总称。按照由低到高对程序逻辑的覆盖程度，又可区分为以下几种覆盖：

语句覆盖 使被测程序的每条语句至少执行一次

判定覆盖 使被测程序的每一分支至少执行一次，故又称分支覆盖

条件覆盖 要求判定中的每个条件均按“真”、“假”两种结果至少执行一次

路径测试 基于程序图设计的一种白盒测试方法。程序图本质上是一种简化了的流程图，它保持了控制流的全部轨迹，包括了所有的判定结点，但由于舍弃了许多细节，使画面远比流程图为简洁。覆盖标准如下：

结点覆盖 程序的测试路径至少经过程序图中每个结点一次，相当于逻辑覆盖法中的语句覆盖。

边覆盖 程序的测试路径至少经过程序图中的每条边一次，它相当于逻辑覆盖法中的判定覆盖。

同时满足上述两种覆盖的覆盖称为完全覆盖。

7. 如何面对一个命题，设计和给出测试用例的问题。(课件)—课堂练习的测试题目和讲解内容

8. 集成测试及其主要方法的分类？(390-392)(驱动，桩的概念)
   8.4节。

8 测试系统

8.1 提到的考点

1. 系统测试的主要步骤及各自含义？(420 页, 图9.2)
   见上一章。

2. 什么是系统配置？软件配置管理？(423 页)(或见课件)
   9.1节。

3. 功能测试的含义极其作用？(430 页)
   9.2节。

4. 功能测试的基本指导原则？(431)
   9.2节。

5. 性能测试的含义与作用？(436 页)
   9.3节。

6. 性能测试的主要分类？(436 页)
   9.3节。

7. 确认测试, 确认测试分类？(基准测试和引导测试)(447-448 页)
   9.5节。

8. 什么是α测试？β测试？(448 页)
   9.5节。