chapter01:

1.软件工程的定义、目的、方法及作用：

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、过程和原则。是一个系统工程，既有对技术问题的分析与解决，也有对开发过程和参与者的管理。

目的：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。软件工程师使用工具、技术过程和范型来提高软件产品的质量，他们的目标是使用高效的、高效率的方法形成相关问题的有效解决方案。

方法：

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

方法和技术：是产生某些结果的形式化过程。

工具：是用更好的方式完成某件事情的设备或自动化系统

过程：把工具和技术结合起来，共同生产特定产品。

范型：表示构造软件的特定方法与哲学。

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。

2.开发模式：

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：

错误：错误是在软件生产过程中人为产生的错误（需求说明中的错误、代码中的错误等）

缺陷：缺陷是在功能实现中产生的问题（一个错误可能产生几个静态缺陷）

失败：失败是相对于系统指定行为的偏离（动态存在的）

缺陷是系统内部的观点，就像从开发者的角度看待问题那样；失败是外部观点：即用户所看到的问题，并非每个缺陷都对应相应的失败。

4.软件质量应该从哪几个方面来衡量？论述之。

产品的质量：

用户从外部特性看，软件具有足够的功能并且易于学习和使用就说明软件有高的质量；

开发者从内部特性看，错误数量和类型来评判软件质量的高低。

过程质量：很多过程都能影响最终产品的质量，如果任何一个活动除了差错，整个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同等重要。

商业应用背景下的软件质量：在商业环境中，质量是根据软件所出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，而我们只根据最终产品的技术质量来作出决定。

5.软件系统的系统组成（系统的要素有哪些）：

对象（实体）+ 活动 + 关系 + 系统边界

活动：活动是发生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，活动通过改变属性把一个事物编程另一个事物。
对象：活动中涉及的元素称为对象。
关系：是指活动与对象之间的关系。
系统边界：即系统包含的功能与系统不包含的功能之间的界限。

6.现代软件工程大致包含的几个阶段以及各个阶段文档?

需求分析：主要包括问题定义、可行性研究、需求分析、复审等。（所有参与者）文档：《需求规格说明书》
系统设计：主要包括用户界面和软件结构图。文档：系统结构图
程序设计：主要包括模块功能算法与数据描述、复审等。（开发者）文档：UML 图
程序实现：主要包括编程和复审（设计者和编码人员）文档：源代码和注释
单元测试：模块测试与性能测试，复审（测试团队）文档：测试报告
集成测试：按照结构图进行测试产生测试报告，复审（测试团队）
系统测试：按照 SRS 对系统总体功能进行测试，复审（开发者和用户）
系统交付：交付产品，复审（使用者和操作人员）文档：用户培训
维护：修改软件过程，为满足改错或新需求，复审（维护团队）。文档：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?

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?

抽象是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。
通常，抽象可以认为是标识对象的类，使得我们能够把多个项组合在一起。

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?

软件过程：软件开发活动中的各种组织及规范方法。可以将一组有序的任务称为过程，它涉及活动、约束和资源使用的一系列步骤，用于产生某种想要的输出。
软件过程的重要性：
（1）它强制活动具有一致性和一定的结构，是程序的集合组合起来以产生满足目标和标准的产品
（2）过程结构允许我们分析、理解、控制盒改进组成过程的活动，并以此来指导我们的行动
（3）它能使我们获取经验并把它传授给他人。
软件过程的阶段：9 个（就是软件工程的 9 个阶段。）每一阶段本身就可以描述为一组活动的过程，并且每一个活动都包括约束、输出、和资源。每个活动涉及输入，子活动，约束，输出，和资源。
*过程阶段化的目的: 尽量通用化，以确保最终产品的高质量.

10.什么是重用等软件工程主要概念?

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。我们是指重复使用软件系统的任何部分，包括文档、代码、设计、需求、测试用例和测试数据等。
从复用者的角度来看有两种复用：生产者复用创建可复用的构建，而消费者复用是在以后的系统中使用它们。

chapter02:

1.什么是软件过程，软件过程的重要性是什么？ （P45-46）

软件过程：将软件开发中的一组有序的任务称为软件过程，它涉及活动、约束和资源使用的一系列步骤，用于产生某种想要的输出。

重要性：

（1）它强制活动具有一致性和一定的结构，是程序的集合组合起来以产生满足目标和标准的产品

（2）过程结构允许我们分析、理解、控制盒改进组成过程的活动，并以此来指导我们的行动

（3）它能使我们获取经验并把它传授给他人。

2.瀑布模型及各阶段文档，优缺点?

