**软件工程-zhy**

# 第一课

软件工程：软件工程就是用工程化的方法进行软件开发

**工程化**：任何生产活动都要按照目标化、规范化、文档化、标准化进行。

**软件**=程序+数据+文档

**软件特性**：复杂性、易变形、不可见性、服从性、非连续性

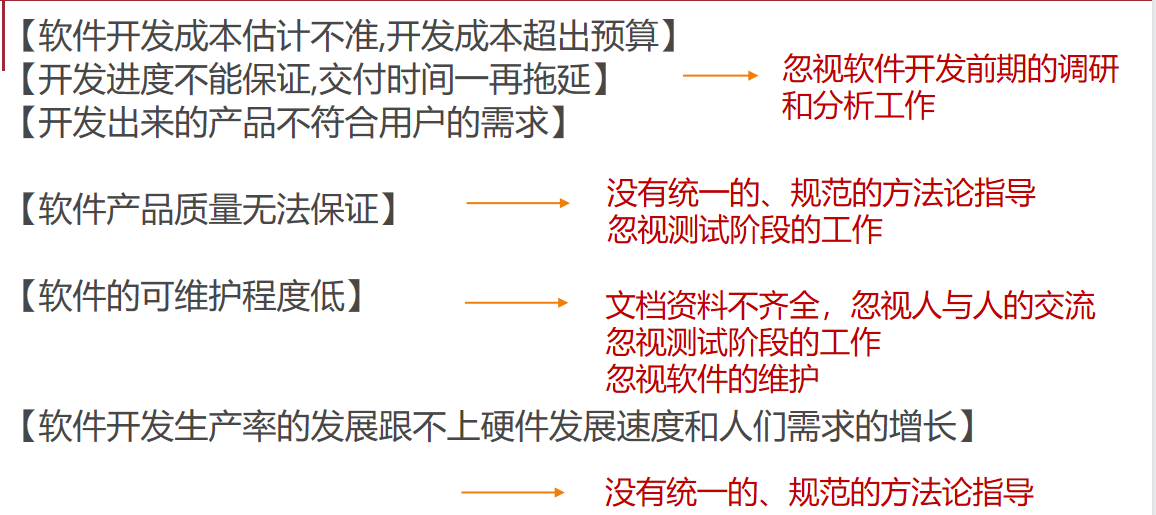
## 第二次课

#### 软件危机：

表现：ppt 2 11-15了解

概念：在计算机软件的开发和维护过程中所遇到的一系列严重问题。

原因：



#### 软件工程：

概念：软件工程就是用工程化的方法进行软件开发

发展：

三要素：过程、方法、工具

过程:支持软件开发各个环节
的控制和管理

方法:完成软件开发任务的
技术手段

工具:为软件开发方法提供自动的或半自动的软件支撑环境

#### 软件生命周期：

概念：软件从功能确定、设计到开发成功投入使用，并在使用中不断地修改、增补和完善，直到被新的需要所代替而停止使用该软件的全过程。

## 第三次课

#### 软件生命周期：

概念：

任务：软件定义、需求开发、软件设计、软件构造、软件测试、软件维护

软件定义、软件开发、运行维护（8个阶段）：1问题定义 2可行性研究3需求分析 4总体设计 5详细设计 6实现 7测试 8运行维护

#### 软件过程模型：

瀑布模型：适合于用户需求明确、完整、无重大变化的软件项目开发

瀑布模型的成功在很大程度上是由于它基本上是一种文档驱动的模型。

原型模型：主要用于不确定的软件开发 类型：抛弃型、演化型

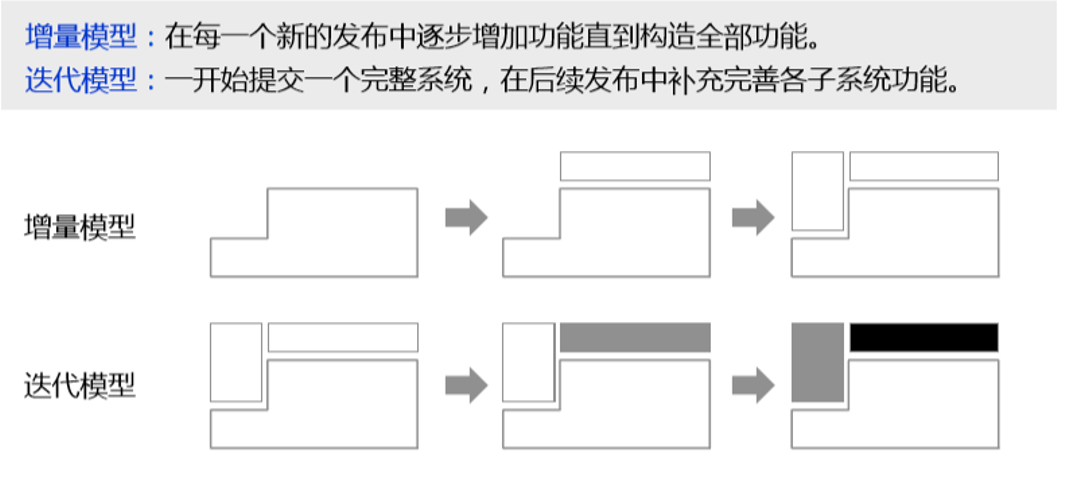
在用户不能给出完整、准确的需求说明，或者开发者不能确定算法的有效性、操作系统的适应性或人机交互的形式等许多情况下，可以根据用户的一组基本需求，快速建造一个原型。

