**软件工程复习纲要**

**软件工程复习纲要 1**

**1概述 4**

1.1 软件 4

*1.1.1* *软件的概念(H) 4*

*if*

*1.1.2* *软件的特征(M) 4*

1.2 软件危机 4

*1.2.1* *软件危机的概念(M) 4*

*1.2.2* *软件危机的产生原因(H) 4*

1.3 软件工程 4

*1.3.1* *软件工程的定义(H) 4*

*1.3.2* *软件工程的三要素(H) 4*

*1.3.3* *软件工程的发展过程(L) 4*

1.4 其他 4

**2** **过程模型 4**

2.1 软件生命周期概念(L) 4

2.2 软件过程概念(M) 4

2.3 能力成熟度模型CMM概念(L) 5

2.4 软件过程模型(M) 5

*2.4.1* *瀑布(H) 5*

*2.4.2* *增量(H) 5*

*2.4.3* *原型(H) 5*

*2.4.4* *螺旋(H) 5*

*2.4.5* *喷泉(M) 5*

*2.4.6* *敏捷(M) 5*

2.5 ，其他 5

**3** **需求分析 5**

3.1 需求分析的概念(H) 5

3.2 需求分析的过程(L) 6

*3.2.1* *需求确认(H) 5*

3.2.1.1 需求获取(M) 5

3.2.1.2 需求提炼(M) 5

3.2.1.3 需求描述(M) 6

3.2.1.4 需求验证(M) 6

*3.2.2* *需求变更(L) 6*

3.3 需求分析三类建模(H) 6

*3.3.1* *功能模型(M) 6*

3.3.1.1 SA功能模型：数据流图(H) 6

3.3.1.2 OOA功能模型：用例图(H) 6

*3.3.2* *数据模型(L) 6*

*3.3.3* *行为模型(L) 6*

3.4 其他 6

**4** **系统设计 6**

4.1 系统设计(M) 6

*4.1.1* *概念 6*

*4.1.2* *概要设计(L) 6*

*4.1.3* *详细设计(L) 6*

4.2 与设计相关的8个概念 7

*4.2.1* *抽象(L) 7*

*4.2.2* *体系结构(H) 7*

*4.2.3* *设计模式(L) 7*

*4.2.4* *模块化(H) 7*

*4.2.5* *信息隐藏(H) 7*

*4.2.6* *功能独立(H) 7*

*4.2.7* *细化(L) 8*

*4.2.8* *重构(L) 8*

4.3 系统设计四方面 8

*4.3.1* *体系结构设计(M) 8*

*4.3.2* *数据设计(L) 8*

*4.3.3* *接口设计(L) 8*

*4.3.4* *组件设计 8*

4.3.4.1 SD组件设计：流程图(H) 8

4.3.4.2 OOD组件设计：顺序图(H) 8

4.4 其他 8

**5** **质量保证 8**

5.1 质量保证的概念(L) 8

5.2 测试策略V模型 8

*5.2.1* *测试四级别(H) 8*

5.2.1.1 单元测试(M) 8

5.2.1.2 集成测试(M) 9

5.2.1.3 系统测试(M) 9

5.2.1.4 验收测试(M) 9

*5.2.2* *测试各级别与开发的各阶段对应关系(H) 9*

5.3 回归测试的概念(L) 9

5.4 测试的常见术语 9

*5.4.1* *软件缺陷(H) 9*

*5.4.2* *验证和确认(L) 9*

*5.4.3* *测试与质量保证(M) 9*

*5.4.4* *质量与可靠性(L) 9*

*5.4.5* *调试与测试(L) 9*

*5.4.6* *测试用例(H) 9*

5.5 测试技术 9

*5.5.1* *白盒测试 9*

5.5.1.1 逻辑覆盖(H) 9

5.5.1.2 控制流图覆盖(M) 9

*5.5.2* *黑盒测试 10*

5.5.2.1 等价类划分(H) 10

5.5.2.2 边界值分析(M) 10

*5.5.3* *静态分析(L) 10*

5.6 其他 10

