第一章 概论

1、什么是软件工程？它的各种定义方式

2、软件生存周期的几个阶段及任务（ISO/IEC）

3、各类软件过程模型的特点及使用场景

4、CASE软件的概念和分类

5、什么是CMMI，英文含义，分几个过程

第二章 系统工程

1、可行性分析的概念（...可行性3个）

第三章 需求工程

1、需求工程的3个阶段

第四章 设计工程

1、模块 模块化设计的概念（4.4）

2、各种类型的模块耦合度和内聚度

4、什么是模块独立性，如何度量？

第五章 结构化设计与分析

1、DFD图具体应用：数据字典，判定树，判定表（用况分析）

2、SD（结构化设计）什么是启发式设计，设计策略？

3、如何判断数据流图的一致性和完整性

第七章 面向对象方法基础（应用，选择，15+）大于二十分

1、OO的基本概念 面向对象的概念

2、用况图，用况之间的关系

3、类图，类之间的关系，类中的属性和方法

4、状态机图，顺序图，结构图的基本概念

第一章 概论

1、什么是软件工程？它的各种定义方式

- NATO会议上给出的定义：

软件工程是建立和使用一套合理的工程原则，以便获得经济的软件，这种软件是可靠的，可以在实际的机器上高效运行

- IEEE在软件工程术语汇编中的定义：

软件工程是：

①将系统化的、严格约束的、可量化的方法应用于软件的开发、运行和维护，即将工程化应用于软件

②对在①中所述的方法的研究

- 《计算机科学技术百科全书》中的定义：

软件工程是应用计算机科学理论和技术以及工程管理原则和方法，按预算和进度实现满足用户要求的软件产品的工程，或以此为研究对象的学科

2、软件生存周期的几个阶段及任务（ISO/IEC）

1、计算机系统工程：

- 确定待开发软件的总体要求和范围，以及该软件与其他计算机系统元素之间的关系
- 进行成本估算，做出进度安排
- 进行可行性分析，即从经济、技术、法律等方面分析待开发的软件是否有可行的解决方案，并若干个可行的解决方案中租出选择

2、需求分析：

- 主要解决待开发软件需要“做什么”的问题，确定软件的功能、性能、数据、界面等要求，生成软件需求规约（也称软件需求规格说明）

3、设计：

- 主要解决待开发软件“怎么做”的问题，软件设计通常分为系统设计（也称概要设计或总体设计）和详细设计。
- 系统设计的主要任务是设计软件系统的体系结构，包括软件系统的组成成分、各成分的功能和接口、成分间的连接和通信，同时设计全局数据结构。
- 详细设计的任务是设计各个组成成分的实现细节，包括局部数据结构和算法。

4、编码：

- 用某种程序设计语言，将设计的结果转换成可执行的程序代码。

5、测试：

- 发现并纠正软件中的错误和缺陷，主要包括单元测试，集成测试，确认测试和系统测试。

6、运行和维护：

- 在软件运行期间对投入运行的软件进行维护，即发现了软件中潜藏的错误或需要增加的功能或使软件适应外界环境的变化等情况出现时，对软件进行修改。

3、各类软件过程模型的特点及使用场景

软件过程模型习惯上也称为软件开发模型，是软件开发全部过程和任务的结构框架

1、瀑布模型，上一阶段的活动完成并经过评审后才能开始下一阶段的活动

- 特征：

接收上一阶段活动的结果作为本阶段活动的输入

依据上一阶段活动的结果实施本阶段应完成的任务

对本阶段的活动进行评审

将本阶段的结果作为输出，传递给下一阶段

- 适用场景：

2、演化模型

- 特征：

在获取了一组基本的需求之后，通过快速分析，构造出该软件的一个初始可运行版本，通常称之原型，然后，根据用户在使用原型的过程中提出的意见和建议，或者增加新的需求，对原型进行改造，获得原型的新版本，重复这一过程，最终得到令客户满意的软件产品

