《软件工程》教学内容回顾 2014 下与整理

(下述问题仅是课件中的主要部分回顾，问题答案以课件为主要参考)

Chapter01

SE 的定义、目的、方法及作用（P2 / P16）

定义：是一个系统工程，是采用工具、技术等用来解决现实问题的综合过程，既有对技术问题的分析与综合，也有对开发过程和参与者的管理。

目的、作用：以计算机科学理论和计算机功能为基础，通过对要解决问题的本质的了解，采用相应的工具和技术，实现设计方案，推出高质量的软件产品。确保软件具有技术高质量和实际商业价值。

方法：是把软件产品（就像其他工业品一样）看作是一个工程产品来处理，把需求计划、可行性研究、工程审核、质量监督等工程化的概念引入到软件生产中，期望达到一般工程项目的三要素：进度、经费、质量。

//开发模式（paradiam）（P4）

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）

错误：计算机计算得到的、观察到的、测量到的数值或者条件和理论上得到的正确的数值或者条件之间存在的差异//在软件开发过程中人为错误

缺陷：程序或者软件中不正确的步骤、过程或者数据定义等。//功能实现过程中产生的问题

失败：软件系统或单元无法实现需求文档中规定的功能特性或者非功能特性。或者说单元/系统产生的结果与期望交付的服务或者结果存在偏差。外部的失效/失败是内部缺陷在执行测试软件时的外部反映。//软件运行中的失败

联系：一个错误往往对应了多个缺陷；缺陷是静态存在的，而失败是动态存在的

举例：Sperry Corporation 项目计划不周 Jet fighter 软件未能预期的使用

软件质量应从哪几个方面来衡量？论述之。（9--12 页）

产品的质量：用户从外部特性看软件是否具有足够多的功能并且易于学习和使用，或者对应用于复杂的功能虽然难以学习和使用却值得；从业人员从内部特性把故障的数量和类型以及缺陷分布区间作为产品质量的证据。

生产该产品过程的质量：很多活动会影响到最终的产品质量，如果任何一个活动出了差错，产品的质量就会受到影响。（过程建模与过程改进）

在产品将使用的商业环境下产品的质量（商业应用背景下的软件质量或商业质量）：目标是将技术价值与商业价值统一起来，商业价值一般指机构对软件是否与其战略利益相吻合的一种价值评估，其一般需要在需求阶段预判。

//软件系统的系统组成（P16）

现代软件工程大致包含的几个阶段及各个阶段文档（P23-24）

需求分析：问题定义、可行性研究、需求分析（**SRS 软件需求规格说明书**）、复审

系统设计：用户界面、**系统结构图**、复审

程序设计：**模块功能算法与数据描述**、复审

程序实现：编程与调试（**源代码与注释**）、复审

单元测试：模块功能与性能测试（**测试报告**）、复审

集成测试：结构图综合测试（**测试报告**）、复审

系统测试：按照 SRS 对系统总体功能等进行测试、复审

系统提交：交付、复审（**用户手册**）

维护：修改软件的过程、为改错或满足新需求（**维护报告**）、复审

// 使现代 SE 实践发生变化的（七个）关键因素是什么？（28--29 页）

什么是抽象？（30 页）

基于某种层次归纳水平的问题描述。它使我们将注意力集中在问题的关键方面而非细节。

什么是软件过程？软件过程的重要性是什么？包含几个阶段？（32 页）（45 页）

软件开发活动中的各种组织及规范方法。是其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。

其他问题参见*第二章*。

什么是重用？（34 页）

重复采用以前开发的软件系统中具有共性的部件，用到新的开发项目中去。

Chaoter02

什么是软件过程？软件过程的重要性是什么？ （P45-46）

软件开发活动中的各种组织及规范方法。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 1 - 通用性（一致性与结构性）：一致性与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。

重要性 2 – 指导性：分析、检查、理解、控制和提升活动。

瀑布模型及各阶段文档，优缺点？（P49）

定义：将开发阶段描述为从一个阶段瀑布般的转换到了另一个阶段。一个必须在另一个完成后才能完成。

特点：阶段间的依赖性和连续性；尽可能推迟程序的物理实现；每个阶段必须完成规定的文档，每个阶段结束前都要对所完成的文档进行评审

文档：

需求分析——SRS 软件需求规格说明书；
系统设计——系统设计文档如软件结构图；
程序设计——模块功能算法和数据描述文档；
编码——源程序和注释；
单元测试和集成测试——单元测试报告；
系统测试——系统测试报告；
验收测试——验收测试报告；
运行与维护——维护报告。

优点：可描述的软件开发活动，使用里程碑、提交物，便于过程评价；简化，便于向用户解释；其他复杂模型的技术，可以增加反馈循环与额外活动

缺点：面临软件变动时，该模型无法处理实际过程中的重复开发问题，因为软件是一个创造的过程，不是一个制造的过程；文档转换有困难

原型的概念（P51）

一种部分开发的产品，用来让用户和开发者共同研究，提出意见，为最终产品定型。

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）

定义：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。

基本分类：

A 增量式开发： 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。

B 迭代式开发： 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，最后版本是各个子系统的功能达到最强。

特点： 每个软件版本的周期减少

螺旋模型四个象限的任务及四重循环的含义？ （P58）

任务：计划、目标/可选方案、风险评估、开发和测试

循环：操作概念、软件需求、软件设计、系统实现与执行

P80--81 页 习题 2， 3。（见作业）

//在所有的软件开发过程模型中，你认为哪些过程给予你最大的灵活性以应对需求的变更？（81 页习题 11）

什么是 UP， RUP?