**6** **软件维护 10**

6.1 软件维护的基本概念(M) 10

6.2 软件维护的四个基本类型(H) 10

*6.2.1* *纠错性维护(M) 10*

*6.2.2* *适应性维护(M) 10*

*6.2.3* *完善性维护(M) 10*

*6.2.4* *预防性维护(M) 10*

6.3 可维护性的决定因素(L) 11

6.4 软件维护过程模型(L) 11

6.5 软件再工程(L) 11

6.6 逆向工程(L) 11

6.7 其他 11

**7** **项目管理 11**

7.1 项目管理四要素(H) 11

7.2 软件度量 12

*7.2.1* *度量 12*

7.2.1.1 基于规模(H) 12

7.2.1.2 基于功能点(M) 12

*7.2.2* *估算 12*

7.2.2.1 算法成本模型(L) 12

7.2.2.2 COCOMO模型(M) 12

7.3 项目计划 12

7.4 其他 12

1. **概述**

## **1.1软件**

### **1.1.1软件的概念(H)**

蓝老师：软件=程序+文档+数据

熊不是一般的熊：

程序：按事先设计的功能和性能需求执行的指令序列

数据：是程序能正常操纵信息的数据结构

文档：与程序开发、维护和使用有关的图文材料

### **1.1.2软件的特征(M)**

冂冂马：

1. 软件是开发的
2. 软件是简单的拷贝
3. 软件测试非常困难
4. 软件需要维护，维护易产生新的问题
5. 软件开发时间和工作量难以估计
6. 软件开发时间进度几乎没有客观衡量标准
7. 软件不会磨损，但会退化和废弃

## **1.2软件危机**

### **1.2.1软件危机的概念(M)**

404:

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。

冂冂马：

例如：①项目超出预算②项目超过计划完成时间③软件运行效率很低④软件质量差⑤软件通常不符合要求⑥项目难以管理并且代码难以维护⑦软件不能交付

**1.2.2软件危机的产生原因(H)**

404:

1. 软件自身特点：

a、软件是逻辑部件，缺乏可预见性

b、软件维护困难

c、软件越来越庞大复杂，需要分工协作

1. 软件开发和维护的方法不正确

a、需求分析不充分或者存在错误

b、开发的过程不规范

c、不注重文档工作，软件难以维护

d、缺少软件评测手段

## **1.3软件工程**

### **1.3.1软件工程的定义(H)**

404:

1. 应用系统的、规范的、可度量的方法来开发，运行和维护软件，即把工程应用到软件。
2. 对（1）中各种方法的研究。

### **1.3.2软件工程的三要素(H)**

第四人称：

方法、工具、过程

方法：软件工程方法是完成软件工程项目的技术手段。它支持项目计划和估算、系统和软件需求分析、设计、编程、测试和维护。软件工程方法依赖一组原工具开发的系统称“计算机辅助软件工 程(CASE)”，系统中某一工具的信 息加工结果可以作为另一工具的输 入。 集成的软件工程工具再加上人 的因素构成了软件工程环境。则，它贯穿软件工程的各个环节。软件工程方法分两类:结构化方法和面向对象方法。

过程：过程贯穿软件开发的各个环节，在各环节 之间建立里程碑; 管理者在软件工程过程 中对软件开发的质量、进度、成本进行评 估、管理和控制; 技术人员采用相应的方法和工具生成软件工程产品(模型、文档、数据、报告、表格等)。

工具：它为软件工程的过程和方法提供自 动化或半自动化的工具支持。 将若 干工具集成起来，与软件工程数据 库和计算机系统构成一个支持软件软件工程方法是完成软件工程项目的技术手段。它支持项目计划和估算、系统和软件需求分析、设计、编程、测试和维护。软件工程方法依赖一组原开发的系统称“计算机辅助软件工 程(CASE)”，系统中某一工具的信 息加工结果可以作为另一工具的输 入。 集成的软件工程工具再加上人 的因素构成了软件工程环境。