- 适用场景：

对软件需求缺乏明确认知的情况，典型的演化模型有增量模型、原型模型、螺旋模型

3、增量模型：

- 特征：

融合了瀑布模型的基本成分（重复地应用）和演化模型的迭代特征，强调每一次增量都发布一个可运行的产品。

- 适用场景：

特别适用于需求经常发生变化的软件开发。

市场急需而开发人员和资金不能在设定的市场期限之前实现一个完善的产品的软件开发。

4、原型模型：

- 特征：

优点：

1）开发人员和用户在“原型”上达成一致。这样一来，可以减少设计中的错误和开发中的风险，也减少了对用户培训的时间，而提高了系统的实用、正确性以及用户的满意程度。

2）缩短了开发周期，加快了工程进度。

3）降低成本。

缺点：

1）当告诉用户，还必须重新生产该产品时，用户是很难接受的。这往往给工程继续开展带来不利因素。

2）开发者为了使一个原型快速运行起来，往往在实现过程中采用这种手段。

3） 不宜利用原型系统作为最终产品。采用原型模型开发系统，用户和开发者必须达成一致：原型被建造仅仅是用户用来定义需求，之后便部分或全部抛弃，最终的软件是要充分考虑了质量和可维护性等方面之后才被开发。

- 适用场景：

用户不能完全准确地表达对未来系统的全面要求，开发者对要解决的应用问题模糊不清，以至于形成的需求规格说明常常是不完整的、不明确的、有时甚至是歧义的，此外，整个开发过程中用户可能会产生新的要求，导致需求的变更（原型模型适用于那些不能预先确切定义需求的软件系统的开发，更适用于那些项目组成员（包括分析员、设计员、程序员和用户）不能很好的交流或者通信的情况下。）