增量模型：是渐进地开发逐步完善的软件版本的模型。 不断应用瀑布模型和原型模型

螺旋模型：降低风险，适用于规模大的复杂系统。 讲瀑布模型和增量模型加入风险分析

喷泉模型：自下而上，用户需求为动力，支持面向对象

迭代开发：整个项目被拆分成一系列小的迭代。通常一个迭代的时间都是固定的，不会太长，例如 2-4 周。每次迭代只实现一部分功能，做能在这个周期内完成的功能。



## 第四次

软件生命周期：问题定义 可行性研究 需求分析 总体设计 详细设计 实现 测试 维护

软件过程模型：瀑布模型 原型模型 增量模型 螺旋模型 喷泉模型

#### RUP过程模型（统一过程模型）：

开发生命周期(四个阶段、九个核心工作流)：

四个阶段：初始阶段、构建阶段、细化阶段、提交阶段

九个核心工作流：6个核心过程工作流：业务建模、需求、分析与设计、实现、测试、部署，

3个核心支持工作流：配置与变更管理、项目管理、环境

特点：RUP是风险驱动的、基于UseCase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。

#### 敏捷开发：

特点：以人为核心、循环迭代、响应变化

四条基本价值观：个体和交互、可以工作的软件、客户合作、响应变化

开发模型的特点：

scrum：Scrum以英式橄榄球争球队形（Scrum）为名，有明确的目标、高度自主权，紧密的沟通合作，高度的弹性迎接各种变化和挑战，确保每个阶段都朝着目标推进。

极限编程(XP)，是一种“轻量型”的以编码为核心任务的灵活软件开发方法。敏捷开发中有名的一个

XP目标：在最短的时间内将较为模糊、变化较大的用户需求转化为符合用户要求的软件产品。

开发流程：

XP软件开发过程：聆听、测试、编码、设计的迭代循环过程

Scrum和极限编程的特点：scrum偏重项目管理、XP偏重项目实践

RUP（统一过程模型）与XP（极限编程）：

#### 项目计划阶段：

问题定义、可行性研究、项目计划

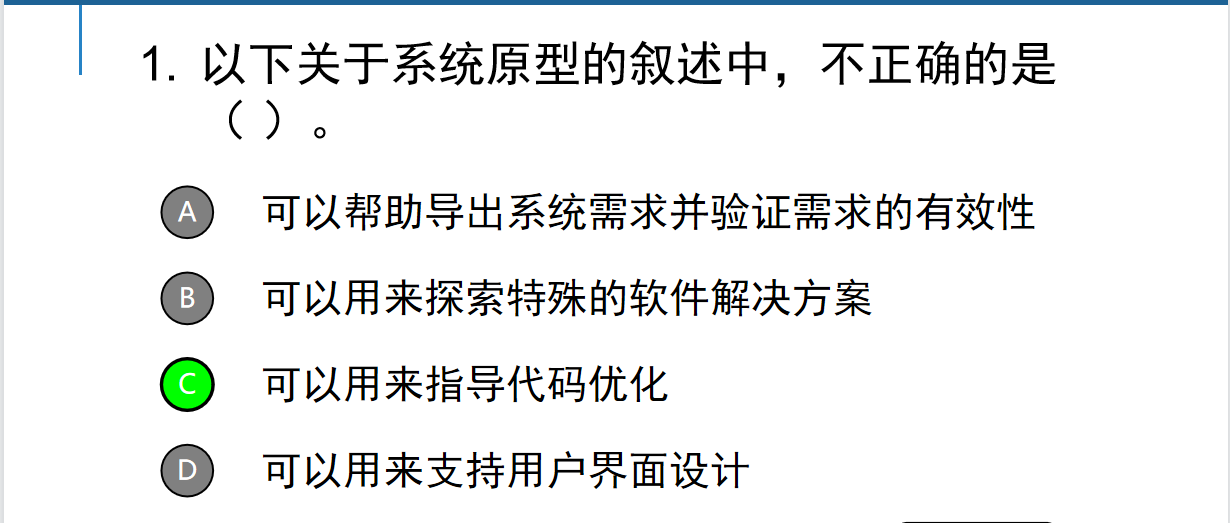
问题定义目的：弄清软件的性质、规模和限制条件，写出问题定义报告。

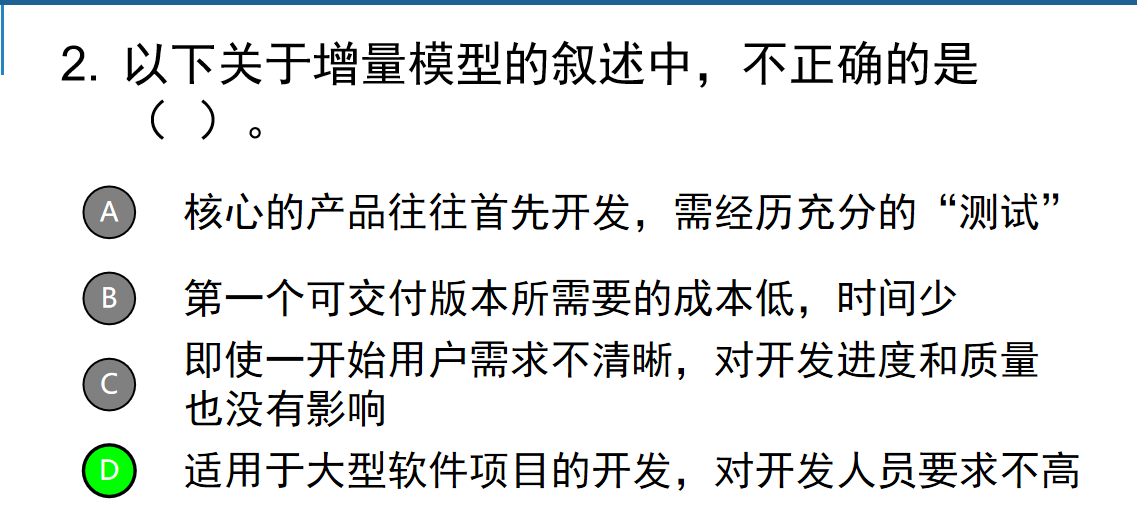
可行性研究：从技术、经济和法律角度弄清该软件是否值得开发，写出可行性分析报告。 结论：可行、基本可行、不可行

