**英文版《软件工程》教学内容回顾2016上**

**(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**

**1. SE的定义、目的、方法及作用（P2 / P16）**

定义：在将有关软件开发与应用的概念科学体系化的基础上，研究如何有计划、有效率、经济的开发和利用能在计算机上正确运行的软件的理论和技术的工程方法学，一些开发和维护软件的方法、过程、原则。

目的：在给定成本、进度的前提下，开发出具有适用性、有效性、可修改性、可靠性、可理解性、可维护性、可重用性、可移植性、可追踪性、可互操作性和满足用户需求的软件产品。追求这些目标有助于提高软件产品的质量和开发效率，减少维护的困难。

方向：面向对象模式、结构化模式、基于过程的模式等。

作用：付出较低的开发成本，达到要求的软件功能，取得较好的软件性能，开发的软件易于移植，需要较低的维护费用，能按时完成开发工作，及时交付使用。（是一个系统工程，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）

**2.//开发模式（paradiam）（P4）**

软件开发的全部过程、活动和任务的结构框架，它能直观的表达软件开发全过程，明确要完成的主要活动、任务和开发策略。

**3.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（error）：当人们在进行软件开发活动过程中出的错（在软件生产过程中人为产生的错误：需求说明的错误、代码中的错误等）

缺陷（fault故障）：在功能实现过程中产生的问题。（一个缺陷对应多个错误，静态存在）

失败（failure失效）：指系统违背了它应有的行为。（相对于系统指定行为的偏离：动态存在的）

联系：故障是系统的内部视图，这是从开发人员的角度看待系；而失效是系统的外部视图，它是用户所看到的问题。并非每一个故障都对应于一个失效。

**4.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：用户从外部特征看软件具有足够的功能并且易于学习和使用，就说明软件具有高质量，开发者从内部特征来看错误数量和类型来判断软件质量的高低。

过程的质量：有很多活动会影响到最终的产品质量。只要有活动出了差错，产品的质量就会受到影响。

商业环境背景下的质量（商业质量）：将技术价值和商业价值统一起来。

**5.//软件系统的系统组成（P16）**

活动和对象：活动（activity）是发在系统中的某些事情，通常描述为由某个触发器引发的事件，活动通过改变某一特性将一个事物转变成另一个事物。活动中涉及的元素被称为对象（object）或实体（entity）。通常，这些对象以某种方式相互联系。

关系和系统边界。

**6.现代软件工程大致包含的几个阶段及各个阶段文档（P23-24）**

需求分析和定义（项目计划、可行性研究报告、软件需求规格说明书SRS）

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、数据描述文档）

编写程序/程序实现（源程序、注释）

单元测试（由模块测试和性能测试产生的测试报告）

集成测试（按照结构图进行测试产生的测试报告）

系统测试（按照SRS对系统总体功能进行测试产生的测试报告）

系统提交（交付的产品，用户手册或操作手册）

系统维护（修改软件的过程，为满足改错或新需求产生的维护报告）

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

商用产品投入市场时间的紧迫性；计算技术在经济中的转变：更低的硬件成本，更高的开发、维护成本；功能强大的桌面计算的可用性；广泛的局域网和广域网；面向对象技术的采用及其有效性；使用窗口、图标、菜单和指示器的图形用户界面；软件开发瀑布模型的不可预测性。

**8.什么是抽象？（30页）**

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。

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

定义：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（在一组活动上保持一致性和结构性。一致性和结构性可以使我们知道是否已经做好了工作，还能使别人以同样地方式做工作，因而具有相对通用性）；
2. 指导性（分析、检查、理解、控制、改善活动）。
3. 可以将经验传授给他人。

阶段：需求分析和定义、系统设计、程序设计、编写程序、单元测试、集成测试、系统测试、系统交付、维护。

**10.什么是重用等软件工程主要概念？（34页）**

重（复）用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。（这里的重用不仅仅是代码的重用。）

**Chaoter02**

1.什么是软件过程？软件过程的重要性是什么？ （P45-46）

**2.瀑布模型及各阶段文档，优缺点？（P49）**

优点：用里程碑和各个阶段的提交文档描述软件开发过程；它的简明性使得人们能够容易的向不熟悉软件开发的顾客清楚地作出解释；是其他模型的基础。

缺点：面临变动时，该模型无法处理实际过程中的重复开发的问题；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）**