### **1.3.3软件工程的发展过程(L)**

第一代:传统软件工程（结构化方法学60s末-70s）将软件工程纳入工程化的轨道

第二代:对象工程（面向对象方法学80s中-90s）面向对象的分析与设计

第三代:过程工程（80s中开始）提高软件生产率，保证软件质量的关键，“软件过程”是软件开发和维护中的管理和支持能力，形成软件过程工程。

第四代:构建工程（90s起）可复用构件

## **1.4其他**

1. **过程模型**

## **2.1软件生命周期概念(L)**

软件产品或软件系统从设计，投入使用到被淘汰的全过程

## **2.2软件过程概念(M)**

软件过程定义了软件生产的一系列活动，这些活动贯穿于软件开发的整个过程

## **2.3能力成熟度模型CMM概念(L)**

冂冂马：

概念：软件能力成熟度模型是一种对软件组织在定义、实施、[度量](https://wapbaike.baidu.com/item/%E5%BA%A6%E9%87%8F/6990327)、控制和改善其软件过程的实践中各个发展阶段的描述形成的标准。

分级：

1. 初始级
2. 可重复级
3. 已定义级
4. 量化管理级
5. 优化级

## **2.4软件过程模型(M)**

软件过程模型是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。也称为:软件开发模型、软件生存周期模型

### **2.4.1瀑布(H)**

1. 软件开发过程与软件生命周期是一致的，也称经典的生命周期模型。
2. 以文档为驱动的模型
3. 特点和优点:

①每个阶段都有与其相关联的里程碑和可交付产品

②每个阶段结束前完成文档审查，及早更正错误

③阶段间具有顺序性和依赖性

④推迟实现的观点

4.缺点:

①各个阶段划分完全固定，阶段间产生大量文档，极大增加工作量

②由于开发模型是线性的，用户只有等到整个过程末期才能看到开发结果，从而增加了开发风险

③早期的错误可能要等到开发后期的测试阶段才能发现，进而带来严重后果

④无法适应需求不明确和需求的变化

⑤不能反映实际的开发方式，软件开发需要迭代

5.适用场合:

瀑布模型适用于系统需求明确、技术成熟、工程管理较严格的场合，如银行、航天、医疗、军工等。

### **2.4.2增量(H)**

冂冂马：

优点：

1. 只有一个增量出现，开发即可进行。
2. 初期不需投入太多的人力资源
3. 增量可以有效管理技术风险
4. 软件能够更早的投入市场

缺点

1. 软件必须具备开放式体系结构（困难）
2. 易退化成边做边改的方式，使软件过程控制失去整体性
3. 每个增量必须提供一些系统功能，这使得开发者很难根据客户需求给出大小适合的增量

适用场合：适用于软件开发中需求可能发生变化、具有较大风险、或者希望尽早进入市场的项目

### **2.4.3原型(H)**

优点:

减少需求不明确带来的风险

缺点:

1. 构造原型采用的技术和工具不一定主流
2. 快速建立起来的系统加上连续修改可能导致原型质量低下
3. 设计者在质量和原型中进行折中
4. 客户意识不到一些质量问题

### **2.4.4螺旋(H)**

404：

概述:

螺旋模型沿着螺旋线，四个象限分别表示四个方面的活动

1、制定计划：确定软件目标，选定实施方案。弄清项目开发分限制条件

2、风险分析：分析所选方案，考虑如何识别和消除风险

3、实施工程：实施软件开发

4、客户评估：评价开发工作，提出修改意见

优点：

1. 支持用户需求的动态变化
2. 原型可看作形式的可执行的需求规格说明，易于用户和开发人员共同理解，还可作为继续开发的基础，并为用户参与所有关键决策提供了方便
3. 螺旋模型特别强调原型的可扩充性和可修改性，原型的计划贯穿整个软件生存周期，将有助于目标软件的适应能力。
4. 螺旋模型为项目管理人员及时调整管理决策提供了方便，进而可降低开发风险。