5、螺旋模型：

- 特征：

强调风险分析

- 适用场景：

螺旋只适合于大规模软件项目。

6、喷泉模型

- 特征：

迭代和无间隙特性。迭代是指各开发活动常常重复工作多次，相关的功能在每次迭代中随之加入演进的系统。无间隙是指开发活动之间不存在明显的边界。

- 适用场景：

适应于面向对象的软件开发过程。

7、基于构件的开发模型

- 特征：

i. 支持软件复用。

ii. 利用预先包装好的软件构件来构造应用系统。

- 适用场景：

8、形式化方法模型

- 特征：

用严格的数学语言和语义描述功能规约和设计规约，通过数学的分析和推导，易于发现需求的岐义性、不完整性和不一致性，易于对分析模型、设计模型和程序进行验证。

- 适用场景：

4、CASE软件的概念和分类

CASE（computer aided software engineering），计算机辅助软件工程

概念：

计算机辅助软件工程是指使用计算机及相关的软件工具辅助软件开发、维护、管理等过程中各项活动的实施，以确保这些活动能高效率，高质量地进行。

分类：

- 支持软件开发过程的工具

需求分析工具、设计工具、编码工具、排错工具、测试工具等

- 支持软件维护过程的工具

版本控制工具、文档分析工具、开发信息库工具、逆向工程工具、再工程工具等

- 支持软件管理过程和支持过程的工具

项目管理工具、配置管理工具、软件评价工具等

5、什么是CMMI，英文含义，分几个过程

CMM（Capability Maturity Model）即能力成熟度模型，是用于评价软件机构的软件过程能力成熟度的模型。

CMM提供了一个成熟度等级框架： 1级-初始级、 2级-可重复级、 3级-已定义级、 4级-已管理级和5级-优化级。

1.初始（initial）级：

软件过程的特点是无秩序的，甚至是混乱的。几乎没有什么过程是经过妥善定义的，成功往往依赖于个人或小组的努力。

2.可重复（repeatable）级：

建立了基本的项目管理过程来跟踪成本、进度和功能特性。制定了必要的过程纪律，能重复早先类似应用项目取得的成功。

3.已定义（defined）级：

已将管理和工程活动两方面的软件过程文档化、标准化，并综合成该机构的标准软件过程。所有项目均使用经批准、剪裁的标准软件过程来开发和维护软件。

4.已管理（managed）级：

收集对软件过程和产品质量的详细度量值，对软件过程和产品都有定量的理解和控制。

5.优化（optimizing）级：

整个组织关注软件过程改进的持续性、预见及增强自身，防止缺陷及问题的发生。过程的量化反馈和先进的新思想、新技术促使过程不断改进。

第二章 系统工程

1、可行性分析的概念（...可行性3个）

- 经济可行性

主要进行成本效益分析，从经济角度，确定系统是否值得开发。

- 技术可行性

主要根据系统的功能、性能、约束条件等，分析在现有资源和技术条件下系统能否实现。通常包括风险分析、资源分析和技术分析。

- 法律可行性

主要研究系统开发过程中可能涉及到的合同、侵权、责任以及各种与法律相抵触的问题。

第三章 需求工程

1、需求工程的6个阶段

- 需求获取
  - 系统分析人员通过与用户的交流、对现有系统的观察及对任务进行分析，确定系统或产品范围的限制性描述、与系统或产品有关的人员及特征列表、系统的技术环境的描述、系统功能的列表及应用于每个需求的领域限制、一组描述不同运行条件下系统或产品使用状况的应用场景以及为更好地定义需求而开发的任意原型。
  - 需求获取的工作产品为进行需求分析提供了基础。
- 需求分析和谈判
  - 需求获取结束后，分析活动对需求进行分类组织，分析每个需求其它需求的关系来，检查需求的一致性、重叠和遗漏的情况，并根据用户的需要对需求进行排序。
  - 在需求获取阶段，经常出现以下问题：
    - 用户提出的要求超出软件系统可以实现的范围或实现能力；
    - 不同的用户提出了相互冲突的需求
- 系统建模
  - 建模工具的使用在用户和系统分析人员之间建立了统一的语言和理解的桥梁，同时系统分析人员借助建模技术对获取的需求信息

进行分析，排除错误和弥补不足，确保需求文档正确反映用户的真实意图。

- 常用的分析和建模方法有面向数据流方法、面向数据结构方法和面向对象的方法。

- 需求规约
  - 软件需求规约是分析任务的最终产物，通过建立完整的信息描述、详细的功能和行为描述、性能需求和设计约束的说明、合适的验收标准，给出对目标软件的各种需求。
  - 需求规约作为用户和开发者之间的一个协议，在之后的软件工程各个阶段发挥重要作用。
- 需求验证
  - 作为需求开发阶段工作的复查手段，需求验证对功能的正确性、完整性和清晰性，以及其它需求给予评价。为保证软件需求定义的质量，评审应以专门指定的人员负责，并按规程严格进行。
- 需求管理
  - 需求工程包括获取、分析、规定、验证和管理软件需求，而“软件需求管理”则是对所有相关活动的规划和控制。
  - 换句话说，需求管理就是：一种获取、组织并记录系统需求的系统化方案，以及一个使用户与项目团队对不断变更的系统需求达成并保持一致的过程。

第四章 设计工程

1、模块 模块化设计的概念（4.4）

模块：模块是数据说明、可执行语句等程序对象的集合，它是单独命名的，并且可以通过名字来访问。

模块化：模块化即把软件按照规定原则，划分为一个个较小的、相互独立的但又相互关联的部件，实际上是系统分解和抽象的过程。

模块化设计：模块化设计是指在对一定范围内的不同功能或相同功能不同性能、不同规格的产品进行功能分析的基础上，划分并设计出一系列功能模块，通过模块的选择和组合可以构成不同的产品，以满足市场的不同需求的设计方法。

2、各种类型的模块耦合度和内聚度

课本P51。

3、软件设计的任务，设计原则

软件设计的任务：

软件设计的输入是软件分析模型。使用一种设计方法，软件分析模型中通过数据，功能和行为模型所展示的软件需求信息被传送给设计阶段，产生数据/类设计、体系结构设计、接口设计、部件级设计

设计原则：

- 抽象与逐步求精
- 模块化
- 信息隐藏

4、什么是模块独立性，如何度量?