瀑布模型：瀑布模型将开发阶段描述为从一个开发阶段瀑布般地转换到另外一个阶段，一个开发阶段必须在另一个开发阶段开始之前完成。瀑布模从一种非常高层的角度描述了开发过程中进行的活动，并且提出了要求开发人员经过的时间序列。

阶段：需求分析 →系统设计→ 程序设计 →编码→ 单元测试和集成测试→ 系统测试 →验收测试 →运行和维护

优点：

（1）瀑布模型一直用来规范软件开发活动，每一个过程活动都有与其相关联的里程碑和可交付产品，以便于项目经理能够用模型判断在某一时刻项目里最后完成还有多远。

（2）它的简单性使得开发人员很容易向不熟悉软件开发用户作出解释。

（3）很多更复杂的模型实际上是在瀑布模型的基础上的润色，如加入反馈循环以及额外的活动。

缺点：

（1）它并不能反映实际的代码开发方式。除了一些理解非常充分的问题之外，实际上软件是通过大量的迭代进行开发的

（2）它没有揭示每一个活动如何把一种制品转化为另外一种制品

（3）没有吧软件看做一个问题求解的过程，而是从制造业的角度来看待软件开发的，软件开发应该是一个创造的过程，而不是制造的过程。

3.原型的概念

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?

含义：从编写需求文档到系统交付使用会经过若干年，称为循环周期，使用这种方法设计系统时使其能够一部分一部分的交付，从而在系统其余部分正在开发的同时，用户已经获得了一部分功能。 因此，通常会有两个系统并行运行，运行系统（产品系统）或开发系统。

基本分类：增量开发、迭代开发

增量开发：在增量开发中，需求文档中指定的系统按功能划分为子系统，定义发布时首先定义一个小的功能子系统，然后在每一个新的发布中增加新功能。

迭代开发：迭代开发是在一开始就提交一个完整的系统，然后在每一个新的发布中改变每个子系统的功能。

特点：

（1）即使还缺少某些功能，但在早期的发布中就可以开始培训。

（2）可以及早为那些以前从未提供的功能开拓市场。

（3）当运行系统出现未预料到的问题时，经常性的发布可以使开发人员能全面、快速修复这些问题

（4）针对不同的发布版本，开发团队将重点放在不同的专业领域技术上。

5.螺旋模型四个象限的任务及四重循环的含义?

定义：它以需求和一个初始的开发计划为起点，在产生操作概念文档之前，该过程进入一个评估风险以及可选原型的步骤。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

任务：

确定目标，可选方案及约束（2 左上）；

评估可选方案及风险（1 右上）；

计划（3 左下）；

开发与测试（4 右下）

四重循环的含义：

（1）操作概念是第一次迭代的产品；

（2）需求是第二次迭代的主要产品；

（3）第三次迭代产中，系统开发产生设计；

（4）第四次迭代能够进行测试。

（5）螺旋模型的每一次迭代都根据需求和约束进行风险分析，以权衡不同的选择，并且在确定某一特定选择之前，通过原型化验证可行性或期望度。当风险确认之后，项目经理必须决定如何消除或最小化风险。