含义：系统被设计成部分提交，每次用户只能得到部分功能，其他部分处在开发过程中。

分类：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）、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：培训过程可以很早开始，可以使得开发人员观察用户的反应，为后续版本提供更好的建议；可以为那些从未实现的功能提前开发市场；不断发布新的版本使得开发人员可以快速的修正系统中的问题；不同的开发版本可以针对不同的专业领域。

**5.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。

含义：操作概念，软件需求，软件设计，系统测试。

**\*P80--81页 习题2、3（针对本章描述的每一种过程模型，讨论使用该模型的优点和缺点分别是什么，讨论该模型是如何处理开发后期重要的需求变化的。）**

**7.//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）**

阶段开发模型和螺旋模型。

**8.什么是UP， RUP?**

统一过程（UP）：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。

统一开发过程（RUP）：一个面相对象且基于网络的程序开发方法论。描述了如何有效地利用商业的可靠的方法开发和部署软件。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**

**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。进度还描绘这些活动之间的交互，并估算每项任务或活动将花费的时间。因此，进度是一个时间线，说明活动将在什么时候开始、在什么时候结束以及相关的开发产品将在什么时候完成。

活动：项目的一部分，它在一段时间内发生，具有开始和结束。

里程碑：活动的完成——某一特定的时刻，专门指定的活动的结束。

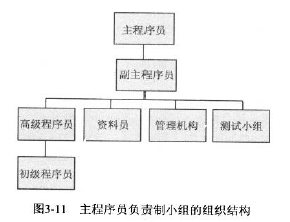
**2.如何计算软件项目活动图的关键路径？（习题2，3）冗余时间？最早和最迟开始时间（课堂习题讲解）**

**3.软件团队人员应该具备的能力是什么？（96页）**

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。

**4.软件项目组织的基本结构？（101页）**

主程序员负责制组：



主程序员：总体负责系统的设计和开发，其他小组成员向该主程序员汇报，主程序员对每一个决定有最终决策权。主程序员监督所有其他小组成员、设计所有程序、把代码开发分配给其他小组成员。

副主程序员（后备程序员）：在必要时替代主程序员。

资料员：负责维护所有的项目文档，编译和链接代码，并对提交的所有模块进行初步测试。

在有着高度确定性、稳定性、一致性和重复性的项目中，使用像主程序员负责制这样的等级组织结构会更有效。当项目中涉及大量不确定性时，采用更为民主的方法更好。

忘我方法：不是把责任放在单个人身上，而是让每个人平等的担负责任。而且，过程与个人是分开的：批评是针对产品或结果的，并不涉及个人。忘我小组结构是民主式的，不论讨论的是设计问题还是测试技术，小组成员投票产生决策。

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

专家估算法：工作量估算方法依赖于专家判断，预测的精确性基于估算者的能力、经验、客观性和洞察力。

算法估算法：E=（a+bS^c）m（X）其中S是估算的系统规模，而a、b和c是常量。X是从x1到xn的一个成本因素的向量，m是基于这些因素的一个调整因子。

**6.试述MIANMO模型的三个阶段基本工作原理或含义。（111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模型知之甚少，因此用应用点（其创建者对它的命名）来估算规模。

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，使用功能点对模型进行测量。

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来进行规模估算，而且可以较为轻松的估算很多成本因素。

**7.什么是软件风险？主要风险管理活动？有几种降低风险的策略？（119、122页）**

软件风险：一种具有负面后果的、人们不希望发生的事件。

风险管理活动：风险评价（识别、分析、优先级分配）；风险控制（降低、管理计划、化解）。

降低风险的策略：通过改变性能或功能需求避免风险；通过把风险分配到其他系统中或者购买保险以便在风险成为事实时拟补经济上的损失从而转移风险；假设风险会发生接受并用项目资源控制风险。

**8.找出图3.23和3.24（139页）的关键路径。**

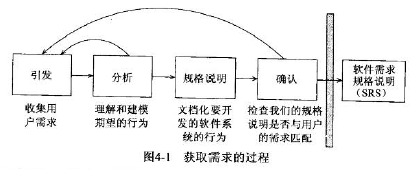
**Chapter04**

**1.需求的含义是什么? (143页)**

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

需求就是对期望的行为的表达。需求处理的是对象或实体，他们可能处于的状态，以及用于改变状态或对象特性的功能。需求阶段的目标是理解客户的问题和需要。因此，需求集中于客户和问题，而不是解决方案和实现。我们通常说，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）**