缺点：

1. 如果每次迭代效率不高，致使迭代次数过多，增加成本，推迟提交时间
2. 该模型需要有相当丰富的风险评估经验和专门知识
3. 对开发人员的要求较高

适用场景：

支持需求不明确，特别是大型软件开发，并支持面向规格说明，面向过程，面向对象等软件开发方法，是一种具有广阔前景的模型。

### **2.4.5喷泉(M)**

概述：

以用户需求为动力，以对象为驱动的模型，主要用于描述面向对象的软件开发过程。

优点：

1. 喷泉模型不像瀑布模型具有顺序性，必须前一阶段完成才能进行下一阶段开发，喷泉模型各个阶段没有明确的界限，开发人员可以同步进行开发，其优点是可以提高开发效率，节省开发时间。

缺点：

1、各个开发阶段重叠，开发过程需要大量的开发人员，不利于项目的管理，需要严格的管理文档，使得审核难度加大，尤其是当面对随时加入各种信息，需求和资料的情况。

### **2.4.6敏捷(M)**

墨。:

优点：

对变化和不确定性有更快更敏捷的反应

在快速的同时保持可持续的开发速度

能较好地适应商业竞争环境下对小项目提出的有限资源和有限开发时间的约束

缺点：

极限编程中的测试驱动开发可能会导致系统通过了测试但不是用户期望的

重构而不降低体系结构的质量是困难的

用于大型项目有很多问题

## **2.5其他**

基于构件的模型

概述：

基于构件的模型大概有四个阶段

1. 需求
2. 组件分析：根据需求规格说明书搜索可满足该需求的组件，通常情况下，没有完全匹配的情况，因而组件通常需要加以修改。
3. 系统设计：与其他模型的系统设计不同，该模型需要基于可重用的概念进行软件设计
4. 开发集成：将软件集成到项目系统中

优点：

组件可重用，降低了成本和风险，节约了时间

缺点：

1. 模型复杂
2. 可能导致需求折中，导致系统不能符合要求
3. 无法完全控制所开发系统的演化
4. 项目划分的好坏影响项目结果的好坏
5. **需求分析**

## **3.1需求分析的概念(H)**

afh123:确定系统必须具有的功能和性能，系统要求的运行环境，并且预测系统发展的前景。

换句话说需求就是以一种清晰、简洁、一致且无二义性的方式，对一个待开发系统中各个有意义方面的陈述的一个集合。

## **3.2需求分析的过程(L)**

### **3.2.1需求确认(H)**

#### **3.2.1.1需求获取(M)**

afh123:软件需求获取指的是软件需求的来源以及软件工程师收集这些软件需求的方法。它也称为需求抓取、需求发现和需求获得。

#### **3.2.1.2需求提炼(M)**

afh123：对应用问题及环境的理解和分析，为问题涉及的信息、功能及系统行为建立模型。将用户需求精确化、完全化，最终形成下一步的需求规格说明书

#### **3.2.1.3需求描述(M)**

404：

1. 需求分析工作完成的一个基本标志就是形成一份完整的，规范的需求规格说明书
2. 需求规格说明书的编制是为了使用户和软件开发者双方都对该软件的初始规定有一个共同的理解，使之成为整个开发工作的基础
3. 需求规格说明的原则

a、分离功能，描述做什么而不是怎样实现

b、使用面向处理的规格说明语言（系统定义语言）

c、若开发大系统中的一个元素，也需要对整个大系统进行描述

d、必须包括系统运行环境

必须是一个认识模型

必须是可操作的

必须容许不完备性并允许扩充

必须局部化和松散耦合扩充

必须局部化和松散耦合

#### **3.2.1.4需求验证(M)**

需求验证的重要性：如果在后续的开发或当系统投入使用时才发现需求文档中的错误，就会导致更大代价的返工。

### **3.2.2需求变更(L)**

变更管理是将个人、团队和组织从现有状态转移/过渡到期望状态的结构化方法。它授权雇员接受并理解当前业务环境中的变更。在项目管理中，变更管理是指项目变更被引入和接受后的项目管理过程。