模块的功能独立性，也就是说在设计程序模块时，使得模块实现独立的功能并且与其他模块的接口简单，符合信息隐蔽原则，模块间关联和依赖程度尽可能小。

独立性可以由两项指标来衡量：内聚度和耦合度。内聚度衡量同一个模块内部的各个元素彼此结合的紧密程度，耦合度衡量不同模块彼此之间的相互依赖的紧密程度。

第五章 结构化设计与分析

1、DFD图及其具体应用，数据字典，判定树，判定表（用况分析）

2、SD（结构化设计），什么是启发式设计策略?

结构化设计是将结构化分析得到的数据流图映射成软件体系结构的一种设计方法

启发式设计策略：

a. 改造程序结构图，降低耦合度，提高内聚度

b. 避免高扇出，并随着深度的增加，力求高扇入

i. 一种较好的结构图形态应该是“椭圆”型，即顶层模块的删除较大，中间层模块的扇出较小，而底层模块具有高扇入

c. 模块的影响范围应限制在该模块的控制范围内

i. 模块的影响范围是指受该模块中决策影响的所有其他模块

ii. 模块的控制范围是指模块自身以及它可以直接或间接调用的所有模块

d. 降低模块接口的复杂程度和冗余程度，提高一致性

i. –模块接口上应尽可能传递简单数据，而且传递的数据应保持与模块的功能相一致，即不传递与模块

功能无关的数据

e. 模块的功能应是可预测的，避免对模块施加过多的限制

i. –模块功能可预测是指该模块对相同的输入能产生相同的输出

ii. –限制一个模块只处理单一的功能，那么，这个模块体现出高内聚

f. 尽可能设计单入口和单出口的模块

i． 单入口和单出口的模块能有效地避免内容耦合

3、如何判断数据流图的一致性和完整性

分层数据流的一致性：

1. 父图与子图平衡：父图与子图平衡是指任何一张DFD子图边界上的输入输出数据流必须与其父图中对应加工的输入输出数据流保持一致。

2. 数据守恒：①一个加工所有输入数据流中的数据，必须能从该加工的输入数据流中之间获得，或者能从该加工的处理而产生。②多余数据流：加工未使用其输入数据流中的某些数据项

3. 局部文件：讨论分层数据流图中一个文件应画在哪些DFD中，而不该画在哪些DFD中。

4. 一个加工的输出数据流不能与该加工的输入数据流同名

分层数据流流图的完整性:

1. 每个加工至少有一个输入数据流和一个输出数据流

2. 在整套分层数据流图中，每个文件至少有一个加工读该文件，有另一个加工写该文件

3. 分层数据流图中的每个数据流和文件都必须命名（除了流入或流出文件的数据流），并保持与数据字典一致

4. 分层DFD中的每个基本加工（即不再分解成子图的加工）都应有一个加工规约（也称加工小说明）

第七章 面向对象方法基础（应用，选择，15+）大于二十分

1、OO的基本概念 面向对象的概念

面向对象方法是一种把面向对象的思想应用于软件开发过程中，指导开发活动的系统方法，是建立在对象概念（对象，类和继承）基础上的方法，简称OO方法

- 对象：对象是指一组属性及这组属性上的专用操作的封装体。属性通常是一些数据，有时也可以是另一个对象，每个对象都有自己的属性值，表示该对象的

状态，对象中的属性只能通过该对象所提供的操作来存取或修改，操作也称为方法或服务，操作规定了对象的行为，表示对象所能提供的服务。

- 封装：封装（encapsulation）是一种信息隐蔽技术，用户只能看见对象封装界面上的信息，对象的内部实现对用户是隐蔽的。封装的目的是使对象的使用者和生产者分离，使对象的定义和实现分开。
- 类： 类是一组具有相同属性和相同操作的对象的集合。
- 继承： 继承是类间的基本关系，它是基于层次关系的不同类共享数据和操作的一种机制。父类中定义了其所有子类的公共属性和操作，在子类中除了定义自己特有的属性和操作外，可以继承其父类（或祖先类）的属性和操作，还可以对父类（或祖先类）中的操作重新定义其实现方法。
- 消息：消息传递是对象间通信的手段，一个对象通过向另一个对象发送消息来请求其服务。一个消息通常包括接收对象名、调用的操作名和适当的参数（如果有必要的话）。消息只告诉接收对象需要完成什么操作，但并不指示接收者怎样完成操作。消息完全由接收者解释，接收者独立决定采用什么方法完成所需的操作。
- 多态性：多态性是指同一个操作作用于不同的对象上可以有不同的解释，并产生不同的执行结果。例如“画”操作，作用在“矩形”对象上，则在屏幕上画一个矩形，作用在“圆”对象上，则在屏幕上画一个圆。也就是说，相同操作的消息发送给不同的对象时，每个对象将根据自己所属类中定义的这个操作去执行，从而产生不同的结果。
- 动态绑定: 动态绑定是指在程序运行时才将消息所请求的操作与实现该操作的方法连接起来。

传统的程序设计语言的过程调用与目标代码的连接（即调用哪个过程）放在程序运行前（即编译时）进行（称为静态绑定），而动态绑定则是把这种连接推迟到运行时才进行。

2、用况图，用况之间的关系

3、类图，类之间的关系，类中的属性和方法

4、状态机图，顺序图，活动图的基本概念

第十章 敏捷软件开发

1、敏捷开发的定义，特点，价值观及原则

敏捷开发以用户的需求进化为核心，采用迭代、循序渐进的方法进行软件开发。在敏捷开发中，软件项目在构建初期被切分成多个子项目，各个子项目的成果都经过测试，具备可视、可集成和可运行使用的特征。换言之，就是把一个大项目分为多个相互联系，但也可独立运行的小项目，并分别完成，在此过程中软件一直处于可使用状态。

- 特点：

（1）敏捷开发的过程有着更强的适应性而不是预设性，从敏捷宣言的第四条响应变化高于预设计划便可以看出来。因为软件开发过程的本身的不可预见性，很多用户在项目开始时不可能对于这个项目有着一个完整而明确的预期。很多对软件的预期都在后期的修改和完善过程中产生。因此高适应性显然更加符合软件工程开发的实际。而敏捷开发实现其适应性的方式主要在于，第一，缩短把项目提交给用户的周期；第二，增加用户，业务人员，开发人员这三者之间的交流；第三，通过减少重构的成本以增加软件的适应性。

（2）敏捷开发的过程中，更加的注重人的因素。在传统软件工程中，个人的因素很少的被考虑到分工中，每个个体都是只是整个代码开发机器的一个小小的螺丝钉，个人的意志和创造力很大程度上的被抹去为了更好的为集体服务。而在敏捷开发过程中，每个个人的潜力被充分的考虑，应用什么技术很大程度上直接由在第一线开发的技术人员决定；每个人的特点和创造力都可以充分地发挥，这样开发出来的软件更加的具有生命力，因为他融入了开发者的心血和创意，开发者不再是进行机械的乏味的堆砌，而是创造属于自己的艺术品，这样的条件下产生的代码必然在质量上更占优势。

（3）在敏捷开发的过程中，整个项目是测试驱动的而不是文档驱动的。不仅每个模块有着自己的相应的测试单元，开发人员在开发自己的模块的过程中必须保证自己所开发的模块可以通过这一单元的测试，并且集成测试贯穿了整个开发过程的始终。集成测试每天会进行十几次甚至几十次，而不是像传统方法一样只有当各个模块的编码都结束了之后再进行联合调试。这样，在软件开发的进程中每一点改动所引起的问题都容嘉容易暴露出来，使得更加容易在错误刚刚产生的时候发现问题从而解决问题。这样就避免了在最后整个系统完成时错误隐藏的太深给调试造成极大的困难。

