**Chpt.1 软件工程概述**

SE的定义、目的、方法及作用**（**P2 / P16）

|  |
| --- |
| 定义：软件工程是一种系统工程，不止包括对技术问题的分析与解决，还包括对开发过程和给参与者分配合适的角色等方面的管理  目的：生产出高质量的软件进而找到解决方案，并考虑那些对质量有影响的特性  方法及作用：  分析（analysis）---分析问题，调查软件正反两方面，  设计（design）---给出解决方案，  发展团队（developing team）---描述在团队中的人员的角色和职责，  发展（develop）---实现解决方案（实现对象、活动、封装等等），  项目管理（project management）---将系统分为小部分，逐步明确过程，控制进度，处理每个改变等等 |

//开发模式（paradiam）（P4）

|  |
| --- |
| 它表示开发软件时特定的方法或哲学。 |

说明错误、缺陷、失效的含义与联系。（请举例说明）（6页）（44页习题3）

|  |
| --- |
| **错误[**error]，是进行软件开发过程中人为出错造成的  例如，设计人员可能误解了某个需求，创建出与需求分析人员和用户的实际意图不相符的设计。这个设计故障是一种错误的编码，可能导致其他故障，如不正确的代码或用户手册中不正确的描述等。  **故障/缺陷**[fault]：当人们在进行软件开发活动的过程中出现错误时，就会引起缺陷。（静态存在）  **失效**[failure]是指系统违背了它应有的行为（由于**故障**产生）。（动态存在）  例如，需求文档可能会包含故障，所以即使系统按照需求规格来运行，如果它未进行应有的行为，也称为失效。    **联系**：单个错误可能产生多个故障。故障是系统的内部视图，这是从开发人员的角度看待系统；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（不执行故障代码就不会是代码失效）。 |

软件质量应从哪几个方面来衡量？论述之。

|  |
| --- |
| **产品质量**     特性的重要性取决于分析这个软件的人，如果软件用易于学习或是易于使用的方式做了用户想做的事情，用户就断定软件是高质量的。     软件还必须由那些设计和编写代码的人员以及维护该程序的人员来评价，这些时间人员倾向于考虑产品的内部特性，有时甚至会在产品交付给用户之前就考虑这些内部特性。  **过程质量**  有很多活动会影响到最终的产品质量。只要活动出了差错，产品的质量就会受到影响。因此，许多软件工程师认为开发和维护过程的质量与产品的质量是同等重要的。  **商业价值**     在商业环境中，质量是根据软件所处的商业环境提供的产品和服务来看待的。也就是说，我们考虑的是产品的技术价值，而不是更广泛的商业价值。 |

//软件系统的系统组成（P16）

|  |
| --- |
| 1.     活动和对象  2.     关系和系统边界  **A system = entities(实体) + activities(活动) + relationships(关系) + boundary(边界)** |

现代软件工程大致包含的几个阶段及各个阶段文档

|  |
| --- |
| 1.     需求分析和定义───需求规格说明  2.     系统设计────设计描述  3.     程序设计─┐  4.     程序实现─┴─程序文档  5.     单元**[测试](http://lib.csdn.net/base/softwaretest" \t "_blank" \o "软件测试知识库)**─┐  6.     集成测试─┼─测试数据  7.     系统测试─┘  8.     系统交付─┬─培训手册  9.      维护    ─┘ |

// 使现代SE实践发生变化的（七个）关键因素是什么？（28--29页）

|  |
| --- |
| 商业软件的投放市场时间的紧迫性  计算经济学的改变  强力的桌面计算平台的出现  局域网和广域网的延伸  面向对象技术的出现和采用  使用窗口、图标、菜单和指针的图形用户界面  瀑布模型用于软件开发的不可预测性 |

什么是抽象？（30页）

|  |
| --- |
| 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。 |

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：软件开发活动中的各种组织及规范方法。  重要性：具有通用性（一致性、结构性）和指导性。  阶段：上有 |

什么是复用？（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。 |

**Chpt.2 过程和生命周期的建模**

**软件过程、软件过程模型、过程建模工具和技术**

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转到另外一个阶段。一个开发阶段必须在另一个开发阶段之前完成。 |
| 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作出解释。  ·是其他复杂模型的基础 |
| 缺点：·瀑布模型最大的问题是它并不能反映实际的代码开发方式。       ·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       ·文档转换有困难 |