## **3.3需求分析三类建模(H)**

|  |  |  |
| --- | --- | --- |
|  | 面向过程的需求分析 | 面向对象的需求分析 |
| 数据模型 | 实体-联系图（ERD）  数据字典（DD） | 类图、类关系图 |
| 功能模型 | 数据流图（DFD） | 用例图 |
| 行为模型 | 状态变迁图（STD） | 活动图、时序图、状态图 |

### **3.3.1功能模型(M)**

#### **3.3.1.1SA功能模型：数据流图(H)**

#### **3.3.1.2OOA功能模型：用例图(H)**

### **3.3.2数据模型(L)**

|  |  |  |
| --- | --- | --- |
|  | 面向过程的需求分析 | 面向对象的需求分析 |
| 数据模型 | 实体-联系图（ERD）  数据字典（DD） | 类图、类关系图 |

### **3.3.3行为模型(L)**

|  |  |  |
| --- | --- | --- |
|  | 面向过程的需求分析 | 面向对象的需求分析 |
| 行为模型 | 状态变迁图（STD） | 活动图、时序图、状态图 |

## **3.4其他**

1. **系统设计**

## **4.1系统设计(M)**

### **4.1.1概念**

### **4.1.2概要设计(L)**

就酱：

实现目标产品的总体框架包括体系结构设计、数据设计、接口和组件设计。其中体系结构设计设计师概要设计的主要内容。

### **4.1.3详细设计(L)**

就酱：

对概要设计划分出来的模块分别去描述设计，以便能够实现编码实现

## **4.2与设计相关的8个概念**

### **4.2.1抽象(L)**

就酱：

忽略具体的信息将不同的事物看成相同事物的过程，主要包括数据抽象和过程抽象

### **4.2.2体系结构(H)**

就酱：

软件的整体结构和这种结构为系统提供概念上完整性的方式？？（好拗口）

它可以使用一种后多种模型来表达

例如：

结构模型、框架模型、动态模型、过程模型、功能模型

### **4.2.3设计模式(L)**

就酱：

在给定上下文环境中一类共同问题的共同解决方案

在微观结构上表现为实体模式、结构模式、行为模式

### **4.2.4模块化(H)**

就酱：

软件被划分为命名和功能相对独立的多个组件（通常称为模块），通过这些组件的集成来满足问题的需求。

其设计标准包括：模块化的分解性（分解为子问题）、模块化的组合性（组装可重用的组件）、模块化的可理解性（可作为独立单元理解）、模块化的连续性（需求小变化只影响单个模块）、模块化的保护（模块内异常只影响自身）

### **4.2.5信息隐藏(H)**

就酱：

作为模块化基本问题之如何分解软件系统以达到最佳的模块划分，其原则是模块应该具有彼此相互隐藏的特性，即：模块定义和设计时应该保证模块内的信息（过程和数据）不可以被不需要这些信息的其他模块访问