- 价值观：
    - 个体和交互 胜过 过程和工具
    - 可以工作的软件 胜过 面面俱到的文档
    - 客户合作 胜过 合同谈判
    - 响应变化 胜过 遵循计划
- 原则：

1. 我们的最高优先级是持续不断地、及早地交付有价值的软件来使客户满意。

2. 拥抱变化，即使是在项目开发后期，敏捷过程愿意为了客户的竞争优势而接纳变化

3. 经常地交付可工作地软件，相隔几星期或一两个月，倾向于采用较短的周期

4. 业务人员和开发人员必须在整个阶段紧密合作

5. 围绕着被激励的个体构建项目，为个体提供所需的环境和支持，给予新人，从而达成目标

6. 在团队内和团队间沟通信息的最有效和最搞笑的方式是面对面的交流

7. 可工作的软件是进度的首要度量标准

8. 敏捷过程倡导可持续开发。项目发起者、开发人员和用户应该维持一个可维持的步调

9. 持续地追求技术卓越和良好设计，可以提高敏捷性

10. 以简洁为本，它是减少不必要工作的艺术

11. 最好的架构、需求和设计是从自组织的组队中涌现出来的

12. 团队定期地反思如何变得高效，并且相应地调整自身地行为

2、XP方法及特点

- 极限编程是一种敏捷软件开发方法
- 特点：

与传统的开发过程不同，极限编程的核心活动体现在需求→测试→编码→设计过程中，因此对工作环境、需求分析、设计、编程、测试、发布等提出了新的思路和要求。

1、工作环境：XP要求每个参加项目开发的人都担任一个角色（项目经理、项目监督人等），并履行相应的权利和义务。所有的人都在一个开放式的开发环境中工作，最好是在同一个大房间中工作，随时讨论问题，强调每周40小时工作制，不加班。

2、需求分析：客户被纳入开发队伍。由于客户不具备计算机专业知识，无法用专业语言明确描述需求，所以开发人员和客户一起，用讲故事的方式把需求表达出来，这种故事被称为user story，即用user story表示需求。开发人员根据经验将许多user story 组合起来，或将其进行分解，最终记录在story card的小卡片上，这些user story将陆续被程序员在各个小的周期内，按照商业价值、开发风险的优先顺序逐个开发。

3、设计：XP强调简单设计（simple design），即用最简单的办法实现每个小需求。在XP中，没有那种传统开发模式中一次性的、针对所有需求的总体设计，这些设计只要能够满足系统客户在当前的需求就可以了，不需要考虑将来可能的变化，整个设计过程包括在整个螺旋式发展的项目中。

4、编程：成对编程（pair programming）是极限编程的一大特色，即两个人一起使用同一个屏幕，同一个键盘，共同完成一段程序的编码。成对编程的好处是，可以提高纪律性，更容易写出优质的代码，同时保证编程的流畅

进行，更重要的是，能够使得整个团队更方便地分享编程经验，有利于新手的快速成长。

5、测试：在极限编程中，测试是非常重要的一个环节，它首先要求在开始写程序之前先写好测试，其目的是为了提高软件的可测试性。XP要求开发人员经常把开发好的模块整合到一起，每次整合后都要运行单元测试；做任何的代码复核和修改，都要运行单元测试；发现了漏洞，就要增加相应的测试。除了单元测试之外，还要进行整合测试、功能测试、负荷测试和系统测试等。所有这些测试是极限编程开发过程中最重要的文档之一，也是最终交付给用户的内容之一。

6、发布：XP要求按照开发计划，每经过一个开发周期，软件就发布一次，而不是像传统的开发方法那样，整个软件开发完成后才发布。在一个开发周期内，开发人员要求客户选择最有价值的user story作为未来一两个星期的开发内容，一个开发周期完成后，提交给客户的系统虽然不是最终的产品，但它已经实现了几个客户认为是最重要的story，开发人员将逐步在其基础上增加新的模块，而且在发布前软件都经过单元测试和集成测试，因此，虽然软件并不完备，但是，发布的软件客户还是可以真正使用的。