UP:统一过程是用例驱动的、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，它使用对象管理组织的 UML 并与对象管理组织的软件工程原模型等相兼容。

RUP：统一开发过程将重复一系列生命期，每个周期包括四个阶段：开始阶段、确立阶段、构建阶段、移交阶段。每个阶段可以进一步划分为多次迭代。其定义了三个支持工序:配置变更管理工序、项目管理工序和环境配置管理工序。定义了六个核心工序：业务模型工序、需求工序、分析设计工序、实现工序、测试工序、部署工序。

Chapter03

什么是项目调度？活动？里程碑？（83 页）

项目调度是对特定项目的软件开发周期的刻画。包括对项目阶段、步骤、活动的分解，对各个离散活动的交互关系的描述，以及对各个活动完成时间及整个项目完场时间的初步估算。

活动：项目的一部分，一般占用项目进度计划中的一段时间。

里程碑：指特定的时间点，标志着活动的结束，通常伴随着提交物

如何计算软件项目活动图的关键路径？（习题 2，3）冗余时间？最早和最迟开始时间（课堂习题讲解）

关键路径术语

（1） 关键路径：从源点到汇点的路径长度最长的路径叫关键路径。

（2） 活动开始的最早时间e(i)

（3） 活动开始的最晚时间l(i) 定义e(i)=l(i）的活动叫关键活动。

（4） 事件开始的最早时间ve(i)

（5） 事件开始的最晚时间vl(i)

设活动ai由弧<j,k>；（即从顶点j到k）表示，其持续时间记为dut(<j,k>；），则

e(i)=ve(j)

l(i)=vl(k)-dut(<j,k>)（6_6_1）

求ve(i）和vl(j）分两步：

· 从ve（1）=0开始向前递推

ve(j)=Max{ ve(i)+dut(<i,j>) }（6_6_2）

<i,j>T,2<=j<=n

其中，T是所有以j为弧头的弧的集合。

· 从vl(n)=ve(n）开始向后递推

vl(i)=Min{ vl(j)-dut(<i,j>) }（6_6_3）

<i,j>S,1<=i<=n-1

其中，S是所有以i为弧尾的弧的集合。

两个递推公式是在拓扑有序和逆拓扑有序的前提下进行。

4、 求关键路径的算法

（1） 输入e条弧<j,k>；，建立AOE网的存储结构。|

（2） 从源点v1出发，令ve（1）=0，求 ve(j) 2<=j<=n。

（3） 从汇点vn出发，令vl(n)=ve(n），求 vl(i) 1<=i<=n-1。

（4） 根据各顶点的ve和vl值，求每条弧s（活动）的最早开始时间e(s）和最晚开始时间l(s），其中e(s)=l(s）的为关键活动。

求关键路径是在拓扑排序的前提下进行的，不能进行拓扑排序，自然也不能求关键路径。

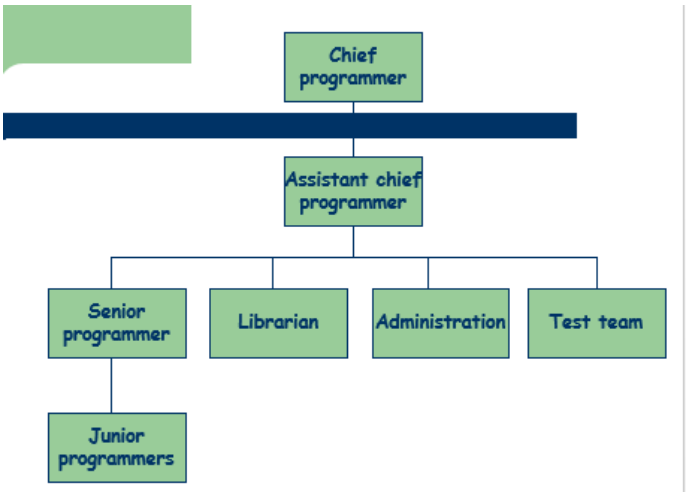
软件人员应该具备的能力是什么？（96 页）

1 完成工作的能力 2 对工作的兴趣 3 开发类似应用的经验、使用类似工具成语言的经验、使用类似技术的经验、使用类似开发环境的经验 4 培训 5 与其他人交流的能

力 6 与其他人共同承担责任的能力 7 管理技能

软件项目组织的基本结构？（101 页）

主程序员组：



优点：高度确定性、稳定性、一致性和重复性

// 专家估算法的大致含义？（106 页），算式估算法的大致含义？（108 页）

试述 COCOMO 模型的三个阶段基本工作原理或含义。（111 页）

阶段一——应用组装：1 项目构建原型以解决高风险问题；2 使用应用点来估算规模

阶段二——早期设计：1 设计人员必须研究几种可选的体系结构和操作概念 2 使用需求文档中的功能点来估算规模

阶段三——后体系结构：1 开发已经开始，软件已经部分实现 2 使用需求文档中的功能点或代码行来进行规模估算

什么是软件风险？有几种降低风险的策略？（119、122 页）

软件风险：在软件生产过程中不希望看到的、有负面结果的事件

策略：

1 避免风险：改变性能或功能需求

2 转移风险：把风险分配到其他系统中，或者购买保险

3 假设风险：接受并用项目资源控制风险

找出图 3.23 和 3.24（139 页）的关键路径。

见上面算法描述。

Chapter04

需求的含义是什么? (143 页)