原型的概念（P51）

|  |
| --- |
| 原型（prototype）是一个部分开发的产品，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）

|  |
| --- |
| definition:  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。  分类及特点：  **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。 |

螺旋模型四个象限的任务及四重循环的含义？ （P58）

|  |
| --- |
| 四象限：·确定目标、可选方案及约束；         ·评估可选方案及风险         ·计划         ·开发与测试  **操作概念**是第一次迭代的产品，而**需求**则是第二次迭代的主要产品，第三次迭代系统开发**产生设计**，第四次迭代能够**进行测试**。 |

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？

针对本章描述的每一种过程模型，讨论该模型是如何处理开发后期重要的需求变化的？

|  |
| --- |
| **瀑布模型**  V模型  **原型化模型**  可操作规格模型  分阶段开发模型  螺旋模型  瀑布模型从一种非常高层的角度描述了软件开发过程中进行的活动，并且提出了要求开发人员经过的事件序列。该模型适用于项目开始时需求已确定的情况。V模型是瀑布模型的变种，它说明测试活动是如何与分析和设计相联系的。原型模型允许开发人员快速地构造整个系统或系统的一部分以理解或澄清问题。原型的用途是获知用户的真正需求，因此原型模型可以有效地引发系统需求。螺旋模型把开发活动和风险管理结合起来，以将风险减到最小并控制风险。 |

//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？

|  |
| --- |
| 1.     设计对于分析模型应该是可跟踪的：软件的模块可能被映射到多个需求上。  2.     设计结构应当尽可能的模拟实际问题。  3.     设计应当表现出一致性。  4.     不要把设计当成编写代码。  5.     在创建设计时就应该能够评估质量。  6.      评审设计以减少语义性的错误。 |

什么是UP， RUP?

|  |
| --- |
| 统一过程（RUP/UP，Rational Unified Process）是一种以用例驱动、以体系结构为核心、迭代及增量的软件过程模型，由UML方法和工具支持，广泛应用于各类面向对象项目。  统一过程是一个面向对象且基于网络的程序开发方法论。 |

**Chapter03 计划和管理项目**

**项目进展追踪、项目人事组织、**

**工作量估算、风险管理**

什么是项目调度？活动？里程碑？（83页）

|  |
| --- |
| **项目调度**：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。  **项目进度**是对特定项目的软件开发周期的刻画。  **活动**：是项目的一部分，它在一段时间内发生。  **里程碑**：是活动的完成——某一特定的时刻。 |

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**是一条每个节点的时差都为零的路径。最长路径就是一条关键路径。  时差=可用时间-真实时间  时差=最晚开始时间-最早开始时间 |

软件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、开发环境、技术的经验，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 |

软件项目（团队）组织的基本结构？（101页）

|  |
| --- |
| -主程序员  -副主程序员  -高级程序员（-初级程序员）、资料员、管理机构、测试小组 |

// 专家估**[算法](http://lib.csdn.net/base/datastructure" \t "_blank" \o "算法与数据结构知识库)**的大致含义？（106页），算式估算法的大致含义？（108页）

|  |
| --- |
| **专家估算**：请几位专家做出3种预测，来形式化地表示类推过程：一个悲观的预测（x）、一个乐观的预测（y），和最可能的预测（z），通过公式（x+4y+z）/6计算这些数的beta概率分布的平均值。通过使用这种技术，产生的估算是对个人估算的“规范化”。 |
| **算式估算（这个不用看）**：研究人员已经创建出表示工作量和影响工作量的因素之间关系的模型。这些模型通常用方程式描述。大部分模型认为项目规模是方程式中影响最大的因素，表示工作量的方程是：E = (a + bSc) m(X)。其中S是估算的系统规模，而a、b、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。 |

试述COCOMO模型的三个阶段基本工作原理或含义。（111页）

|  |
| --- |
| 阶段1（应用组装）：根据高层的工作量生成器来获取项目的规模。  阶段2（早期设计）：使用功能点对规模进行测量。  阶段3（后体系结构）：根据功能点或代码行来进行规模预算。 |

什么是软件风险？有几种降低风险的策略？（119、122页）

|  |
| --- |
| 风险：人们不希望看到的、有负面结果的事件。包括与该事件有关的损失，事件发生的可能性，我们能改变结果的程度。  风险估算：定义、分析、优先级  风险控制：降低、管理计划、化解  风险杠杆：（降低前的风险成本-降低后的风险成本）／降低风险所需的成本  ·通过改变性能或功能需求来降低风险  ·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弥补经济上的损失  ·假设风险会发生，接受并用项目资源控制风险 |