6.习题 2：

	优点	缺点
瀑布模型	（1）瀑布模型一直用来规范软件开发活动，每一个过程活动都有与其相关联的里程碑和可交付产品，以便于项目经理能够用模型判断在某一时刻项目里最后完成还有多远。 （2）它的简单性使得开发人员很容易向不熟悉软件开发用户作出解释。 （3）很多更复杂的模型实际上是在瀑布模型的基础上的润色，如加入反馈循环以及额外的活动。	（1）它并不能反映实际的代码开发方式。除了一些理解非常充分的问题之外，实际上软件是通过大量的迭代进行开发的。 （2）它没有揭示每一个活动如何把一种制品转化为另外一种制品 （3）没有吧软件看做一个问题求解的过程，而是从制造业的角度来看待软件开发的，软件开发应该是一个创造的过程，而不是制造的过程。
V 模型	使得隐藏在瀑布模型中的迭代和重做活动更加明确。它通过开发和测试同时进行的方式来缩短开发周期，提高开发效率	V 模型仅仅把测试过程作为在需求分析、系统设计及编码之后的一个阶段，忽视了测试对需求分析,系统设计的验证，需求的满足情况一直到后期的验收测试才被验证。
原型化模型	可以得到比较良好的需求定义，容易适应需求的变化；有利于开发培训同步；需求表达清楚，利于确认各项系统服务的可用性；降低开始风险和开发成本。	不适用于开发大型的信息系统；系统难以维护；如果用户合作不好，盲目纠错，会拖延开发进度；准确的原型设计比较困难，且不利于开发人员的创新。
可转换模型	通过去除某些主要开发步骤来设法减少出错的机会，简化了开发过程	应用转换方法的主要障碍在于需要一个精确表述的形式化的规格说明，这样才可以基于它进行操作。
螺旋模型	1）设计上的灵活性,可以在项目的各个阶段进行变更。 2）以小的分段来构建大型系统,使成本计算变得简单容易。 3）客户始终参与每个阶段的开发,保证了项目不偏离正确方向以及项目的可控	很难让用户确信这种演化方法的结果是可以控制的。建设周期长，而软件技术发展比较快，所以经常出现软件开发完毕后，和当前的技术水平有了较大的差距，无法满足当前用户需求。 利用螺旋模型需要具有相当丰富的风险

	性。 4）随着项目推进,客户始终掌握项目的最新信息，从而他或她能够和管理层有效地交互。 5）客户认可这种公司内部的开发方式带来的良好的沟通和高质量的产品。	评估经验和专门知识，在风险较大的项目开发中，如果未能及时标识风险，势必会造成重大损失。

7.习题 3 讨论模型是如何处理开发后期的重要需求的?

瀑布模型：由于瀑布模型的不可逆性，若要重新进行需求调整，需要推翻或对 之前的流程设计进行大调整 ,重新按照流程再来一遍。

原型化模型：重新考虑和变更需求规格说明。

V 模型：重新对左边的步骤进行修正和改进

可转换模型：重新进行转换过程。

螺旋模型：再次进行风险分析并将风险降至最低。

8.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)

9.什么是 UP,RUP?

统一过程（UP）：可以用三句话来表达：它是用例驱动的、以基本架构（体系结构）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（OMG）的 UML 并与对象管理组织（OMG）的软件过程工程原模型（SPEM）等相兼容。 “统一过程”将重复一系列生命期，这些生命期构成了一个系统的寿命。每个生命期都以向客户推出一个产品版本而结束。每个周期包括四个阶段：开始阶段、确立阶段、构建阶段和移交阶段。每个阶段可以进一步划分为多次迭代。

统一开发过程 RUP（Rational Unified Process）：是一个面向对象且基于网络的程序开发方法论。RUP 是风险驱动的、基于 Use Case 技术的、以架构为中心的、迭代的、可配置的软件开发流程。我们可以针对 RUP 所规定出的流程，进行客户化定制，定制出适合自己组织的实用的软件流程。因此 RUP 是一个流程定义平台，是一个流程框架。

chapter03：

1.什么是项目进度？活动？里程碑?

活动：活动是项目的一部分，它在一段时间内发生。

里程碑：里程碑是活动的完成---某一特定时刻。

项目进度：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时间。进度是一个时间线。

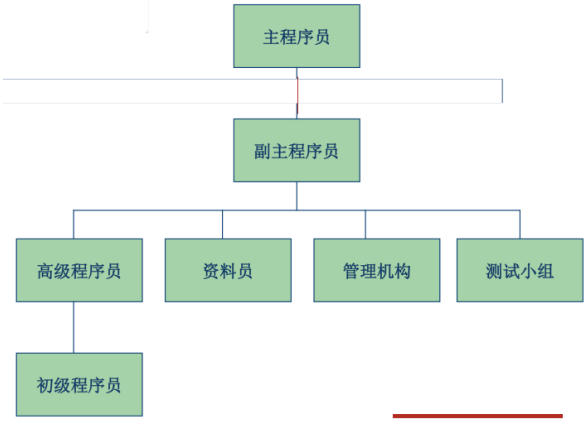
2.计算活动图的关键路径，习题 2，3，冗余时间，最早和最迟开始时间。（14 级考选择题之类的，一分左右）

3.软件团队人员应该具备的能力是什么?