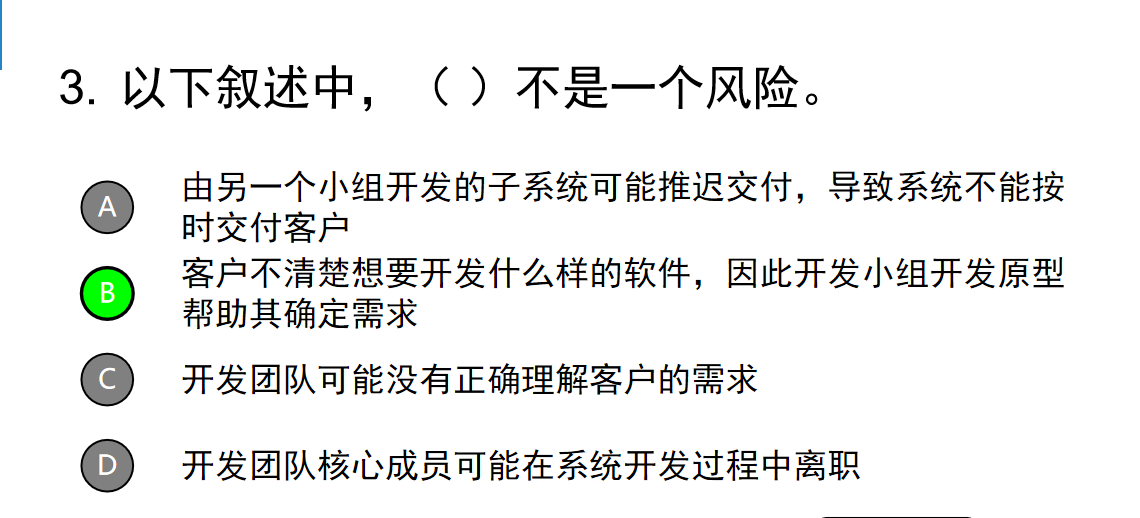
项目计划：确定要开发之后，对项目进度、所需资源进行规划，写出项目开发计划书。

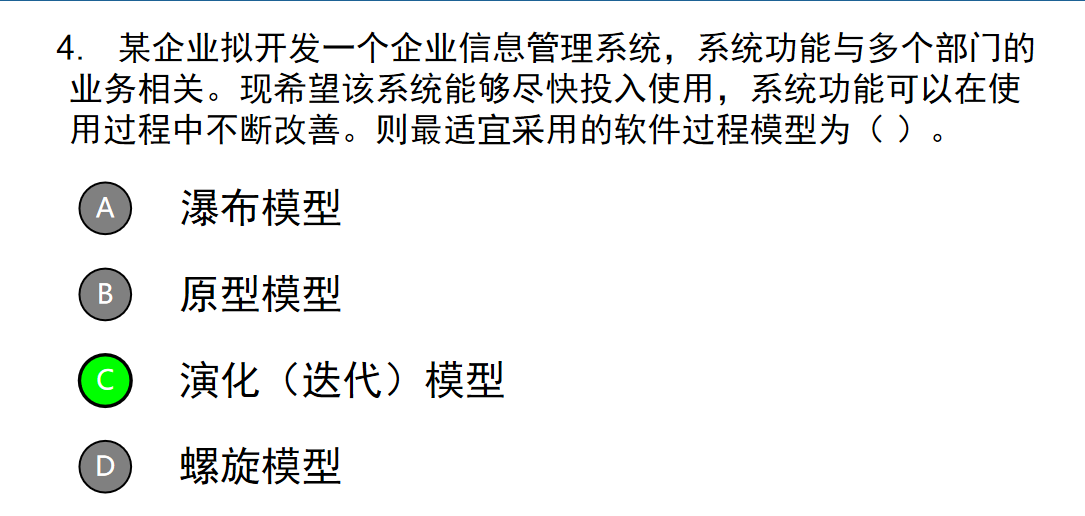
软件项目分为两种类型：委托式加工和开发市售产品

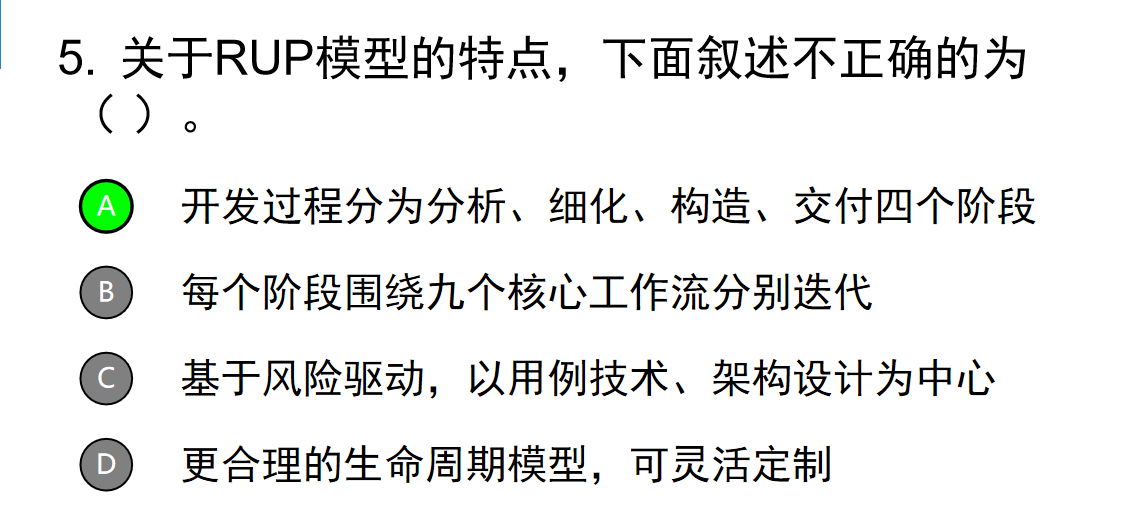
## 第五次

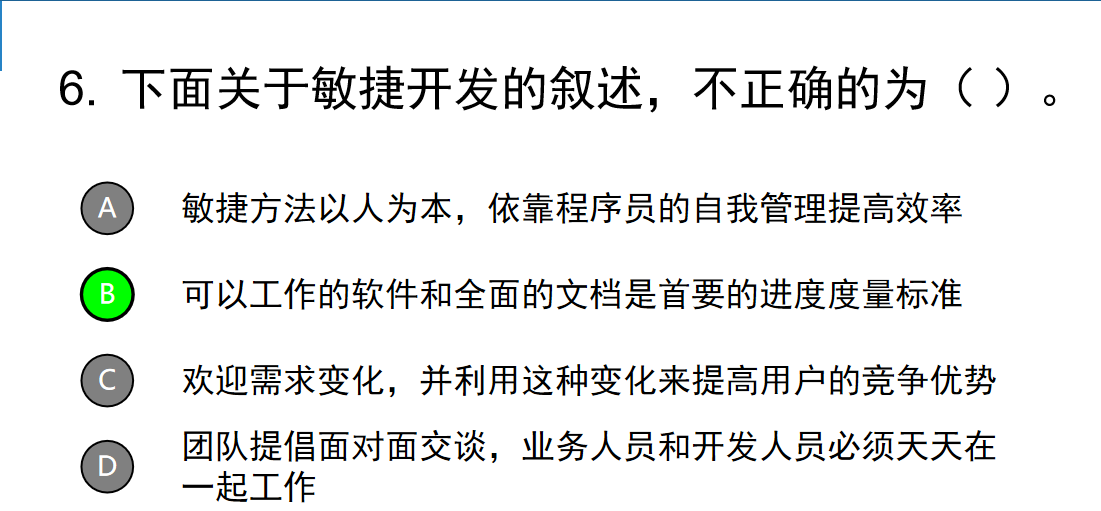


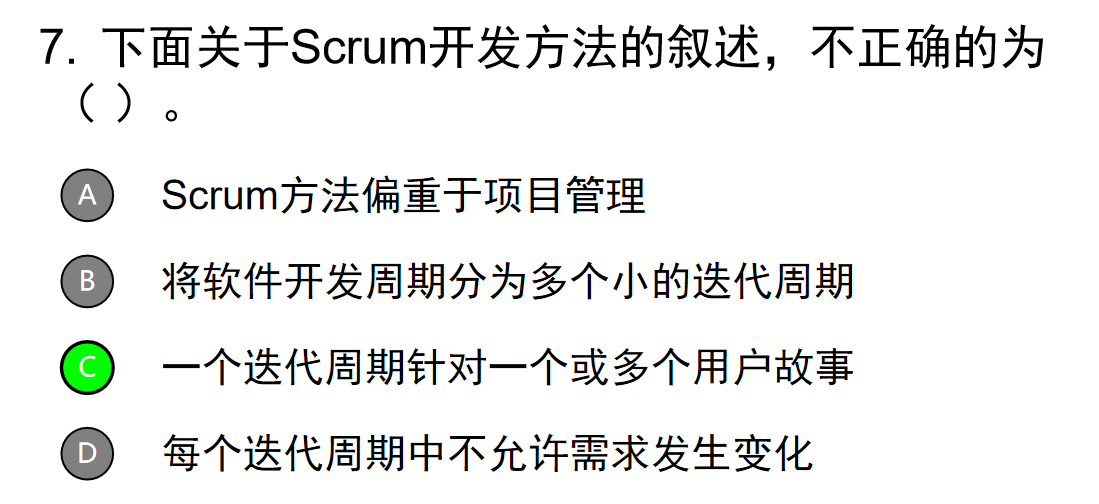




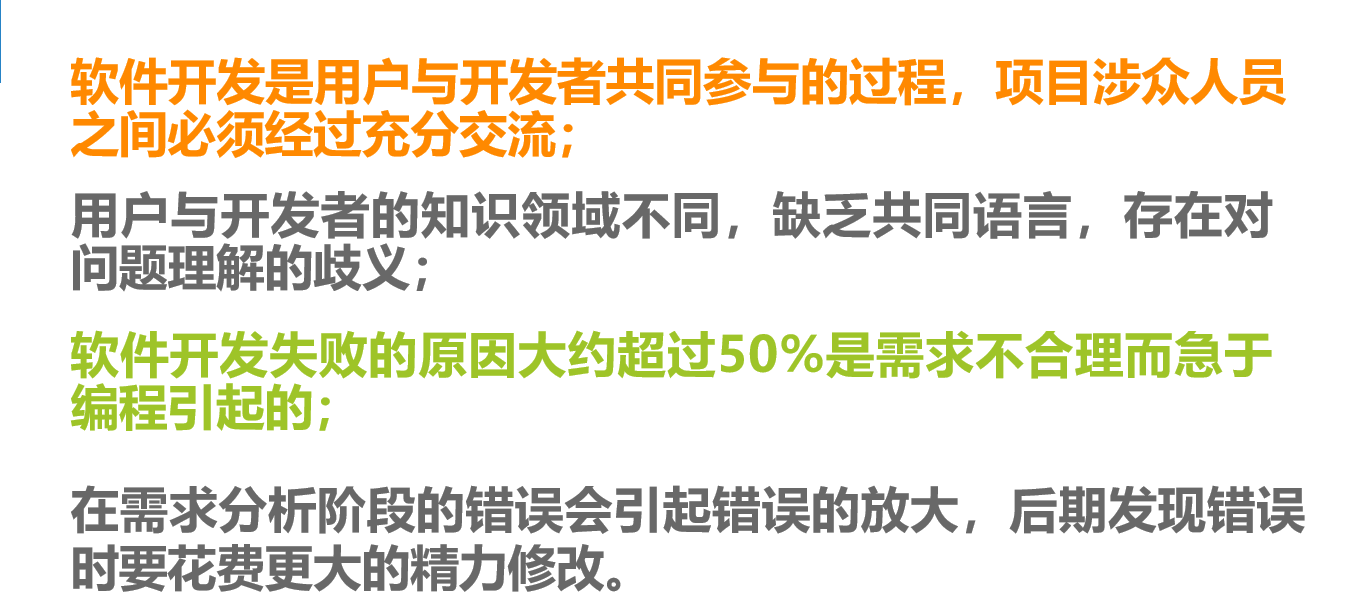








认识到需求分析的重要性:



软件过程模型：传统模型、RUP模型、敏捷模型（scrum和xp xp重要）

了解需求分析阶段的工作任务：系统必须做什么

了解 内容：需求是系统为满足客户期望的目标而完成的行为，要体现出对问题领域的清晰理解，给出系统的使用场景和上下文。过程：

需求不同层次：业务需求、用户需求、功能需求（系统需求） 三个写前三个 ppt5.25

需求工程：需求开发、需求管理

获取需求的方法：

第一阶段：访谈式、第二阶段：诱导式、第三阶段：确认式

需求建模方法：结构化分析建模SA、面向对象分析建模OOA

## 第六次

需求工程过程：需求获取、分析需求、文档编写、需求验证、需求变更

UML（统一建模语言）语言概述：它不仅统一了Booch、Rumbaugh和Jacobson的表示方法，而且对其做了进一步的发展，并最终统一为大众所接受的标准建模语言。