**Chapter04 获取需求**

**需求的过程、引出、种类、特征**

**建模符号描述系统／建模原语（面向对象／UML）**

**需求的表达方式、需求和特定语言**

**原型化需求、需求文档化、需求有效化**

**量化需求**

需求的含义是什么? (143页)

|  |
| --- |
| 需求，就是对期望的行为的表达，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、问题分析、规格说明草稿、需求核准、生成正式的SRS |

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） |

 //如何使需求变得可测试？（151-152页, sidebar4.4）

|  |
| --- |
| A:  针对需求确定一种量化的描述方法，避免模糊的表达方式  B:  将各种指代用词替换为实体的正式名称  C:  每个名词或款项在需求文档中给出唯一定义。 |

需求文档分为哪两类？（153页）

|  |
| --- |
| **需求定义**，完整罗列用户想要系统实现的功能。它面向的是业务相关的人员，例如：委托人，客户以及用户；  **需求规格说明SRS**，用工程术语重述需要实现的功能，它面向的是技术性人员，例如：设计人员、测试人员以及项目经理。 |

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）

|  |
| --- |
| **功能需求**根据要求的活动（如对输入的反应，活动发生时每一个实体之前和之后的状态）来描述需要的行为。  **非功能性需求/质量需求**，描述一些软件解决方案必须拥有的质量特性，如快速的响应时间、易使用性、高可靠性或低维护代价。 |
| **设计约束**是已经作出的设计决策或限制问题解决方案集的设计决策，例如平台或构建接口的选择。  **过程约束**是对于构建系统的技术和资源的限制。例如，客户可能坚持使用[**敏捷**](http://lib.csdn.net/base/agile)方法，以便在继续增加新特征的时候能够使用早期版本。 |

需求的特性？（正确性、一致性、完整性）（155页）

|  |
| --- |
| **·正确**：我们和客户都应该评审需求文档，确保它们符合我们对需求的理解  **·一致**：一般来讲，如果不可能同时满足两个需求，那么这两个需求就是不一致的。  **·无二义**：如果需求的多个读者能够一致、有效地解释需求，那么需求就是无二义性的。  **·完备**：如果需求指定了所有约束下的、所有状态下的、所有可能的输入的输出以及必需的行为，那么这组需求就是完备的。  **·可行**：当用户要求两个或更多的质量需求时，常常会出现可行性问题  **·相关**：有时，某个需求会不必要地限制开发人员，或者会包含与客户需要没有直接关系的功能。  **·可测试**：如果需求能够提示验收测试（明确证明最终系统是否满足需求），需求就是可测试的。  **·可跟踪**：对需求进行精心组织并唯一标记，已达到易于引用的目的；求定义中的每一条都在需求规格说明中有对应，反之亦然。 |

了解DFD图的构成及画法（172页）

|  |
| --- |
| DFD：Data Flow Diagrams 数据流图，面熟数据流进入系统、相互转化、离开系统。重点是信息流而非控制流。  圆形：加工动作。箭头：数据流向。  双横线之间：数据集合。框：外部项。 |

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**：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。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**： 不仅帮助我们回答问题，而且还要演变为最终的产品。用于了解问题，并作为将来准备提交的系统的一部分 |

**Chpt.5 设计体系结构**

**设计过程、系统设计中的分解与模式化**

**软件结构风格和设计策略**

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）

|  |
| --- |
| 体系结构(Architecture):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(Design pattern):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（低于体系结构）。  设计公约(Design Convention):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。  概念设计：告诉客户系统可以做什么（软件结构和功能）。  技术设计：告诉程序员系统会如何做（软件功能和界面接口方法）。包括主要硬件组成及其功能、软件功能和层次、数据结构和数据流。 |

三种设计层次及其关系？（229页）

|  |
| --- |
| **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，相当于系统设计。  **代码设计**:各个部件／模块的算法、数据结构的设计  **运行设计**:最底层设计，内存分配、数据格式、位模式等等 |
| **关系**：流程工作中，先是体系结构设计，然后是代码设计，最后是运行设计；  随着设计人员对解决方案及其含义有更多的理解，他们就会往返于各层次之间。  （程序设计由代码设计和运行设计组成） |