（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（4）使用类似工具或语言的经验（5）使用类似开发环境的经验（6）使用类似技术的经验（7）培训（8）与其他人交流的能力（9）与其他人共同承担责任的能力（10）管理技能

4.软件项目组织的基本结构?

主程序员负责制组：有一个人总体负责系统的设计和开发，其他小组成员向该祝程序员汇报，主程序员对每一个决定由最终决策权。主程序员监督所有其他小组成员、设计所有程序、把代码分配给其他小组成员



忘我方法：让每个人平等地承担责任。民主式，小组投票产生决策。

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：很多工作量估算方法依赖于专家的判断。使用专家的知识和经验，对软件项目的工作量进行评估，预测的精确性基于估算者的能力、经验、客观性和洞察力。是对构建整个系统或其子系统所需的工作量做出经验性的猜测。
算式估算法：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，其中工作量是因变量，而其他因素是自变量。

6.试叙述 COCOMO 模型的三个阶段基本工作原理或含义。

从工程和经济两个方面考虑软件开发，将规模作为主要决定因素，用诸多成本驱动因子调整初始估算。模型充分反应了软件开发充分展开后的各个方面。
阶段一，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可能的规模知之甚少，因此 COCOMOⅡ用应用点来估算规模。
阶段二，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，但是远比第一阶段知道的信息要多。在阶段二，COCOMOⅡ使用功能点对规模进行测量。
阶段三，即后体系结构阶段，开发已经开始，而且已经知道了更多的信息。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。

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？

风险：是一种具有负面后果的、人们不希望发生的事情。
风险的特点（区别风险和其他项目事件）：
（1）与事件有关的损失：与风险有关的损失称为风险影响
（2）事件发生的可能性：对风险进行的测量称为风险概率
（3）更够改变结果的程度：能降低或消除风险所采取的行动称为风险控制
（4）风险成本（风险暴露）=风险影响*风险概率
主要风险管理活动 risk management：了解和控制项目风险的各种活动
风险评价：风险识别、风险分析、风险优先级分配
风险控制：风险降低、风险管理计划、风险化解
三种策略来降低风险：
（1）通过改变性能或功能需求，避免风险

（2）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弥补经济上的损失，从而转移风险。
（3）假设风险会发生，接受并用项目资源控制风险。

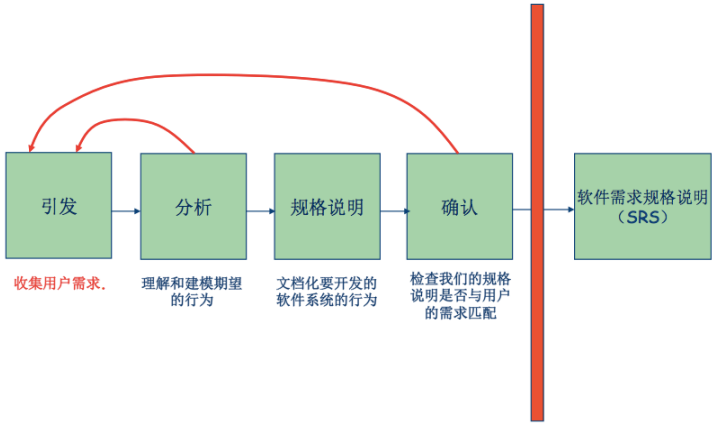
8.找出图 3.23 和 3.24 的关键路径。（习题 2.3）

chapter04：

1.需求的含义是什么?

需求：是对期望行为的表达。需求处理的是对象或实体，它们可能处于的状态，以及用于改变状态或对象特征的功能。
需求的目标：是理解客户的问题和需要，需求集中于客户和问题，而不是解决方案的实现。

2.需求作为一个工程，其确定需求的过程是什么?



3. 举例说明获取需求时，若有冲突发生，如何考虑根据优先级的进行需求分类及相互关系?

请求客户对需求进行优先级划分通常是有用的，这可以迫使客户思考提议的服务或特征中哪些是最重要的。一种大致的优先计划分方案可能将需求分为 3 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）
按照类型对需求进行优先级的分类，能够帮助所有相关人员理解自己到底需要什么。当软件开发项目受到时间或资源的限制时，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，考虑是去掉还是延期。还可解决与质量需求之间的矛盾。

4.如何使需求变得可测试？ （151-152 页, sidebar4.4）

5.需求文档分为哪两类?