### **4.2.6功能独立(H)**

就酱：

含义为：每个模块只负责需求中特定的子功能，并且从程序结构的其他部分看，该模块具有简单的接口

好处为易于开发、维护和测试；模块的独立性强表现为高内聚低耦合

内聚性、耦合性

### **4.2.7细化(L)**

就酱：

相对于抽象来讲，抽象使设计师确定过程和数据，但不局限于底层细节，精化有助于设计者在设计过程中揭示底层细节

### **4.2.8重构(L)**

就酱：

含义为：不改变组件功能和行为条件下，简化组件设计的一种重组技术

## **4.3系统设计四方面**

### **4.3.1体系结构设计(M)**

就酱：

数据中心架构；数据流体系架构；层次架构; 调用和返回架构；面向对象架构；

### **4.3.2数据设计(L)**

就酱：

构建高层抽象的数据模型、信息模型；概念数据模型？物理数据模型

### **4.3.3接口设计(L)**

### **4.3.4组件设计**

就酱：

也称为过程设计，位于数据设计、体系结构设计和接口设计完成之后。详细设计工具分为：.图形设计符号（流程图、盒图）、表格设计符号（决策表）、程序设计语言（PDL）

任何程序总可以用三种结构化的构成元素来设计实现：顺序，条件、重复

#### **4.3.4.1SD组件设计：流程图(H)**

就酱：

流程图的基本结构：顺序结构、选择结构，循环结构