需求是对来自用户的关于软件系统的期望行为的综合描述，涉及系统的对象、状态、约束、功能等。

需求作为一个工程，其确定需求的过程是什么？（144 页 图 4.1）

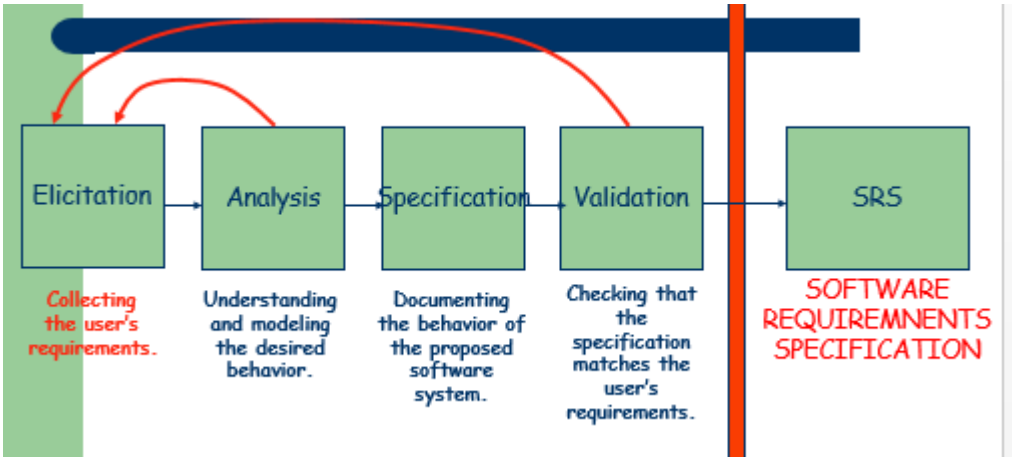
原始需求获取（问题定义来自用户）：收集用户需求

问题分析：理解和建模期望的行为

规格说明草稿：文档化要开发的软件系统的行为

需求核准：检查我们的规格说明是否与用户的需求匹配

正式需求文档：SRS 软件需求规格说明书



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）

可以分类为以下为三类：

1 绝对要满足的需求（必须的）

2 非常值得要的但并非必须的需求（值得要的）

3 可要可不要的需求（可选的）

之所以有上述分类的原因：有的软件开发项目受到了时间和资源的限制

//如何使需求变得可测试？（151-152 页, sidebar4.4）

需求文档分为哪两类？（153 页）

1 需求定义：完整罗列了客户上想要的每一件事情

2 需求规格说明：重述了技术术语与符号的定义

3* 配置管理：使软件过程各阶段文档保持一致的系列过程

什么是功能性需求和非功能性需求/质量需求？

功能性需求：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

质量需求 /非功能性需求：描述软件方案必须具备的某些质量特征

//设计约束？过程约束？（149 页）

需求的特性？（正确性、一致性、完整性）（155 页）

1 正确性 2 一致性 3 完备性（完整性） 4 无二义性 5 可行性 6 相关性 7 可测试的 8 可跟踪的

了解 DFD 图的构成及画法（172 页）

构成：

→：数据流。数据流是数据在系统内传播的路径，因此由一组成分固定的数据组成。如订票单由旅客姓名、年龄、单位、身份证号、日期、目的地等数据项组成。由于数据流是流动中的数据，所以必须有流向，除了与数据存储之间的数据流不用命名外，数据流应该用名词或名词短语命名。

□：数据源（终点）。代表系统之外的实体，可以是人、物或其他软件系统。

○：对数据的加工（处理）。加工是对数据进行处理的单元，它接收一定的数据输入，对其进行处理，并产生输出。

═：数据存储。表示信息的静态存储，可以代表文件、文件的一部分、数据库的元素等。

画法：

(一) 确定系统的输入输出

由于系统究竟包括哪些功能可能一时难于弄清楚，可使范围尽量大一些，把可能有 的内容全部都包括进去。此时，应该向用户了解“系统从外界接受什么数据”、“系 统向外界送出什么数据”等信息，然后，根据用户的答复画出数据流图的外围。

(二) 由外向里画系统的顶层数据流图

首先，将系统的输入数据和输出数据用一连串的加工连接起来。在数据流的值发生 变化的地方就是一个加工。接着，给各个加工命名。然后，给加工之间的数据命名。 最后，给文件命名。

(三) 自顶向下逐层分解，绘出分层数据流图

对于大型的系统，为了控制复杂性，便于理解，需要采用自顶向下逐层分解的方法 进行，即用分层的方法将一个数据流图分解成几个数据流图来分别表示。

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）

抛弃式原型：仅用于了解问题、探索可行性，并不打算用来作为将来实际提交系统的一部分，而是用完扔掉。

演化型原型：用于了解问题，并作为将来准备提交的系统的一部分。

// 用 DFD 图简单描述 ATM 机的工作原理（主要功能和数据流）（220 页习题 7）

Chapter05

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）

软件体系结构：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：一种针对单个软件模块或少量模块而给出的一般性解决方案，它提供较低层次的设计决策。

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：将需求中的问题描述转变成软件解决方案的创造性过程。

概念设计：确切地告诉客户系统要做什么（软件系统和功能）

技术设计：告诉系统构建人员怎样做（软件功能和接口的实现方法）

三种设计层次及其关系？（229 页）

1 体系结构设计：由软件需求中的系统能力与系统部件关联起来而得到软件整体结构的过程

2 代码设计：各个部件（模块）的算法、数据结构的设计

3 运行设计：最底层设计，包括内存分配、数据格式、位模式等

4 联系： 1） 自顶而下工作：按照体系结构、代码、运行的三个步骤设计

2） 体系结构设计≈系统设计 代码+运行设计≈程序设计

3） 重复设计

什么是模块化？什么是抽象？（238 页）

模块化：一种把系统中各不相关的部分进行分离的原则，模块有清晰的输入和输出，设计明确，功能独立，可以做独立测试。//模块有清晰的输入和输出，设计目的明确，功能独立，可以做独立检测