**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）**

（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）。

例如，信用卡记账系统必须能够列出最近的费用，将它们加起来并要求在某个日期前支付，这是必须的需求。但是，该记账系统也可能按照购买类型区分费用，以帮助购买者理解购买的模式，这种购买类型分析是值得要的需求，但是可能不是必须的需求。最后，记账系统可能要求用黑颜色来打印贷方账目而用红颜色打印借方账目，这种需求是有用的，但它是可选的需求。当软件开发项目受到时间或资源的限制时，这种做法也是很有用的。如果系统的成本太高或者开发的时间太长，就可以去掉可选需求，并对值得要的需求进行分析，考虑是去掉还是延迟到后续版本中。

**4. //如何使需求变得可测试？（151-152页, sidebar4.4）**

（1）指定每个副词和形容词的定量描述，这样限定词的含义就清楚、明了了；（2）用特定实体的名称替换代名词；（3）要确保在需求文档的某个地方，准确地定义每个名词。

**5.需求文档分为哪两类？（153页）**

需求定义：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。

**6.什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）**

功能性需求：根据要求的活动来描述需要的行为。定义问题解决方案空间的边界。描述系统内部功能或系统与外界环境的交互作用。

质量需求/非功能性需求：描述一些软件解决方案必须拥有的质量特性。

设计约束：已经做出的设计决策或限制问题解决方案集的设计决策。

过程约束：对用于构建系统的技术和资源的限制。

**7.需求的特性？（正确性、一致性、完整性）（155页）**

正确性、一致性、无二义性、完备性、可行性、相关性、可测试性、可跟踪性。

**8.了解DFD图的构成及画法（172页）**

数据流图（DFD）：建模功能以及从一个功能到另一个功能的数据流。一个泡泡表示一个加工或功能，它转换数据。箭头表示数据流，其中进入泡泡的箭头表示其功能性的输入，从泡泡出去的箭头表示其功能的输出。前面步骤的结果流入后面的步骤，单个计算使用之外的持久性数据保存在数据存储中，它是一个正式的库或信息库，表示为两个平行的条。数据源或者数据接收器表示为矩形，称为参与者：提供输入数据或接收输出结果的实体。

**9.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）**

抛弃型原型：是为了对问题或者提议的解决方案有更多的了解而开发的软件，永远不会作为交付软件的一部分。

演化型原型：不仅帮助我们回答问题，而且还要演变为最终的产品。

10// 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**

**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？（223-227页）**

软件体系结构：用以解释如何将系统分解为单元以及这些单元又是如何相互关联，还描述这些单元的所有外部可见特征。

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，它提供相对较低层次的设计决策。

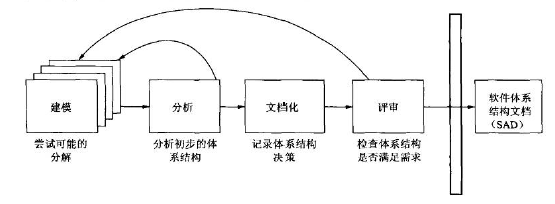
设计公约：一系列设计决策和建议的集合，采用这些设计决策和建议，能够提高系统某方面的设计质量。

设计：是一种创造性的过程，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/系统设计：确切地告诉客户系统要做什么。侧重于系统的功能，用顾客理解的语言描述系统功能，通过解释看得见的系统外部特征使顾客理解系统的功能。

技术设计：将概念设计转换为更为详细的文档。描述了系统采用的工艺，用计算机行话和技术术语描述，对技术规格说明的技术性描述。

**2.设计过程模型的几个阶段？**



**3.三种设计层次极其关系？（229页）**

体系结构：将需求规格说明中确定的系统能力与实现这些能力的系统构件关联起来。

代码设计：包含算法和数据结构，并且其构件是编程语言原语、数据操纵原语以及组装机制。

可执行设计：在比代码设计的层次还要低的静态层次处理代码设计，讨论内存分配、数据格式、位模式等。

关系：自顶向下设计：首先设计体系结构，然后进行代码设计，最后是可执行设计。

**4.什么是模块化？什么是抽象？（238页）**

模块化：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。

**5.论述设计用户界面应考虑的问题。（242页）**

关键要素：隐喻（可以认识和学习的基本术语、图像和概念）、头脑中的模型（数据、功能、任务和角色的组织和表示）、模型的导航规则（如何在数据、功能、活动和角色中移动）、外观（系统向用户传输信息的外观特性）、感觉（向用户提供有吸引力的体验的交互技术）。