第十三章 软件测试（应用，基础概念）

1、测试的目的

软件测试的目的是发现软件中的错误和缺陷，并加以纠正。应该排除对测试的错误观点，设计合适的测试用例，用尽可能少的测试用例，来发现尽可能多的软件错误。

2、什么是白盒测试/黑盒测试

- 白盒测试又称结构测试，这种方法把测试对象看作一个透明的盒子，测试人员根据程序内部的逻辑结构及有关信息设计测试用例，检查程序中所有逻辑路径是否都按预定的要求正确地工作。
- 黑盒测试又称行为测试，这种方法把程序看作一个黑盒子，测试人员完全不考虑程序内部的逻辑结构和内部特性，只根据程序的需求规约说明书，检查程序的功能是否符合它的功能需求。

3、什么是单元、集成、系统、确认测试（基本概念）

- 单元测试：单元测试又称模块测试，着重对软件设计的最小单元——软件构件或模块进行测试，单元测试根据设计描述，对重要的控制路径进行测试，以发现构件或模块内部的错误。单元测试通常采用白盒测试，并且多个构件或模块可以并行进行测试。

- 集成测试；集成测试又称组装测试，经单元测试后的模块通常需集成为软件系统，集成测试是对集成后的软件系统进行测试，主要用来揭露设计阶段产生的错误
- 确认测试：经集成测试后的软件需经过确认测试才能交付使用，确认测试通常采用黑盒测试方法。
- 系统测试：将软件与计算机系统的其他元素集成起来，检测它是否符合系统工程中对软件的要求，能否与计算机系统的其他元素协调地工作，系统测试就是对整个基于计算机的系统进行的一系列测试。

4、软件测试分几个阶段，每个阶段与软件生命周期的什么文档相对应

5、什么是α，β，回归，压力，性能测试

α测试：α测试是用户在开发环境下的测试，或者是开发内部的用户在模拟实际环境下的测试；

β测试：β测试是由软件的一个或多个用户在实际使用环境下进行的测试；

回归测试：回归测试是对已经进行过的测试的测试用例子集的重新测试，以确保对程序的改变和修改，没有非传播非故意的副作用

压力测试：压力测试又称调度测试，是在一种需要非正常数量、频率或容量的方式下执行系统，其目的是检查系统对非正常情况的承受程度

性能测试：性能测试用来测试软件在集成的系统中的运行性能

6、白盒：逻辑覆盖，基本路径测试的测试用例（注意步骤，10+）

逻辑覆盖：

基本路径测试：是一种白盒测试技术，这种方法首先根据程序或设计图画出控制流图，并计算其区域数，然后确定一组独立的程序执行路径（称为基本路径），最后为每一条基本路径设计一个测试用例

7、黑盒：等价划分类测试（注意步骤，10+）

第十五章 软件维护与再工程

1、软件维护的概念

软件维护是指软件系统交付使用以后，为了改正错误或满足新的需要而修改软件的过程

2、可维护性的概念

可维护性是指理解、改正、调整和改进软件的难易程度

3、影响软件维护的因素

- 可理解性
- 可测试性
- 可修改性
- 可移植性

4、维护的分类

根据起因不同，软件维护可以分为：

- 纠错性维护
- 适用性维护
- 改善型维护
- 预防性维护

5、再工程的概念

再工程是指在逆向工程所获信息的基础上修改或重构已有的系统，产生系统的一个新版本。

6、逆向工程的概念

逆向工程，是指在软件生存周期中，将软件的某种形式描述转换成更抽象形式的活动。

7、再工程的过程，几个步骤的名字

- 库存目录分析
- 文档重构
- 逆向工程
- 代码重构
- 数据重构
- 正向工程