抽象：对细节的隐藏

论述设计用户界面应考虑的问题。（242 页）

1 设计界面要注意解决的要素

1） 寓意/比喻 可以认识和学习基本术语、图像和概念

2） 思维模型 数据、功能、任务和角色的组织和表示

3） 模型的导航规则 如何在数据、功能、活动和角色中移动

4） 外观 系统向用户传输信息的外观特性

5） 感觉 向用户提供有吸引力的体验交互技术

2 文化差异问题：必须考虑用户的信仰、价值观、习俗等，而使用国际化的设计

3 用户爱好问题：制作可选的界面以供选择

5.5 节----模块独立性----耦合与内聚的概念及各个层次划分？（248----xxx 页）

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）

耦合：两个软件部件之间的相互关联程度

耦合的六个层次：

非直接耦合：模块相互之间没有信息传递

数据耦合：模块间传递的是数据

特征耦合：模块间传递的是数据结构

控制耦合：模块间传递的是控制量

公共耦合：不同模块访问公共数据

内容耦合：一个模块直接修改另一个模块（A 模块直接调用 B 模块的私有数据，或直接转移到 B 模块中去）

内聚：软件部件内部各组成成分的关联程度

内聚的七个层次：

偶然性内聚：不相关的功能、过程、数据等出现在同一个部件中

逻辑性内聚：逻辑上相关或相似的功能或数据放置在同一个部件内

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

通讯性内聚：各个部分访问共享数据

顺序性内聚：各部分有输入输出关系

功能性内聚：各部分组成单一功能

Chapter06

//什么是面向对象？（286 页）OO 有几个基本特征？如何使用高级语言实现这些基本特//征？了解并使用高级语言的 OO 基本编程方法和技巧。（286-291）

//什么是设计模式？

//OO 设计的基本原则？

OO 开发有何优势？（291 页）

（相对于传统的过程式开发）1 语言的一致性 2 过程的一致性

OO 开发过程有几个步骤？（292 页）

1 面向对象需求 2 面向对象上层设计 3 面向对象下层设计 4 面向对象编程 5 面向对象测试

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）

组成：

1 用例：描述一个系统应当展示的部分功能 

2 执行者：与系统交互的实体称作执行者 

3 使用：一个已经定义的用例的重用 

4 扩展：扩展一个用例用来说明一个不同或者更深的视角 

用例图、类图等对面向对象的项目开发的意义是什么？

//用例图的作用：1 阐明需求 2 对寻找需求错误有帮助 3 需求本身是复杂且难以描绘的

UML 是用来描述面向对象解决方案的一套很受欢迎的设计表示法。经过剪裁，它可以适应于不同的开发情形和软件声明周期。UML 可以用来可视化、说明或文档化软件设计。它在描述不同的设计选择，直至最终对设计工件进行文档化时是特别有用的。

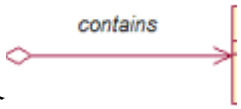
熟悉类图中各个类之间的基本关系分类（303-305）

1 继承/泛化 

父类泛化子类，允许子类继承父类的属性与行为

2 关联

两个类有语义联系。分类：双向关联，单向关联，自关联，多重性关联



3 聚合

表示部分与整体的关系。聚合关系中的成员对象是整体对象的一部分，但是成员对象可以脱离整体对象存在。类的聚合关系是用带空心菱形的直接表示。

4 组合

组合关系表示部分与整体的关系。但是组合关系中整体对象可以控制对象的生命周期，一旦整体对象不存在，成员对象也将不存在，两者是共生关系。类的组合关系是用带实心菱形的直线表示。

5 依赖

是一种使用关系，大多数情况下依赖关系体现在某个类的方法使用另一个雷的对象作为参数。类的依赖关系是用带箭头的虚线表示，由依赖的一方指向被依赖的一方。

6 接口与实现 <<Interface>>

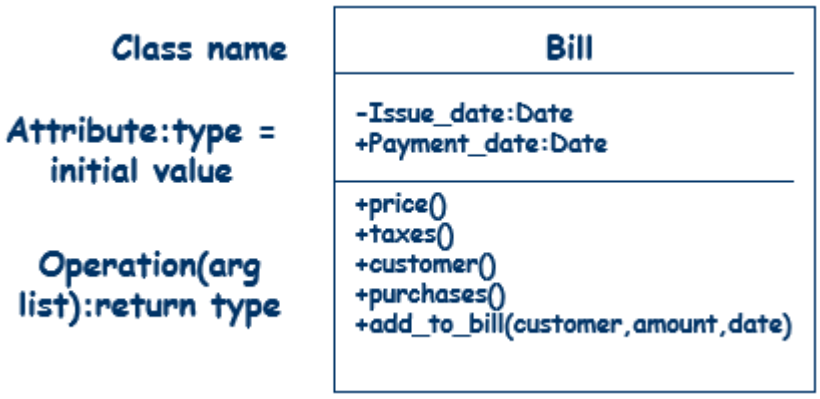
接口中通常没有属性，而且所有的操作都是抽象的。接口之间也有继承和依赖关系，但是接口设计很重要的一种关系是实现关系，即类实现了接口。该关系是用带空心三角形箭头的虚线表示。

熟悉类图等的组成和画法（300-308 页）

组成：

类名：确定一个类

属性/操作



了解 UML 其他图的基本用途。

1 类描述模板：对每个类进行更详细的面书上，模板重复类图中的某些信息：类在整个层次中的位置（根据继承的深度）、输出控制、基数（即类中可能有多少对象），以及和其他类的关联。