#### **4.3.4.2OOD组件设计：顺序图(H)**

## **4.4其他**

问：其他几种流程图（盒图、PAD）需要重点掌握吗？

1. **质量保证**

## **5.1质量保证的概念(L)**

404：

含义：系统地监测和评估一个工程的各个方面，以最大限度地提高正在由生产过程中实现的质量的最低标准

原则：适合用途，该产品应符合预期的目的；

一次成功，错误应该被淘汰。

软件质量保证（SQA）：监控软件工程以确保软件质量的过程

## **5.2测试策略V模型**

### **5.2.1测试四级别(H)**

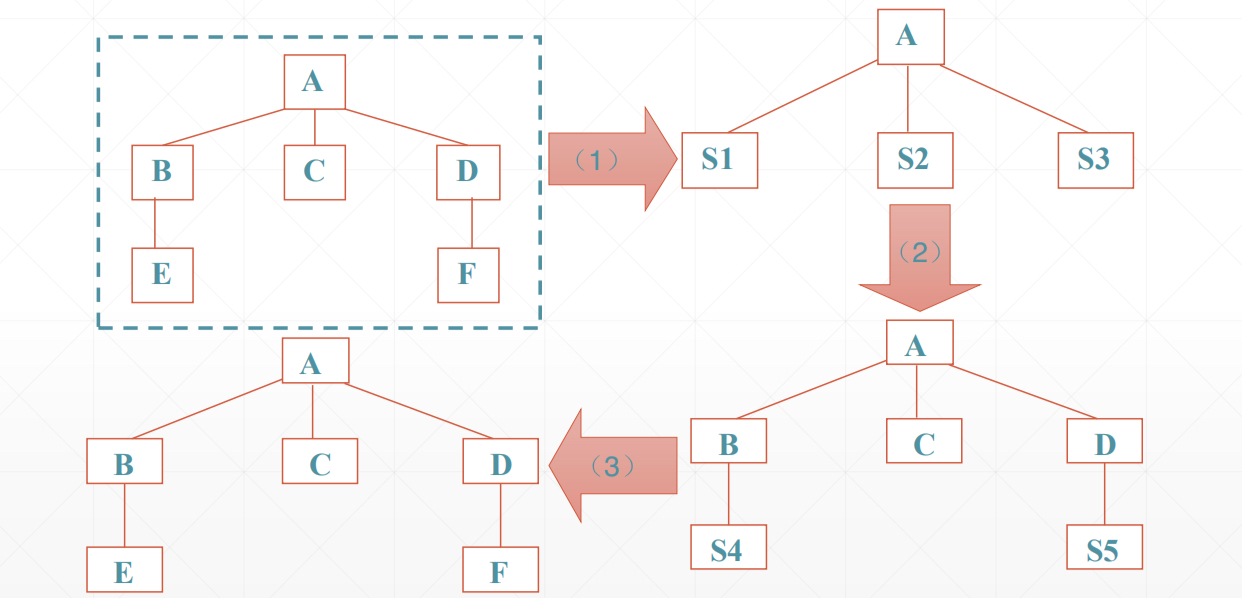
#### **5.2.1.1单元测试(M)**

#### **5.2.1.2集成测试(M)**

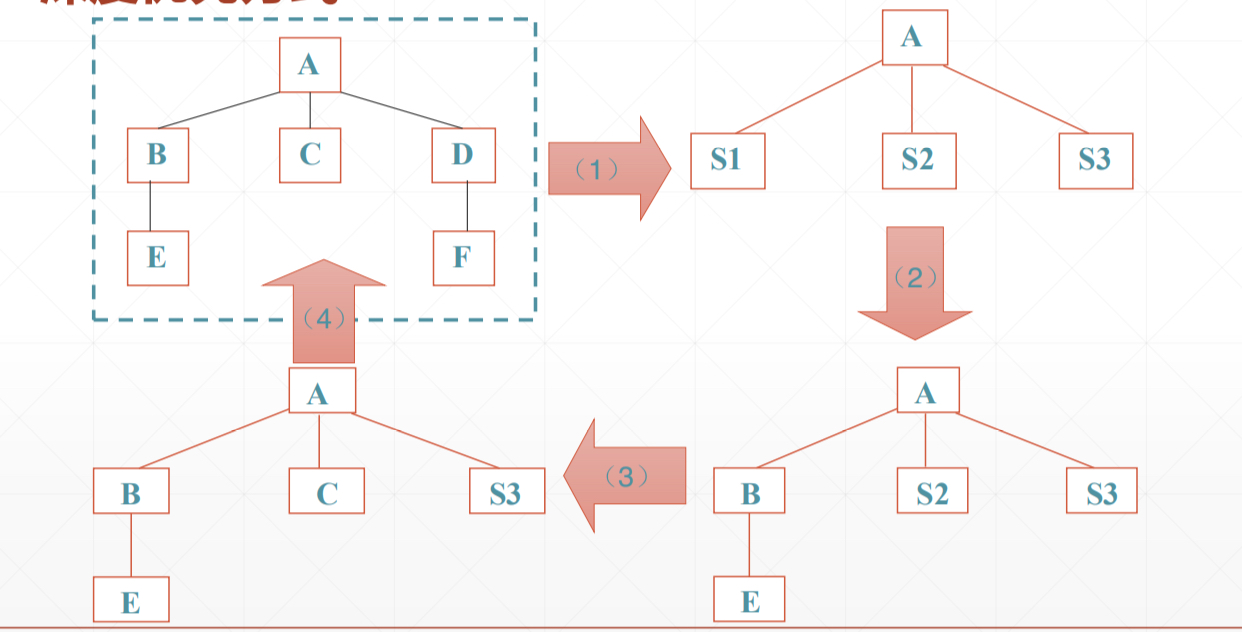
冂冂马：

1. 自顶向下集成：

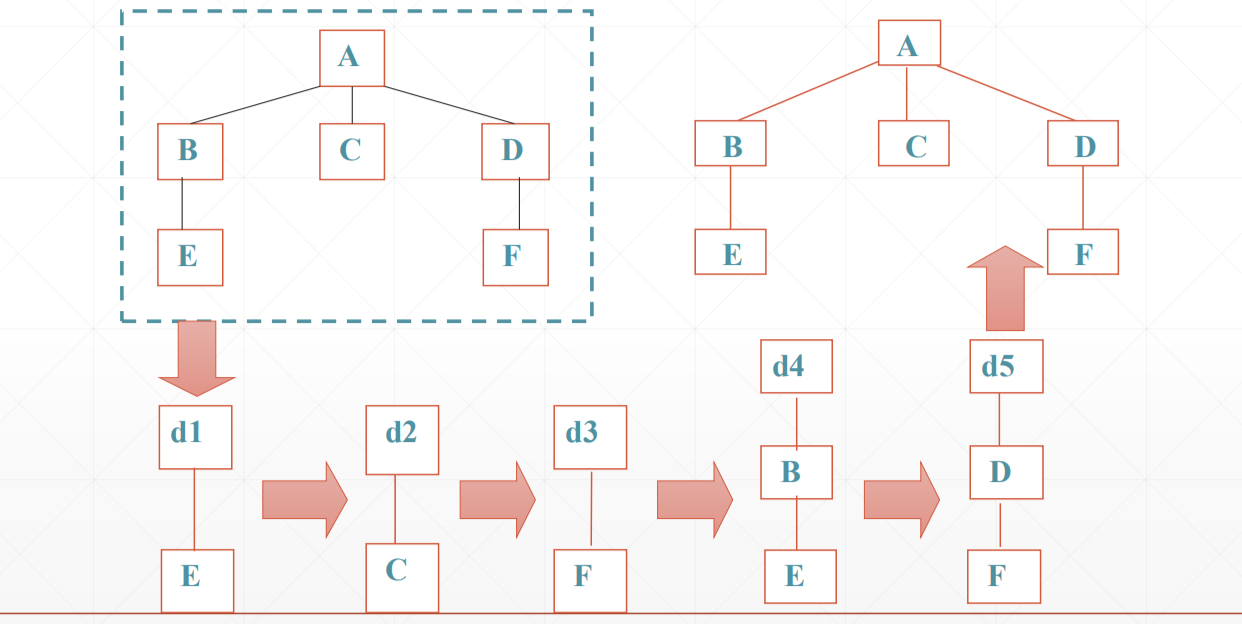
①广度优先



②深度优先



1. 自底向上集成



1. SMOKE方法

#### **5.2.1.3系统测试(M)**

冂冂马：

1. 功能测试
2. 性能测试
3. 强度/压力测试
4. 恢复测试
5. 安全性测试（安全性、保密性）