（1）需求定义：是客户想要的每一件事情的完整列表
（2）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?

功能需求：根据要求的活动来描述需求的行为。（描述系统内部功能或系统与外部环境的交互作用，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）

非功能需求（质量需求）：描述一些软件解决方案必须拥有的质量特征，如快速的响应时间、易使用性、高可靠性或低维护代价等。

设计约束：是已经做出的设计决策或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

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

过程约束：是对用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

7.需求的特性？（正确性、一致性、完整性）P109

（1）正确性: 需求规格说明对系统功能、行为、性能等的描述必须与用户的期望相吻合，代表了用户的真正需求。

（2）一致性: 需求规格说明对各种需求的描述不能存在矛盾，如术语使用冲突、功能和行为特性方面的矛盾以及时序上的不一致等。

（3）无二义性（确定性）

（4）完备性: 需求规格说明应该包括软件要完成的全部任务，不能遗漏任何必要的需求信息，注重用户的任务而不是系统的功能将有助于你避免不完整性。

（5）可行性（6）相关性（7）可测试性（8）可跟踪性

8.了解 DFD 图的构成及画法。P121 数据流图

9.在需求原型化方面，什么是抛弃型原型？什么事演化型原型？

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，而且还要演变成最终产品

这两种技术有时都称为快速原型化，因为它们都是为了回答需求的问题而构建软件。

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？

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

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。(此设计决策低于体系结构) (注: 此处为说明,不是定义)

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。

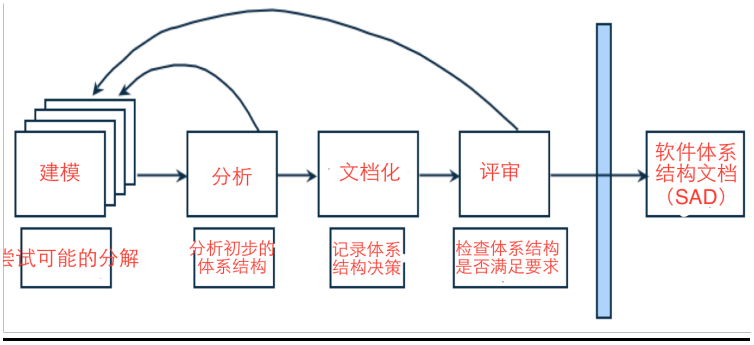
设计：是一种将问题转换为解决方案的创造性过程，他考虑如何实现所有的客户需求。设计所产生的计划也叫设计。

概念设计：确切地告诉客户系统要做什么

技术设计：一旦客户认可概念设计，系统构建人员就将概念设计转换为更为详细的文档，即技术设计，技术设计确切的告诉开发人员系统将如何运转。

* 概念设计强调的是系统功能，而技术设计描述的是系统将要采取的方式。

2.软件设计过程模型的几个阶段？



3.三种设计层次及其关系?

4.什么是模块化? 什么是抽象?

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。
抽象：抽象是一种忽略一些细节来关注其他细节的模型或表示，是基于某种归纳水平的问题描述。

5.论述设计用户界面应该考虑的问题。?

（1）应处理以下几个关键要素：
1.隐喻：可识别和学习的基本术语，图片和概念
2.头脑中的模型：数据、功能、任务和角色的构成和表现
3.模型的导航没规则：怎样在数据、功能、活动和角色中转移
4.外观：系统向用户传输信息的外观特征
5.感觉：向用户提供有吸引力的体验的交互技术
（2）文化问题：需要考虑使用系统的那些用户的信仰、价值观、道德规范、传统、风俗和传说。两种解决方法：1.排除特定的文化参考或偏见，让界面变得尽可能“国际化”2.采用无偏见设计并使之时应使用软件的文化
（3）用户偏爱：可以为不同用户涉及多个界面。

6. 5.5 节 模块独立性—耦合与内聚的概念及各个层次划分? 各个层次划分?