2 UML 包图：使得我们可以将系统看成是由包组成的一个较小的集合而每个包都可以展开为一个由类组成的很大的集合。包图展示了不同包中的类之间的依赖关系。

3 交互图：为了给动态行为建模，使用交互图来描述对象图和实现从操作和行为。使用用例图创建交互图，来说明该用例中那些典型的系统和外部的参与者的消息交换。交互图有四种，顺序图、通信图、状态图、活动图。

1）顺序图展示了活动或行为发生的顺序。对象使用方框表示，位于一条垂直线的顶端，这条垂直线就是该对象的生命线。

2）通信图描述了对象之间的消息顺序，但是它是基于对象模型之上的，使用对象之间的链接作为隐含的信道。

3）状态图展示了对象可能具备的状态、触发状态改变的事件，以及每次状态改变所导致的动作。

4）活动图来为类中的过程流或活动流建模。当根据条件决定调用哪个活动时，用决策节点来表示这种选择。

Chapter07

//为什么说编码工作是纷繁复杂甚至令人气馁？（337 页）

一般性的编程原则应该从哪三个方面考虑？（340-344 页）

1 控制结构：最高层次的程序指引；在隐式调用或面向对象设计中，控制基于系统状态和变量更变；另外设计都要使程序结构反映设计的控制结构，1）重构辅助理解，2）模块化使代码可解读，3）一般性使得控制结构更加通用，4）部件之间的耦合或依赖关系必须是可见的。

2 算法：1）追求效率可能有潜在成本；2）追求效率可能会牺牲代码的清晰性和正确性；3）了解所使用的编译器如何优化。

3 数据结构：1）简化程序；2）使用数据结构决定程序结构

//论述编码阶段实现某种算法时所涉及的问题。（342 页）

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）

1 其他程序注释：1）阶段注释（而非单行注释）；2）代码改变伴随着注释的更新；3）注释应该有新信息；4）像写代码一样写注释，而非之后补注释

2 有意义的变量名和语句标记

3 安排格式以增强理解

4 数据文档化

什么是极限编程(XP)？以及派对编程？(357 页)

极限编程是一种轻量级的软件开发论，属于敏捷开发方法。其主要特征是适应环境变化和需求变化，充分发挥开发人员的主动精神。极限编程承诺降低软件项目风险，改善业务的变化反应能力，提高开发期间的生产力，为软件开发过程增加乐趣。

派对编程属于主要的敏捷开发方法，其开发方式是两个程序员共同开发程序，且角色分工明确。一个负责编写程序，另一个负责复审与测试。两人定期交换角色。

Chapter08

// 产生软件缺陷的原因？（365 页）

//将软件缺陷进行分类的理由？（367 页）

几种主要的缺陷类型？（367-368 页）

1 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

2 计算和精度缺陷：算法或公式在变成实现时出现错误或最终结果达不到精度要求

3 文档缺陷：文档与程序实际做的事情不符

4 压力或过载缺陷：运行程序时，填充数据结构时超过能力

5 能力或编写缺陷：系统活动到达指定的极限时系统性能会变得不可接受

6 时序性缺陷：几个同时执行的或按仔细定义的顺序执行的进程的协调不恰当

7 性能缺陷：系统不能以需求规定的速度执行

8 恢复性缺陷：当系统遇到失效时，不能表现得像设计人员希望的或客户要求的那样

9 硬件和系统软件缺陷：提供的硬件和系统软件实际上并没有按照文档中的操作条件和步骤运作

10 代表的标准和规程缺陷：代码未能遵循组织机构的标准和过程

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。

测试的各个阶段及其任务？（372 页图 8.3）

1 单元测试：将每个程序构件与系统其它构件隔离，对其本身测试，针对预期的输入类型，构件能否适当地运行

2 集成测试：验证系统构件是否能够按照系统和程序设计规格说明中描述的那样共同工作

3 功能测试：对系统进行评估，以确定集成的系统是否确实执行了需求规格说明中描述的功能

4 性能测试：将系统与这些软件和硬件需求的剩余部分进行比较

5 验收测试：根据客户的需求描述对系统进行检查

6 安装测试：确保系统将按照它应该的方式来运行

// 测试的态度问题？（为什么要独立设置测试团队？）(373 页)

测试的方法----黑盒、白盒的概念？（374）

黑盒：将其看作一个不了解其内容的黑盒，仅仅测试测试对象的功能。也就是说，向黑盒提供输入数据，并记录产生的输出。测试的目标是确保每一种输入都被提交，并且观察到的输出与预期的输出相匹配。

白盒：将测试对象看成是一个透明盒，然后根据测试对象的结构用不同的方式来进行测试。

什么是单元测试？ 什么是走查和检查？（376 页）

1 单元测试：将每个程序构件与系统其它构件隔离，对其本身测试，针对预期的输入类型，构件能否适当地运行

2 走查：程序员向评审小组提交代码及其相关文档，然后评审小组评论它们的正确性

3 检查：评审小组按照一个事前准备好的关注问题清单来检查代码和文档

黑盒白盒方法各自的分类？*测试用例的设计和给出方法*（结合补充材料）

黑盒白盒方法的分类，各种覆盖方法等。（课件和补充课件）

黑盒测试方法：

1 等价分类法：把被测程序的输入域划分为若干等价类，把漫无边际的随机测试变成有针对性的等价类测试。分为弱一般等价类，强一般等价类，弱健壮等价类。

2 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。而把测试值选在等价类的边界上，往往有更好的效果。同时适用于考察程序的输出值边界。