论述设计用户界面应考虑的问题。（242页）

|  |
| --- |
| ·设计界面要注意解决的要素（寓意/比喻、思维模型、领航规则、外观、感觉）；  ·文化差异问题；  ·用户爱好问题 |

什么是模块化？什么是抽象？（238页）

|  |
| --- |
| **模块化**，也称作关注点分离，是一种把系统中各不相关的部分进行分离的原则，以便于各部分能够独立研究。  对细节的隐藏称为**抽象**。 |

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**：两个软件部件之间的相互关联程度  **内聚**：软件部件内部的关联程度  层次划分：如下 |

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）

|  |
| --- |
| 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 |
| 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  通讯性内聚：各部分访问共享数据  过程性内聚：各部分有特定次序  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 |

**Chpt.6 设计模块**

//什么是面向对象？（286页）

|  |
| --- |
| 面向对象是一种软件开发方法，它将问题及其解决方法组织成一系列独立的对象，数据结构和动作都被包括在内 |

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）

|  |
| --- |
| 基本特征：一致性、抽象、分类、封装、继承、多态、持久性 |

//什么是设计模式？

|  |
| --- |
| **设计模式**编写了设计决策以及最好的实践，它们根据设计原则来解决一些特定的问题；  它们不是拿来就可以使用的打包的解决方案，而是解决方案的模板，必须针对特定的状况进行修改和调整。 |

//OO设计的基本原则？

|  |
| --- |
| 模块化、接口、信息隐藏、增量式开发、抽象、通用性 |

OO开发有何优势？（291页）

|  |
| --- |
| 语言具有一致性（在同一时期同时描述问题和解决方案）；  过程具有一致性（从需求到测试，所有的过程采用相同的语义构造）。 |

OO开发过程有几个步骤？（292页）

|  |
| --- |
| OO需求、OO高级设计、OO低级设计、OOP、OO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）

用例图、类图等对面向对象的项目开发的意义是什么？

|  |
| --- |
| 这些表示法每种都显现了系统的某个方面，因此相应地，这种表达也提供了对于问题或解决方案的详细描述。 |

熟悉类图中各个类之间的基本关系分类（303-305）

熟悉类图等的组成和画法（300-308页）

了解UML其他图的基本用途。

**Chpt.7 编写程序**

//为什么说编码工作是纷繁复杂甚至令人气馁？（337页）

|  |
| --- |
| 第一，设计对于编码来说并不总是简单明了的；  第二，编码应该是可懂的  第三，需要考虑重用 |

一般性的编程原则应该从哪三个方面考虑？（340-344页）

|  |
| --- |
| 编程标准对自身的用处  编程标准对他人的用处  设计与编程实现相匹配 |

//论述编码阶段实现某种算法时所涉及的问题。（342页）

|  |
| --- |
| 编写更快代码的代价。可能会使代码更加复杂，从而花费更多的时间编写代码。  测试代码的时间代价。代码的复杂度要求有更多的测试用例或测试数据。  用户理解代码的时间代价。  需要修改代码时，修改代码的时间代价。 |

在编写程序内部文档时，除了HCB头部注释版款外，还应添加什么注释信息？（352-354页）

|  |
| --- |
| 1.其他程序注释、2.有意义的变量名和语句标记、3.安排格式以增强理解、4.文档化数据 |

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 。 |
| **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。 |

**Chpt.8 测试程序**

// 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；  ·对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。 |

//将软件缺陷进行分类的理由？（367页）

|  |
| --- |
| 当不存在明显的故障时，我们就测试程序，通过创造一些条件，使代码不能像计划的那样做出反应，看一看能否发现更多的故障。因此，知道我们正在查找什么类型的故障是很重要的。 |

几种主要的缺陷类型？（367-368页）

|  |
| --- |
| A.     算法故障  B.     计算故障和精度故障  C.     文档故障  D.    压力故障或过载故障  E.     能力故障或边界故障  F.     计时故障  G.    性能故障  H.    恢复故障  I.      硬盘和系统软件故障  J.       标准和过程路障 |

 什么是正交缺陷分类？（369页）

|  |
| --- |
| 其中，故障被分为不同的类别，这些类别共同勾画出开发过程的哪些部分需要关注，因为它们是产生很多故障的原因。因此，分类方案必须是产品无关的和组织无关的，并且可适用于开发的所有阶段的。 |