## 第七次

个人感觉这一章没概念题

用例图如何画也在这一章

建立用例模型：1 定义系统 2确定执行者和用例 3 描述执行者和用例关系 4确认模型

UML五类图：用例图、静态图（类图、对象图和包图）、行为图（状态图和活动图）、交互图（顺序图和协作图）、实现图（组件图和配置图）

系统用例建模：

用例描述：简要描述、前置条件、基本事件流、其他事件流、后置条件等

## 第八次 活动图

软件开发方法发展：面向过程、面向对象、面向组件、面向服务面向云计算

#### 结构化软件工程方法

结构化需求分析（SA）：数据流图、数据字典

结构化设计(SD):模块结构图、程序流程图

结构化编程、结构化测测、结构化维护

#### 面向对象软件工程方法：

面向对象分析、面向对象设计、面向对象编程、面向对象测试、面向对象维护

面向对象建模：功能模型、对象模型、动态模型

并发活动：分岔和汇合

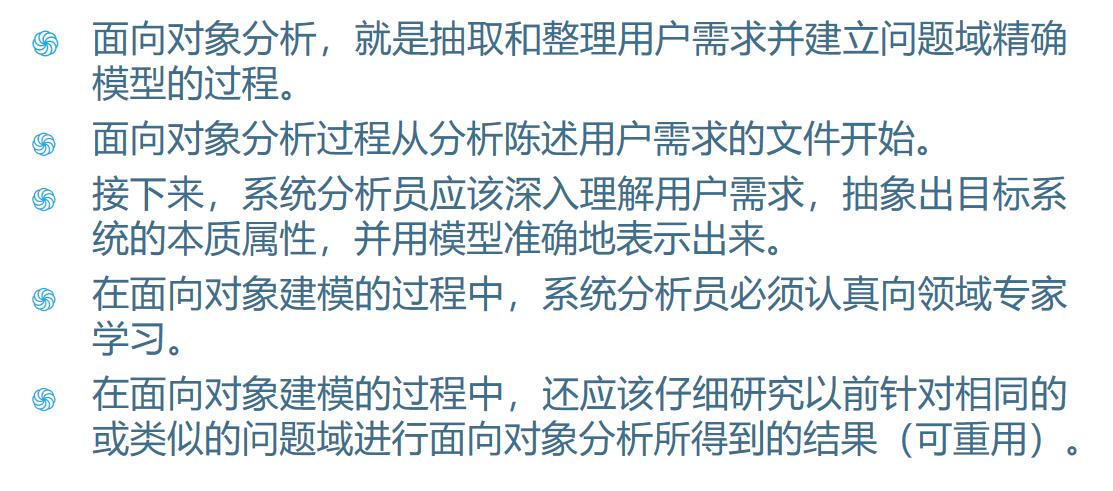
泳道：表示各个活动由谁负责的信息

#### 面向对象方法学：

面向对象的概念：

类和对象
属性和方法
抽象和封装
消息传递
继承和多态、重载

面向对象分析：



#### 需求建模：

需求类型：非功能性需求和功能性需求

#### 获取非功能需求：

几个方面：

可靠性：安全性、事务性、稳定性

可用性：

有效性：性能、可伸缩性、可扩展性

可移植性：

#### 撰写需求规格说明书：重要的过程文档

## 第九次 类图

分类：领域类图、实现类图

类：名称、属性、操作 接口：名称 方法

类的抽象层次：概念车、说明层和实现层

关系：依赖、泛化、实现、关联

特殊关联关系：聚合关系、合成关系

不同层次的类：实体类、控制类、边界类

领域模型的三大模型：视图模型、逻辑模型、实体模型

# 第十次 数据库设计

软件设计两部分：总体设计和详细设计

总体设计：在需求规格说明书的基础上描述系统的架构、功能模块的划分、模块接口的定义、用户界面设计、数据库设计 严谨编写数据字典

详细设计：在需求规格说明书的基础上描述系统的架构、功能模块的划分、模块接口的定义、用户界面设计、数据库设计

数据的持久化：指要永久使用的数据或系统产生的数据如何保持

常见保存方式：数据库管理系统和文件存储模式

#### 系统设计

系统设计阶段的主要任务
数据库设计

#### 系统的静态结构设计

软件架构提供了组件和容器  
体系结构相关知识  
MVC设计模式：模型（model）对象、视图对象（View）、控制器对象（Control）  
DAO设计模式：功能：1 封装数据源 2封装对数据库的操作  
系统包图：包的可见性：共有访问、私有访问、保护访问