3 错误猜测法：猜测被测程序中哪些地方容易出错，并据此设计测试用例。其更多地依赖与测试人员的直觉和经验。

4 因果图法：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情况。因果图是一种简化了的逻辑图，能直观地表明程序输入条件（原因）和输出动作（结果）之间的相互关系。

白盒测试方法：

1 逻辑覆盖法：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方法：

1）语句覆盖：使被测程序的每条语句至少执行一次

2）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

3）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

4）条件组合覆盖：最强的一种覆盖。它与条件覆盖的区别是，它不是简单地要求每个条件都出现“真”“假”两种结果，而是要求让这些结果的所有

可能都至少出现一次

2 路径测试法：是借助程序图设计测试用例的一种白盒方法。在本方法中，测试用例是基于流程图来设计的。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，包括了所有的判定节点。

1）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。

2）边覆盖：程序的测试路径至少经过程序图中每条边一次。它相当于逻辑覆盖法的判定覆盖。

3）完全覆盖：同时满足结点覆盖和边覆盖的覆盖。

4）路径覆盖：程序图中每条路径都至少经过一次。

考虑如何面对一个命题，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

1 在集成测试及其后的测试阶段，除去很小的程序，都采用黑盒方法。其策略包括：

（1）用边值分析法和（或）等价分类法提出基本的测试用例；

（2）用猜错法补充新的测试用例；

（3）如果在程序的功能说明中含有输入条件的组合，宜在一开始就用因果图法，然后再按以上（1）、（2）两步进行。

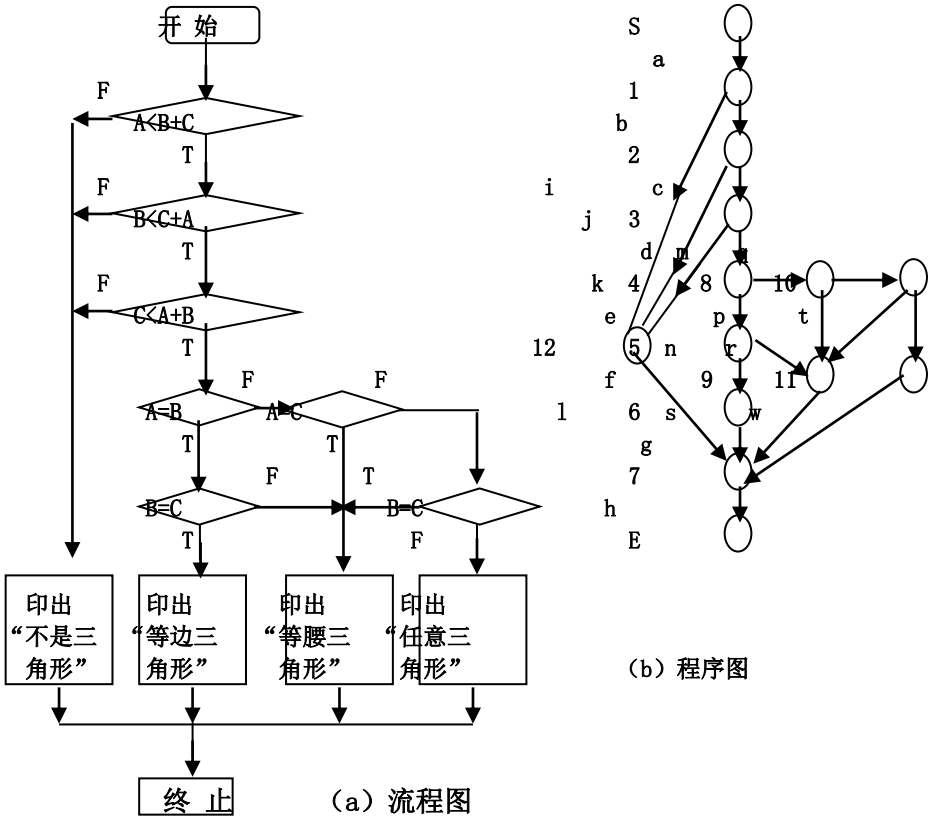
2 单元测试的设计策略稍有不同。因为在为模块设计测试用例时，可以直接参考模块的源程序。所以单元测试的策略，总是把白盒法与黑盒法结合运用。具体的作法又有两种：

（1）先仿照上述步骤用黑盒法提出一组基本的测试用例，然后用白盒法作验证。如果发现用黑盒法产生的测试用例未能满足所需的覆盖标准，就用白盒法增补新的测试用例来满足它们覆盖的标准应根据模块的具体情况确定。对可靠性要求较高的模块，通常应满足条件组合覆盖或路径覆盖标准。

（2）先用白盒法分析模块的逻辑结构，提出一批测试用例，然后根据模块的功能说明用黑盒法进行补充。

在一般情况下，小型单元测试应该大多以白盒方法为主。

3 课堂练习：三角形程序设计一组测试用例。这个程序的功能是，读入代表三角形边长的 3 个整数，判断它们能否组成三角形。如果能够，则输出三角形是等边、等腰或任意三角形的识别信息。



（a）流程图　　（b）程序图

本例将先用黑盒法设计测试用例，然后用白盒法进行检验与补充。

第一步：运用等价分类法划分与定义等价类，然后用边值法和猜错法补充。其结果如图 5.7 所示。

等价分类法

有效等价类		无效等价类	
输入 3 个正整数：		含有零数据	⑨
3 数相等	①	含有负整数	⑩
3 数中有 2 数相等	A、B 相等② / B、C 相等③ / C、A 相等④	少于 3 个整数	⑪
3 数不相等	⑤	含有非整数	⑫
2 数之和不大于第 3 数	最大数为 A⑥ / 最大数为 B⑦ / 最大数为 C⑧	含有非数字符	⑬