文化问题

用户偏爱

**6.5.5节----模块独立性----耦合与内聚的概念及各个层次划分？（248----xxx页）**

耦合度：两个部件之间的相互关联的程度。

分为：非直接耦合（模块之间没有信息传递）

数据耦合（模块之间传递的是数据）

特征耦合（模块之间传递的是数据结构）

控制耦合（模块之间传递的是控制量）

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）。

内聚度：构成构件内部的粘合程度。

分为：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中）

时间性内聚（部件各个部分要求在同一时间完成）

过程性内聚（各部分由特定的次序）

通讯性内聚（各部分访问共享数据）

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）。

**7.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）**

**Chapter06**

**1.//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？**

面相对象：是一种软件开发方法，他将问题和问题的解决方案组织为离散对象的集合，数据结构和行为都包含在对象的表示中。

7个基本特征及实现：标识（将数据组织为离散的、可区别的实体，称为对象。单个对象具有相关联的状态和行为。）、抽象（OO系统中的抽象有助于表示正在开发的系统中的不同观点。合起来，这些抽象形成一个层次，以说明不同的系统观点是如何彼此相关的。）、分类（将具有共同属性和行为的对象分组。）、封装（类封装对象的行为和属性，隐藏实现细节。）、继承（一开始先广义的定义一个类，然后将他们细化为更具体的子类。某个子类可以继承父类的结构、行为以及属性。）、多态（同样的行为在不同点类或子类中会有不同的表现。）和持久性（对象的名称、状态和行为超越时间或空间的能力，换句话说，当转换对象时，保存对象的名称、状态和行为。）。

**2.使用高级语言的OO基本编程方法和技巧。（286-291）**

**3.什么是设计模式？**

模式：是抽象体系结构元素的模版，可以用来指导生成设计。

设计模式：是一套被反复使用、多数人知晓的、经过分类编目的、代码设计经验的总结。使用设计模式是为了可重用代码、让代码更容易被他人理解、保证代码可靠性。

**4.OO设计的基本原则？**

开闭原则、里氏代换原则、依赖倒转原则、接口隔离原则、笛米特法则、合成/聚合复用原则。

**5.OO开发有何优势？（291页）**

语言的一致性：可以使用同样的术语描述问题及其解决方案：类、对象、方法、属性和行为。

过程的一致性：保持术语和视角的一致性，所有过程使用同样的术语，便于测试需求。使用封装的数据和行为，组成独立的单元。

**6.OO开发过程有几个步骤？（292页）**

需求、设计、编码和测试。（需求分析、高层设计、低层设计、面相对象的程序设计、测试）

**7.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）**

用例：通过建模用户、外部系统或其他外部实体与要开发的系统之间的对话，描述系统应该具有或展现的特定功能。

要素：参与者（与系统交互的实体，可以是某个用户、某个设备或其他系统）、用例（对系统某些方面的功能性描述，对参与者可见，参与者通过这些用例反映自己对系统的观点）、扩充（扩展用例，以说明不同的或更深层次的视角）和使用（实际上对已定义用例的复用）。

**8.用例图、类图等对面向对象的项目开发的意义是什么？**

对功能的完整描述；便于用户、设计者、测试者之间的交流；是系统分析中更多正是建模的基础。

**9.熟悉类图中各个类之间的基本关系分类（303-305）**

泛化：在一个继承关系中，超类泛化子类。

关联：两个类一起出现，并且它们之间的关系必须保持一段时间。

聚合：一个类是另外一个类的一部分。松散的部分与整体的关系。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（此时不会影响整体对象的定义，

组装：强的部分与整体的关系。组合关系中整体对象可以控制成员对象的生命周期，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。

依赖：一个项的定义发生改变，则会引起另外一个项的改变变化。

**10.熟悉类图等的组成和画法（300-308页）**

**11.知道UML其他图示结构的基本用途。**

**Chapter07**

**1.//为什么说编码工作是纷繁复杂甚至令人气馁？（337页）**

第一，设计人员可能没有处理平台和编程环境的所有特性。易于用图表描述的结构和关系并不是总能够直截了当地编写成代码。第二，我们必须一这样一种方式编写代码：不仅要在再次使用代码进行测试的时候便于自己理解，而且当系统随着时间演化时，也便与他人理解。第三，在创建易于复用的代码的同时，还必须利用这些特性：设计的组织结构、数据结构、程序设计语言的概念。