## 第十一次

一个构件就是程序的一个功能要素，也称为模块。包括：程序的处理逻辑、需要的内部数据、保证构件被调用和实现数据传递的接口。

### 软件设计的基本原理和相关概念

### 模块化：软件系统越来越复杂，需要将其划分为若干部分分而治之。

#### 抽象：为了降低问题的复杂度，以得到问题领域中较为简单的概念。

支持过程抽象和数据抽象

过程抽象：把用文字形成的功能描述演变成代码。

数据抽象：文字表达的数据结构演变成数据库表和存储过程。数据抽象与过程抽象一样，能使设计者按不同的详细程度表示数据对象。

#### 信息隐蔽和局部化：：为了避免某个模块的行为干扰同一系统中的其他模块，把必要的信息隐藏起来。

信息隐藏是实现抽象、模块化机制的基本支撑。

#### 模块独立：每个模块只完成特定的单一功能，不同模块之间的联系尽可能少，尽可能减少公共变量和数据结构，有助于系统维护及软件的复用。

两个定性标准度量：耦合度和内聚度

独立性标准：高内聚、低耦合

耦合度七个等级：① 非直接耦合 ② 数据耦合 ③ 特征耦合④ 控制耦合 ⑤ 外部耦合⑥ 公共耦合⑦ 内容耦合

三种内聚：服务内聚、类内聚、一般-特殊内聚

内聚七个等级：偶然性内聚
逻辑性内聚
时间性内聚
过程性内聚
通讯性内聚
顺序性内聚
功能性内聚

深度/宽度、扇入/扇出 深度控制层数、宽带控制跨度、扇出数 直接控制其他模块数（依赖性）扇入数 能直接控制的上级模块数（复用性）

软件重用：是指同一事物不作修改或稍加改动就多次重复使用。

层次:知识重用、方法和标准的重用、软件成分的重用

软件成分重用三个级别：1几种形式的代码重用、 2设计结果重用、3分析结果重用

面向对象设计原则：抽象，信息隐蔽，模块独立，高内聚，低耦合 主要前五个原则

单一责任原则、依赖倒置原则、里氏替换原则、开放封闭原则、接口隔离原则、合成复用原则、迪米特法原则。

链是对象间的发送消息的路径

## 十二课 状态图

状态图：状态和转换

### 建立系统状态模型

状态图建模技术：使用状态图一般是对系统中反映型对象建模，特别是对类、用例和系统实例的行为建模。