耦合：指构件之间的相互依赖性，从高到低可分为
（1）内容耦合：一个构件直接修改了另外一个构件（当一个构件修改了另外一个构件的内部数据项时，或一个构件内的分支转移到另外一个构件中的时候，就可能出现内容耦合）
（2）公共耦合：不同构件访问公共数据。例如，一个公共变量可以被不同的构件修改。
（3）控制耦合：某个构件通过传递参数或返回代码来控制另外一个构件的活动，模块间传递的是控制量。
（4）标记耦合：用一个复杂的数据结构来从一个构件到另一个构件传送信息，而且传递的是该数据结构本身。
（5）数据耦合：构件间通过传递数据来完成信息的传递。
（6）非耦合

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。

内聚：指构件内部元素（比如数据、功能、内部模块）的“粘合”程度，从低到高可分为：
（1）巧合内聚：构件的各部分互不相关
（2）逻辑内聚：当一个模块中的各个部分只通过代码的逻辑结构相关联时。
（3）时态内聚：设计被划分为几个用来表示不同执行状态的模块：初始化、读进输入、计算、打印输出、和清除，这样的内聚是事态内聚，在这种模块中数据和功能仅仅因在一个任务中同时被使用而形成联系。
（4）过程内聚：通常必须按照某个确定的顺序执行一系列功能，如果模块中的功能组合在一起只是为了确保这个顺序，叫过程内聚。
（5）通信内聚：围绕着同一个数据集构造的模块是通信内聚。

（6）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都是必需的；某个功能内聚的模块不仅执行设计的功能，而且只执行该功能，不执行其他任何功能。
（7）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，就是信息内聚。

8.习题 4，5. 如何改进标记耦合？？

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？了解并使用高级语言的 OO 基本编程方法和技巧。（286-291）

2.什么是设计模式?

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。
它编写了设计决策以及最好的实践，他们根据设计原则来解决一些特定的问题。主要目标是提高设计的模块化。

3.OO 设计的基本原则?

4.OO 开发有何优势?

（1）语言的一致性。我们可以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
（2）过程的一致性。OO 的过程使用数据和行为的封装形成的独立的单元。它从需求到应用实现和测试用相同语义的概念来表示系统。

5.OO 开发过程有几个步骤?

OO 需求，OO 高层设计，OO 低层设计，OO 编码和测试

6.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。

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

7.用例图、类图等对面向对象的项目开发的意义是什么?

（1）采用标准的图示化的方式说明系统的需求、进行结构设计，通常称作建模。
（2）编程语言，一种开发工具。用它去开发或实现某个系统和产品时，首先需要了解和回答用它要做什么？你可以用文档也可以用图形来说明，但 UML 表示法是最标准和通用的，UML 可让其它与你配合的人更清楚你的想法。
（3）图形是最简洁的表述思想方式。
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。

8.熟悉类图中各个类之间的基本关系分类（303-305）

继承/泛化：父类泛化子类，允许子类继承父类的属性与行为

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

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。

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。

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。

接口与实现： 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。

9.熟悉类图等的组成和画法（300-308 页）

10.知道 UML 其他图示结构的基本用途。

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:

1.为什么说编码工作是纷繁复杂甚至令人气馁？ （337 页）

2.一般性的编程原则应该从哪三？个方面考虑？

（1）控制结构：当设计转变成代码时，我们希望保留组件的控制结构，在隐含调用的面向对象设计中，控制是基于系统状态和变量而变化的。

（2）算法：在编写代码时，程序设计通常会制定一类算法，用于编写组件。

（3）数据结构：编写程序时，应该安排数据的格式并进行存储，这样的数据管理和操作才能简明易懂。

（4）通用性指导原则：有几种全局策略对于在代码中保持设计质量是很有用的。

3.论述编码阶段实现某种算法时所涉及的问题。 （342 页）

4.在编写程序内部文档时，除了 HCB 外，还应该添加什么注释信息？

hcb:头部注释板块

1.其他程序注释，对程序正在做什么，为程序提供逐行的解释。用注释将代码分解成标识主要活动的段，接着每个活动还可以分解成更小的步骤。

2.有意义的变量名和语句标记。

3.安排格式以增强理解。注释的格式能够帮助读者理解代码的目标以及代码如何实现目标的。声明的缩进和间隔能够反映基本的控制结构。
4.文档化数据：数据地图对于当系统处理很多不同类型和目的的数据时是很有用的。

5.什么是极限编程？（XP）以及派对编程？

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。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 。
XP 的支持者强调敏捷方法的 4 个特性：交流、简单性、勇气以及反馈。
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）

2.将软件缺陷进行分类的理由？（367 页）

3.几种主要的缺陷类型?