**2.//一般性的编程原则应该从哪三个方面考虑？（340-344页）**

控制结构：将设计转换成代码时，我们希望能够保留构件的控制结构。在隐含调用和面相对象设计中控制是基于系统状态和变量发生的变化的。程序的结构能够反映出设计的控制结构。模块化编程，强调可读性。

算法：程序设计通常会指定一类算法，用于编写某些构件。注意性能效率所花费的代价，在性能与其他质量因素之间进行平衡。

数据结构：编写程序时，应该安排数据的格式并存储数据，使得数据管理和操纵更为直观。

**3.//论述编码阶段实现某种算法时所涉及的问题。（342页）**

使代码更快运行可能会伴随一些隐藏的代价，不要牺牲代码的清晰度和正确性，必须了解所使用的编译器是如何优化代码的。

**4.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）**

其他程序注释、有意义的变量名和语句标记、安排格式以增强理解、文档化数据。

**5.什么是极限编程(XP)? 以及派对编程？(357页)**

极限编程（XP）：是一种轻量级的软件开发方法论，属于敏捷开发方法。XP从实践中来，是对实践的总结，也是经过实践检验的，其主要特征是要适应环境变化和需求变化，充分发挥开发人员的主动精神。XP承诺降低软件项目风险，改善业务变化的反应能力，提高开发期间的生产力，为软件开发过程增加乐趣等等。两类参与者：客户：定义将要实现的系统之特征，描述测试计划，分配系统实现和测试的优先级；程序员：将客户的上述诉求予以编程实现。

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**

**1.// 产生软件缺陷的原因？（365页）**

规格说明可能是错误的，或者遗漏了某个需求；读于指定的硬件和软件，规格说明中可能包含不可能实现的需求；系统设计中可能包含故障；程序设计中可能包含故障；程序代码可能是错误的。

**2.//将软件缺陷进行分类的理由？（367页）**

在编码完程序构件之后，我们通常要对代码进行检查，以找出故障并立刻去除它们。当不存在明显的故障时，我们就测试程序，创造一些条件，使代码不能像设计的那样做出反应，看一看能否发现更多的故障。

**3.几种主要的缺陷类型？（367-368页）**

算法故障：由于处理步骤中的某些错误，使得对于给定的输入，构件的算法或逻辑没有产生适当的输出。

计算故障/精度故障：一个公式的实现是错误的，或者计算结果没有达到要求的精度。

文档故障：文档与程序实际做的事情不一致。

压力故障/过载故障：填充数据结构时超过了它们规定的能力。

能力故障/边界故障：系统活动到达指定的极限时，系统性能会变得不可接受。

计时故障/协调故障：在开发实时系统时，一个关键的考虑因素是几个同时执行的或按仔细定义的顺序执行的进程之间的协调问题，当协调这些事件的代码不适当时，会出现计时故障或协调故障。

吞吐量故障/性能故障：系统不能以需求规定的速度执行。

恢复故障：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。

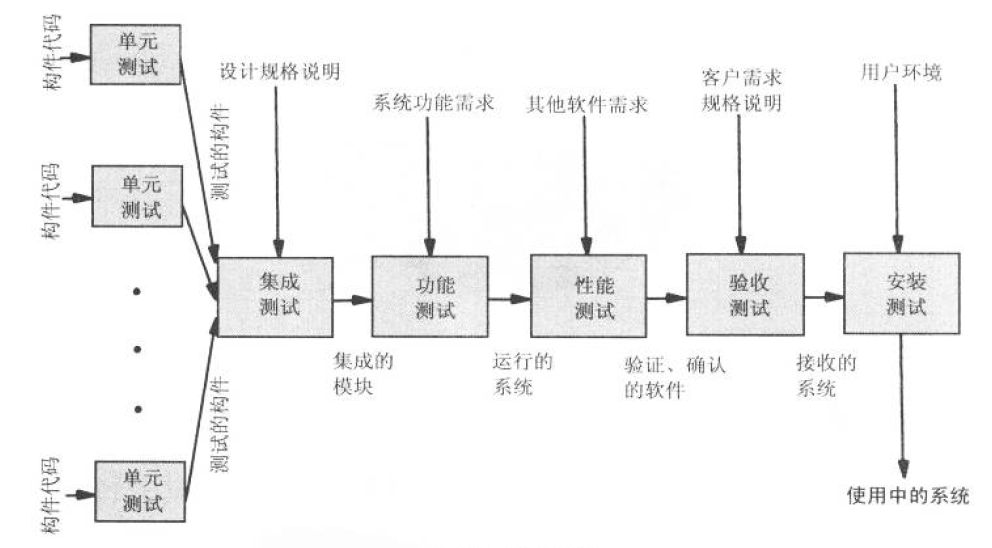
硬件和系统软件故障：提供的硬件和系统软件实际上并没有按照文档中的操作条件和步骤运作。