// 测试的态度问题？（为什么要独立设置测试团队？）（373页）

|  |
| --- |
| 即使用忘我方法开发一个系统，有时也难以从测试过程中排除个人感情。因此，通常使用一个独立的测试小组来测试系统，这样，避免了故障的个人责任与尽可能多地发现故障的需要之间的冲突。 |

测试的各个阶段及其任务？（372页图8.3）

|  |
| --- |
|  |
|  |

测试的方法----黑盒、白盒的概念？（374）

|  |
| --- |
| **黑盒**：从外部观察测试对象，将其看做是一个不了解其内容的闭盒或黑盒，那么，我们的测试就是向闭盒提供输入数据，并记录产生的输出。 |
| **白盒**：将测试对象看成是一个开盒（透明盒）或白盒，然后可以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？什么是走查和审查？（376页）

|  |
| --- |
| 单元测试：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 |
| 在**走查**中，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。  在**审查**中，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（更正式）。 |

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）

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

集成测试及其主要方法的分类？（390-392）（驱动，桩的概念）

|  |
| --- |
| （1）      自低向上集成（驱动程序）  优点：易生成测试用例；适合于面向对象开发系统；底层为通用模块比较好  缺点：顶层设计缺陷为主要的缺陷，不能及时地发现改正  （2）      自顶向下集成（桩）  优点：顶层模块缺陷尽早发现  缺点：产生测试用例比较难，需要大量的桩 |

传统测试和OO测试有何不同？OO测试有何困难？（398-399页）

|  |
| --- |
| 面向对象系统的测试中更容易和更困难的部分： |

// 测试计划涉及的几个步骤？（400页）

|  |
| --- |
| （1）制定测试目标  （2）设计测试用例  （3）编写测试用例  （4）测试测试用例  （5）执行测试  （6）评估测试结果 |

（**了解**）

**Chpt.9 测试系统**

系统测试的主要步骤及各自含义？（420页, 图9.2）

|  |
| --- |
| 1.     功能测试：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：软件与非功能系统需求进行比较  3.     验收测试：根据用户的需求描述检查系统  4.      安装测试：保证系统按照它应有的方式进行 |

//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**是向特定客户交付的一组系统组件。  **配置管理**控制不同系统配置之间的差别，将风险和错误降低到最低程度。  某个特定系统的一个配置有时称为一个**版本**  **基线**是软件文档或源码的一个稳定版本,它是进一步开发的基础（这基线我百度的） |

// 什么是回归测试？（425页）

|  |
| --- |
| **回归测试**是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，它是否以同样的方式执行相同的功能。 |

功能测试的含义及其作用？（430页）

|  |
| --- |
| **功能测试**基于系统功能性需求，属于闭盒测试。我们不必知道正在执行哪个构件，相反，必须知道系统应该做什么。  作用：将系统的实际表现与其需求进行比较。 |

功能测试的基本指导原则？（431）

|  |
| --- |
| ·具有很高的故障检测概率  ·使用独立于设计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 |

性能测试的含义与作用？（436页）

|  |
| --- |
| 在我们确定系统能执行需求所要求的功能之后，就要考虑这些功能执行的方式。  因此，功能测试针对的是功能需求，而性能测试针对的是非功能需求。 |

性能测试的主要分类？（436页）

|  |
| --- |
| 压力测试、容量测试、配置测试、兼容性测试、安全性测试、计时测试、环境测试、质量测试、恢复测试、维护测试、文档测试、人为因素测试 |

// 什么是可靠性、可用性和可维护性？（438页）

|  |
| --- |
| **可靠性**是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**是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**是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。 |

确认测试, 确认测试分类？（基准测试和引导测试）（447-448页）

|  |
| --- |
| 当功能测试和性能测试完成后，我们确信，系统满足在软件开发的初始阶段过程中指定的所有需求，下一步要询问客户和用户，他们是否持有同样的观点。 |
| 在基准测试中，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。 |
| 在并行测试中，新系统与先前版本并行运转，用户逐渐地适应新系统，但是继续使用与新系统功能相同的老系统。 |

什么是alpha测试？β测试？（448页）

|  |
| --- |
| 在客户进行实际的试验性测试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，这样的内部测试称为**α测试**。 |
| **β测试：**客户的测试，属于试用性质的非正式的验收测试。 |

什么是安装测试？（450页）

|  |
| --- |
| 即在用户的场所安装系统。  如果验收测试已经是在实地进行，就不一定需要安装测试了。 |