（1）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。
（2）计算故障和精度故障：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。
（3）文档故障：文档与程序实际做的事情不一致。
（4）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
（5）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
（6）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，就会出现计时故障。
（7）吞吐量故障（性能故障）：系统不能以需求规定的速度执行
（8）恢复故障：当系统遇到失效时，不能表现的像设计人员希望的或者客户要求的那样
（9）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。
（10）标准和过程故障：程序员未能遵循必须的标准。

4.什么是正交缺陷分类?

故障被分为不同的类别，被分类的任何一个只能属于一个类别，这种分类方案就是正交的。分为疏漏型故障和犯错型故障。

5.测试的各个阶段及其任务?

（1）单元测试：将每个程序构件与系统中其它构件隔离，对其本身进行测试。验证针对设计预期的输入类型，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需求规格说明中描述的功能（功能需求）。
（4）性能测试：将系统与软件和硬件需求的剩余部分进行比较
（5）验收测试：确定系统是按照客户的预期运转的
（6）安装测试：确保系统将按照它应该的方式来运行（在用户环境下）

6.测试的态度问题？（为什么要独立设置测试团队？）（373 页）

7.测试的方法—黑盒、白盒的概念?

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黑盒，在测试的时候向黑盒提供输入数据，并记录产生的输出。在这种情况下，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。

白盒测试：将测试对象看作是一个透明的盒子，然后根据测试对象的结构用不同的方式来进行测试。例如，可以设计执行构件内所有语句或所有控制路径的测试用例。

8.什么是单元测试？什么是走查和检查?

走查：程序员向评审小组提交代码及其相关文档，然后评审小组评论他们的正确性。在走查的过程中，程序员领导并且掌控讨论。注意力集中在代码之上，而不是集中在编码者身上。

审查：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。在审查的过程中，由一个小组主持人来单人会议的领导。

9.黑盒白盒方法各自的分类？测试用例的设计和给出方法。（课件）

——ZWY-P13

黑盒：等价分类法、边界值分析法、错误猜测法、因果图法。

白盒：逻辑覆盖法、路径测试法。

10.黑盒白盒方法的分类，各种覆盖方法等。（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。

11.考虑如何面对一个命题，设计和给出测试用例的问题。

------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? （桩和驱动程序，课件）

1.自底向上集成：通过合并构件来测试较大型系统的流行方法

2.自顶向下集成：顶层构件通常是一个控制构件，独立进行测试，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。

3.一次性集成：当所有构件都分别经过测试，再将他们合在一起作为最终系统进行测试看是否能够一次运行成功。
4.三明治集成：将自顶向下和自底向上结合起来，将系统看成三层，目标层处于中间，在顶层使用自顶向下，底层使用自底向上。

13.传统测试和 OO 测试有何不同？OO 测试有何困难？

主要表现在测试用例生成、源代码分析、覆盖分析、测试开发。
1.需求可能在需求文档中表述，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.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.由于对象的交互性是复杂性的根源，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作用小。

14.测试计划涉及的几个步骤？（400 页）（了解）

chapter09:

1.系统测试的主要步骤及各自含义？

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需求规格说明中描述的功能（功能需求）。
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：确定系统是按照客户的预期运转的
安装测试：确保系统将按照它应该的方式来运行（在用户环境下）

2.什么是系统配置？软件配置管理？基线？（423 页）（或见课件）

3.什么是回归测试？（425 页）

4.功能测试的含义及其作用？

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。

5.功能测试的基本指导原则？

1.具有很高的故障检测概率
2.使用独立与设计人员和程序员的测试小组
3.了解期望的动作和输出
4.测试合法和不合法输入
5.永远不要为了使测试更容易而去修改系统
6.指定停止测试标准。

6.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。

7.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。

9.什么是可靠性、可用性和可维护性？（438 页）

10.确认测试，确认测试分类？（基准测试和引导测试）

验收测试含义：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：
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。
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。
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。

11.什么是α测试？β测试？

试验性测试中，公司内部组织机构进行的内部测试叫α测试，而客户的试验成为β测试。

12.什么是安装测试？

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。