边值法：　2 数之和等于第 3 数　⑭

猜错法：　输入 3 个零　⑮

　　输入 3 个负数　⑯

图 5.7　用黑盒法分析程序的输入

第二步：选择测试数据，得出 22 个基本的测试用例，如表 5．3 所示。

表 5.3 三角形程序的测试用例

序号	测试内容	测试数据			期望结果
		a	B	C	
1	等边	5，5，5			等边三角形
2	等腰	4，4，5	5，4，4	4，5，4	等腰三角形
3	任意	3，4，5			任意三角形
4	非三角形	9，4，4	4，9，4	4，4，9	
5	退化三角形	8，4，4	4，8，4	4，4，8	
6	零数据	0，4，5	4，0，5	4，5，0	不是三角形
7		0，0，0			
8	负数据	-3，4，5	3，-4，5	3，4，-5	
9		-3，-4，-5			
10	遗漏数据	3，4，--			
11	非整数	3.3，4，5			运行出错
12	非数字符	A，4，5			(类型不符)

第三步：用白盒法检验第二步产生的测试用例（见表 5.4）。结果表明，只须使用 22 个例子中的前 8 个，就能满足对程序图的完全边覆盖。可见对本例来讲，用黑盒法设计的测试用例已经够用，不必再进行补充了。[例 5．3 完]

表 5.4　被测试数据覆盖的边和结点

序号	测试数据	覆盖结点	覆盖的边
1	5，5，5	1，2，3，4，5，6，7	abcdefgh
2a	4，4，5	1，2，3，4，5，9，7	abcdensh
2b	5，4，4	1，2，3，4，8，10，9，7	abcdmqrsh
2c	4，5，4	1，2，3，4，8，9，7	abcdmpsh
3	3，4，5	1，2，3，4，8，10，11，7	abcdmqtwh
4a	9，4，4	1，12，7	ailh
4b	4，9，4	1，2，12，7	abjlh
4c	4，4，9	1，2，3，12，7	abcklh

4 课堂练习 2：

例 5.1 某城市的电话号码由 3 部分组成。这 3 个部分的名称与内容分别是:

地区码：空白或 3 位数字；

前　缀：非’0’或’1’开头的 3 位数字；

后　缀：4 位数字。

假定被测程序能接受一切符合上述规定的电话号码，拒绝所有不符合规定的号码，就可用等价分类法来设计它的测试用例。

第一步：划分等价类。表 5．1 列出了划分的结果，包括 4 个有效等价类，ll 个无效等价类。在每一等价类之后均加有编号，以便识别。

表 5.1　电话号码程序的等价类划分

输入条件	有效等价类	无 效 等 价 类
地 区 码	空白①，3 位数字②	有非数字字符⑤，少于 3 位数字⑥，多于 3 位数字⑦，
前　　缀	从 200 到 999 之间的 3 位数字③	有非数字字符⑧，起始位为′0′⑨，起始位为′1′⑩，少于 3 位数字⑪，多于 3 位数字⑫，
后　　缀	4 位数字④	有非数字字符⑬，少于 4 位数字⑭，多于 4 位数字⑮

第二步：确定测试用例。表中有 4 个有效等价类，可以公用以下两个测试用例：

测试数据	测试范围	期望结果
（　）276－2345	等价类①、③、④	有效
（635）805－9321	等价类②、③、④	有效

对 ll 个无效等价类，应选择 ll 个测试用例。例如前 3 个无效等价类可能使用下列的 3 个测试用例：

测试数据	测试范围	期望结果
（2OA） 123-4567	无效等价类⑤	无效
（33 ） 234-5678	无效等价类⑥	无效
（7777）345－6789	无效等价类⑦	无效

（可以看成是无效等价类⑮吗？也可以,因为作者对无效等价类的划分已经考虑被测程序的扫描习惯问题了,但系统一般只保持其中一种习惯，都可以发现错误）

(或者,干脆给出第 16 个无效等价类:电话号码超过 10 位)

后 8 个无效等价类的测试用例留给读者做练习。这样，本例的 15 个等价类将至少需要 13 个测试用例。

集成测试及其主要方法的分类？（390-392）（驱动，桩的概念）

1 集成测试：验证系统构件是否能够按照系统和程序设计规格说明中描述的那样共同工作

2 分类：

1）自底向上测试：每一个出于系统层次中最低层的构件首先被单独测试，接着测试那些调用了前面已测试构件的构件。反复地采用此方法，知道所有的构件都被测试完毕。**构件驱动程序**：调用特定构件并向其传递测试用例的程序。

2）自顶向下测试：顶层构件，通常是一个控制构件，是独立进行测试的。然后，将被测构件调用的所有构件组合起来，作为一个更大的单元进行测试。重复执行这种方法，直到所有的构件都被测试。**桩**：一种专用程序，用于模拟（测试时）缺少构件时的活动。桩应答调用序列，并传回输出数据，使测试过程得以继续运行。

3)一次性测试：当所有构件都分别经过测试，再讲它们合在一起作为最终系统进行测试，看看这个系统是否能一次运行成功。

4）三明治测试：将系统看成三层，目标层处于中间、目标上面有一层，目标下面有一层，在顶层使用自顶而下方法，而在较低的层次使用自底向上方法。测试集中于目标层，目标层是根据系统特性和构件层次结构来选择的。