#### 系统交互界面设计

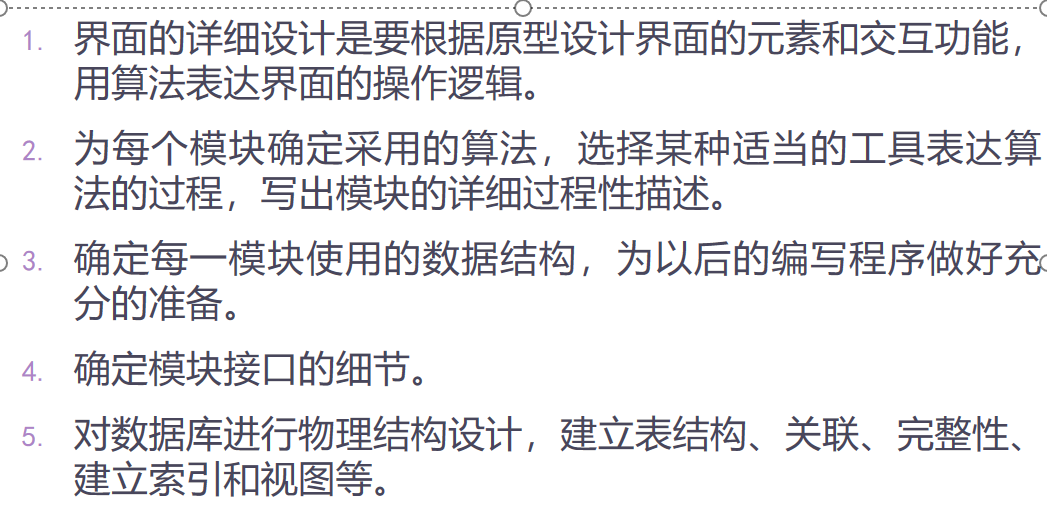
人机界面设计原则：

1让用户拥有控制权   
2减少用户的记忆负担   
3保持界面一致

## 第十三次

### 软件的详细设计

详细设计的工作任务



面向对象的详细设计：属性的数据结构设计、方法的设计、实现接口所需机制的设计

详细设计的工具

程序流程图（重要）
盒图（N-S图）（看懂）
PAD图（了解）
判定表（会用）
判定树（会用）

### 软件的编码实现

为了执行规范，每个软件开发人员必须一致遵守编码规范

编写高质量代码和编码标准规范
代码审查和结对编程

## 第十四次

### 软件系统的测试

四个方面理解软件测试：

软件测试贯穿软件开发的整个生命周期

测试是为了证明程序有错，而不是证明程序无错误。

测试包含了“分析”或“运行”软件。

软件测试有两个基本功能：验证和确认

测试用例设计的内容：正确数据、错误数据、边界数据。

单元测试、集成测试、系统测试、验收测试  
软件测试的概念：软件测试，是为了发现错误而执行程序的过程。  
**软件测试的方法：**

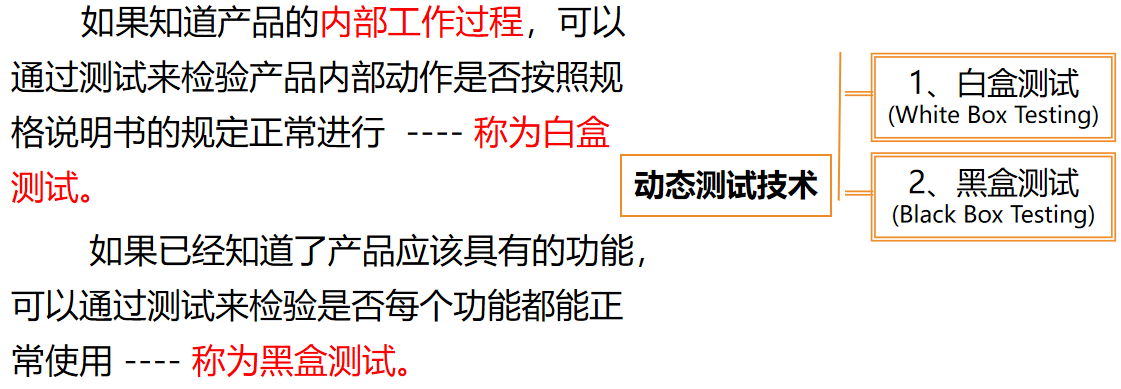
静态方法:通过人工分析或程序正确性证明的方式来确认程序正确性。

人工测试方法、计算机辅助静态分析方法 静态方法

动态方法:通过输入数据，运行软件，观察执行状态和结果，来检验软件的动态行为和运行结果的正确性

白盒测试方法、黑盒测试方法 动态方法 动态测试的两个基本要素：被测试程序、测试数据

测试用例的三个要素：前提条件和操作步骤、预期结果、实际结果

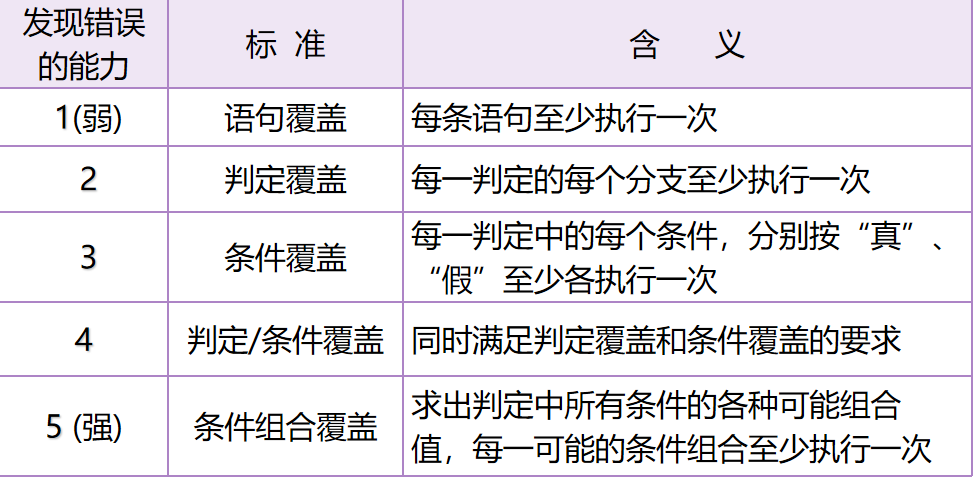


#### 白盒测试技术：

逻辑覆盖：是以程序内部的逻辑结构为基础的设计测试用例的技术

五种逻辑测试标准 57页看一下 知道分别是什么意思

语句测试、判断测试、条件测试、判断覆盖、判断/条件覆盖、条件组合覆盖