标准和过程故障：代码没有遵循组织机构的标准和过程。

**4.//什么是正交缺陷分类？（369页）**

一种故障跟踪方法，分类方案必须是产品和组织无关的，关键特征之一就是正交性（被分类的任何一项只属于一个类别）

**5.测试的各个阶段及其任务？（372页图8.3）**



模块测试/构件测试/单元测试：将每个程序构件与系统中其他构件隔离，对其本身进行测试。

集成测试：验证系统构件是否能够按照系统和设计规格说明中描述的那样共同工作的过程。

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。

性能测试：将系统与软件和硬件需求的剩余部分进行比较。

验收测试：根据客户的需求描述对系统进行检查。

安装测试：确保系统将按照它应该的方式来运行。

**6.// 测试的态度问题？（为什么要独立设置测试团队？）（373页）**

新程序员不习惯将测试看作是一个发现的过程。用户要确保系统在所有条件下都适当运行。忘我编程的态度：把程序看作一个更大系统的构件，而不是那些编写他们的人的财产。

**7.测试的方法----黑盒、白盒的概念？（374）**

黑盒测试：从外部观察测试对象，将其看作是一个不了解其内容的闭盒/黑盒，测试就是向闭盒提供输入数据并记录产生的输出。免于受强加给测试对象内部结构和逻辑的约束，不可能完备测试。

白盒测试：将测试对象看成一个透明盒，根据测试对象的结构用不同的方式来进行测试，可以设计执行构建内所有语句或所有控制路径的测试用例，以确定测试对象是否是适当运作的。

**8.什么是单元测试？ 什么是走查和检查？（376页）**

单元测试：通过通读程序对代码进行检查，试着找出算法、数据以及愈发中的故障。甚至可以将代码与规格说明进行比较，与设计进行比较，以确保已经考虑了所有相关情况。接着，编译代码，排除任何剩余的语法故障。最后，开发测试用例，以证明是否将输入适当的转换成了所期望的输出。

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，非正式的，注意力集中在代码上，发现故障但不必修改它们。

检查：评审小组按照一个事先准备好的关注问题清单来检查代码和文档。正式的。

**9.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）**

黑盒测试分类：等价分类法、边界值分析法、错误猜测法、因果图法。

白盒测试分类：逻辑覆盖法、路径覆盖法。

**10.黑盒白盒方法的分类，各种覆盖方法等。（课件和补充课件）**

**11.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**

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

集成测试：将单个构件组合成一个运转的系统进行测试。

分类：自底向上集成（通过合并构件来测试较大型系统。每一个处于系统层次中最低层的构件首先被单独测试，接着要测试的是那些调用了前面已测构件的构件。反复采用此方法，知道所有的构件都被测试完毕。）、自顶向下集成（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。）