传统测试和 OO 测试有何不同？OO 测试有何困难？（398-399 页）

传统测试：当系统发生了改变时，我们可以测试改变是否正确，并且用已有的测试数据来验证另外的、剩余的功能是否一样。

面向对象测试：

1 当子类用一个同名的局部定义的方法替代一个继承的方法时，必须对重载的子类重新进行测试，并且可能要用不同的测试数据集合。

2 单元测试较为容易，但集成测试一定会涉及面更广。因为封装作为 OO 的一个好的特性需要更广泛的集成测试。

面向对象测试的困难：

1 需求可能在需求文档中凑数，但是，几乎没有什么工具支持表述为对象和方法的需求的验证。

2 同样，帮助测试用例生成的大多数工具并不能处理用对象和方法表述的模型。

3 大多数源代码测度的定义是针对过程代码，而不是针对对象和方法。

4 由于对象的交互是复杂性的根源，代码覆盖测度和工具在面向对象测试中的作用要比在传统测试中的小。

// 测试计划涉及的几个步骤？（400 页）　　（了解）

Chapter09

系统测试的主要步骤及各自含义？（420 页，图 9.2）

1 功能测试：检查集成的系统是否按照需求中指定的那样执行它的功能。

2 性能测试：将集成的构件与非功能系统需求进行比较。这些需求包括安全性、精确性、速度和可靠性，它们约束了系统功能的执行方式。

3 验收测试：使客户和用户能确定构建的系统是否满足了他们的要求，保证构件的系统就是客户想要的系统。

4 安装测试：使得用户能够执行系统功能并记录在实际环境中可能引起的的其他问题。

//什么是系统配置？软件配置管理？基线？（423 页）（或见课件）

// 什么是回归测试？（425 页）

功能测试的含义及其作用？（430 页）

含义：检查集成的系统是否按照需求中指定的那样执行它的功能。

作用：有更高的概率发现缺陷，因为从逻辑上讲，相对于一个大的构件集，在一个小的构件集中更容易找出问题的原因。

功能测试的基本指导原则？（431）

1 较大的查错概率

2 独立的测试团队

3 了解预期的输出结果

4 对合法与非法的输入都予以测试（结社是弱健壮等价类）

5 不能仅仅为了测试的方便而修改系统

6 停止测试应该有前提条件

性能测试的含义与作用？（436 页）

含义：将集成的构件与非功能系统需求进行比较。这些需求包括安全性、精确性、速度和可靠性，它们约束了系统功能的执行方式。

作用：性能测试根据客户规定的性能目标来测量，性能目标表示为非功能需求。可根据对用户命令的响应速度、结果的精确度、数据的可访问性等用户性能规定，检查计算的效果。

性能测试的主要分类？（436 页）

1 强度测试：当系统在短时间内到达其压力极限时，对系统进行的测试。

2 容量测试：验证系统处理巨量数据的能力。

3 配置测试：分析需求中指定的各种软件和硬件配置。

4 兼容性测试：当一个系统与其他系统交互时，检查接口功能是否按照需求执行。

5 回归测试：当正在测试的系统要替代一个现有系统的时候，保证测试的新系统的表现至少与老系统一样好。

6 安全性测试：确保安全性需求得到满足。它测试与数据和服务的可用性、完整性和机密性相关的系统特性。

7 计时测试：评估设计对用户的响应时间和一个功能的执行时间的相关需求。

8 环境测试：考察系统在安装场所的执行能力。

9 质量测试：评估系统的可靠性、可维护性和可用性。

10 恢复测试：强调的是系统对出现故障或丢失数据、电源、设备或服务时的反应。

11 维护测试：为帮助人们发现问题的根源提供诊断工具和过程的需要。

12 文档测试：确保我们已经编写了必需的文档。

13 人为因素测试：检查涉及系统用户界面的需求。

// 什么是可靠性、可用性和可维护性？（438 页）

确认测试，确认测试分类？（基准测试和引导测试）（447-448 页）

含义：使客户和用户能确定我们构建的系统真正满足了他们的需要和期望。

分类：

1 基准测试：客户准备一组代表在实际安装后系统运作的典型情况的测试用例。针对每一个测试用例，客户评估系统的执行情况。由实际用户或执行系统功能的专门小组来负责进行基准测试。基准测试通常用在客户有特殊的需求时进行。

2 引导测试/试验性测试：在试验的环境中安装系统。用户假系统已经永久安装的情况下执行系统。试验性系统以来传统的日常工作来测试所有的功能。客户通常准备一个建议的功能列表，每位用户都设法在日常的工作过程中包含这些功能。

3 并行测试：当一个新系统要替代现有系统，或者该新系统是分阶段开发的一部分，使用并行测试，新系统与先前版本并行运转，用户逐渐地适应新系统，但是继续使用与锡系统的功能同呃老系统。这种逐步过渡的方法使得用户能够将新系统和老系统进行比较和对照。

什么是 alpha 测试？β 测试？（448 页）

1 α 测试：在向客户发布一个系统之前，先让来自自己组织机构或公司的用户来测试这个系统，在客户进行实际的试验性测试之前先“试验”这个系统，这样的内部测试称为 α 测试。

2 β 测试：当要向大范围的客户发布系统时，客户进行的试验性测试称为β测试。

什么是安装测试？（450 页）

在用户的环境中安装系统，使得用户能够执行系统功能并记录在实际环境中可能引

起的的其他问题。

注意：

每一章节的开头中，大的概念性问题是如何引入的？，其讨论请见课件。

试卷答题须知：

1．软件工程课程出题覆盖范围比较广泛，考察手段有灵活和多样化特点。

2．有考察学生软件工程运用能力的题目存在，难题占的分数很少。其他很多是送分的题目。

3．有难度的题目占极少部分，可以根据时间安排，先回答别的题目。

4．所有题目都要写到试卷的指定位置，以免流水阅卷时有遗漏。