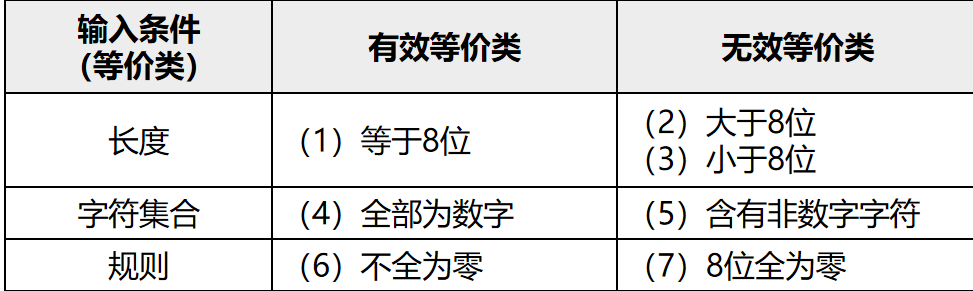
**基本覆盖法：**点覆盖、路径覆盖 65页 自己看一下知道什么是基本覆盖

#### 黑盒测试

黑盒测试技术：**等价划分法**、边界值分析法、错误推测法、因果图法、决策表法、场景法等

如何划分等价类：**有效等价类、无效等价类**  标准：覆盖、不相交、代表性

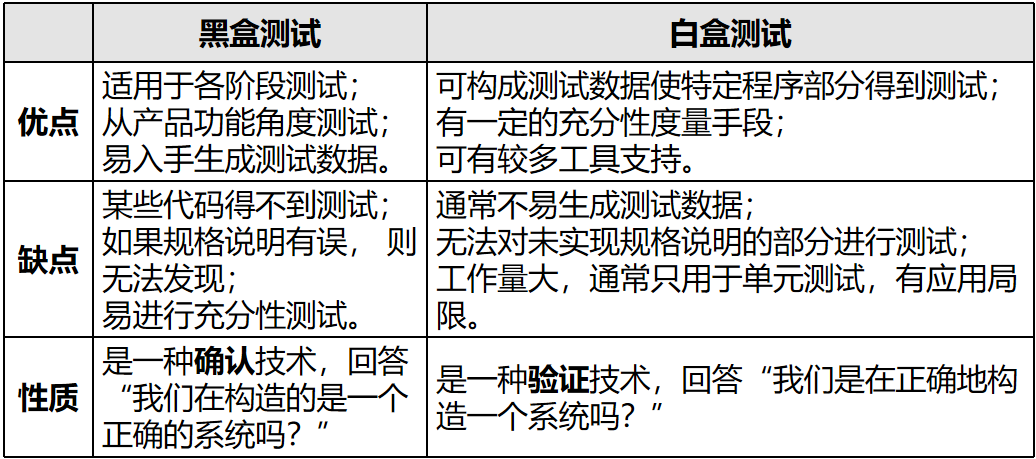
等价类划分举例



边界值分析法：测试内点和测试外点



黑盒白盒比较



软件测试的过程和类型:

单元测试： 额外建立两种模块：桩模块、驱动模块

集成测试： 增量集成法和非增量集成法

系统测试，验收测试：



面向对象测试策略：类内测试、类间测试、基于场景的测试

白盒测试和黑盒测试是软件测试的两类基本方法

测试包含静态测试和动态测试,测试步骤至少分为:

模块测试 --- 单元
子系统测试 --- 局部
系统测试 --- 集成
验收测试 --- 用户参与

## 第十五次

调试就是把症状和原因联系起来的尚未被人深入认识的智力过程。

软件的调试的方法：蛮干法、回溯法、原因排除法  
软件维护的类型与过程：

维护的类型：改正性维护、适应性维护、扩充与完善性维护、预防性维护

软件维护过程：维护就是修改软件；另一个关键词是“过程”，过程暗喻不是单一的活动，是活动的集合和活动的有序组合，过程就需要管理。

文档类型：用户文档和系统文档

软件再工程过：**预防性维护**又称为软件再工程。指的是重新构造或编写现有系统的一部分或全部，但不改变其功能。

重构的概念：就是在不改变软件现有功能的基础上，通过调整程序代码改善软件的质量、性能，使其程序的设计模式和架构更趋合理，提高软件的扩展性和维护性。  
 重构与增强不同时进行 戴一顶帽子只做事

软件维护的4类活动：
（改正性、适应性、完善性、预防性）
决定软件可维护性的基本要素：
（可理解、可测试、可修改、可移植和可重用性）
文档是影响软件可维护性的决定因素