构件驱动程序：调用特定构件并向其传递测试用例的程序。

桩：一种专用程序，用于模拟（测试时）缺少构件时的活动。

**13.//传统测试和OO测试有何不同？OO测试有何困难？（398-399页）**

区别：传统测试当系统改变时，需要新老测试用例（回归测试）；OO测试必须对重载的子类进行测试，可能会使用不同的测试用例，单元测试比较简单，但是集成测试比较广泛。

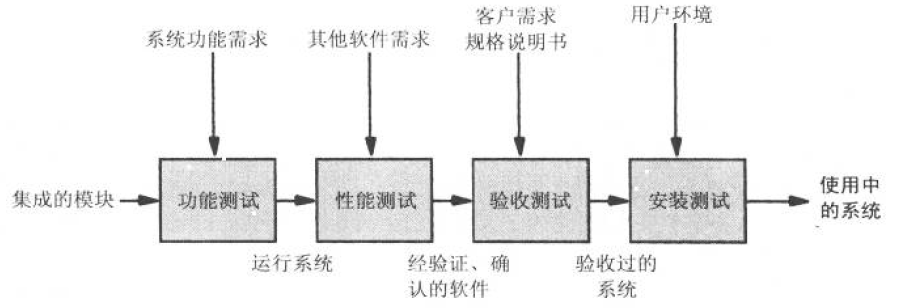
困难：没有什么工具支持表述为对象和方法的需求的验证；帮助测试用例生成的大多数工具并不能处理用对象和方法表述的模型；大多数源代码测度的定义是针对过程代码而不是针对对象和方法；代码覆盖测度和工具在面相对象测试中的作用要比在传统测试中的作用小。

**14.// 测试计划涉及的几个步骤？（400页）**

制定测试目标、设计测试用例、编写测试用例、测试测试用例、执行测试、评估测试结果。

**Chapter09**

**1.系统测试的主要步骤及各自含义？（420页, 图9.2）**



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。

性能测试：将集成的构件与非功能系统需求进行比较。

验收测试：向客户保证构件的系统就是客户想要的系统。

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

**2.//什么是系统配置？软件配置管理？基线？（423页）（或见课件）**

系统配置：向特定客户交付的一组系统构件

软件配置管理：开发和测试不同的配置需要配置管理。配置管理控制不同系统配置之间的差别，将风险和错误减少到最低程度。

基线：是指软件文档和其他资料的集合，它们代表了产品在某一时间点的情况(以及其他参考点)。

**3.什么是回归测试？（425页）**

用于识别在改正当前故障的同时可能引入的新故障，用于新的版本或发布的一种测试，以验证与旧版本或发布相比，它是否仍然以同样的方式执行相同的功能。

**4.功能测试的含义极其作用？（430页）**

含义：对需求规格说明书中的功能描述进行测试。

作用：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。

**5.功能测试的基本指导原则？（431）**

具有很高的故障检测概率；使用独立于测试人员和程序员的测试小组；了解期望的动作和输出；既要测试合法输入，也要测试不合法输入；永远不要为了使测试更加容易而去修改系统；制定停止测试标准。

**6.性能测试的含义与作用？（436页）**

含义：测试非功能需求。

作用：测试功能执行的方式，根据客户规定的性能目标来测量，这些性能目标表示非功能需求。

**7.性能测试的主要分类？（436页）**

压力测试：当系统在短时间内达到压力极限时，对系统进行的测试。

容量测试：强调的是处理系统中的大量数据。

配置测试：分析需求中指定的各种软件和硬件配置。

兼容性测试：检查接口功能是否按照需求执行。

回归测试：保证新系统的表现至少与老系统一样好。

安全性测试：确保安全性需求得到满足。

计时测试：评估涉及对用户的响应时间和一个功能的执行时间的相关需求。

环境测试：考察系统在安装场所的执行能力。

质量测试：评估系统的可靠性、可维护性和可用性。

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。

维护测试：为了帮助人们发现问题的根源提供诊断工具和过程的需要。

文档测试：确保已经编写了必需的文档。

人为因素测试：检查设计系统用户界面的需求。

**8.// 什么是可靠性、可用性和可维护性？（438页）**

可靠性：指一个系统对于给定的时间间隔内、在给定的条件下无失效运作的概率。

可用性：指在给定的时间点上，一个系统能够按照规格说明正确运作的概率。

可维护性：在给定的使用条件下，在规定的时间间隔内，使用规定的过程和资源完成维护活动的概率。

**9.确认测试, 确认测试分类？（基准测试和引导测试）（447-448页）**

确认测试：在确信软件开发的初始阶段系统满足了所有指定的需求后让用户对系统进行的测试。

分类：

基准测试：顾客准备测试用例，这些测试用例代表了系统实际安装投入运行的典型条件，顾客针对每一个测试用例评价系统的性能。

引导测试：在试验的基础上安装该系统，它依赖日常的工作来测试所有的功能。

**10.什么是alpha测试？β测试？（448页）**

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。

β测试：客户的试验。

**11.什么是安装测试？（450页）**

在用户的环境下测试，解决由于开发环境和用户环境的不同而引起的错误。集中于两件事：安装系统的完备性；验证任何可能受场所条件影响的功能和非功能特新。

**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。**