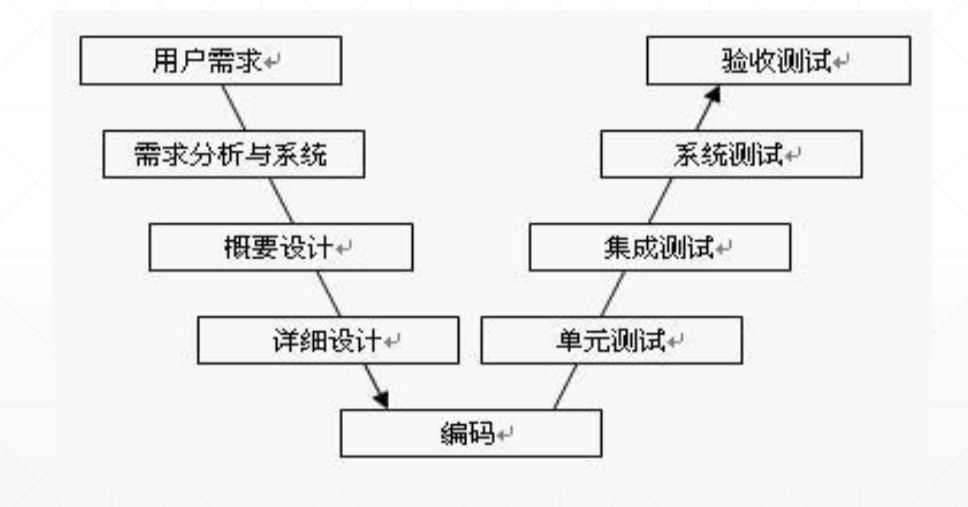
#### **5.2.1.4验收测试(M)**

冂冂马：以用户为主！

1. α测试：用户在开发环境下，在模拟实际操作环境下进行的测试。（开发者场所，开发者在场）
2. β测试：由用户在实际使用环境下进行的测试。（用户场所，开发者不在场）

### **5.2.2测试各级别与开发的各阶段对应关系(H)**

冂冂马：



## **5.3回归测试的概念(L)**

冂冂马：

回归测试范围：缺陷再测试、功能改变的测试、新功能测试、完全回归测试（测试整个系统）

## **5.4测试的常见术语**

### **5.4.1软件缺陷(H)**

冂冂马：

1. 软件未实现产品说明书要求的功能
2. 软件实现产品说明书未要求的功能
3. 软件出现了产品说明书指明不能出现的错误
4. 软件未实现产品说明书虽未明确提及但应该实现的目标
5. 软件难以理解，不易使用，运行缓慢，用户认为不好

### **5.4.2验证和确认(L)**

404

验证：保证软件特定开发阶段的输出已经正确完整地实现了规格说明

确认：对于每个测试级别，都要检查开发活动的输出是否满足具体的需求或与这些特定级别相关的需求。

### **5.4.3测试与质量保证(M)**

404

软件测试人员的目标是尽早找出软件缺陷，并确保缺陷得以修复

软件质量保证人员的主要职责是创建和执行改进软件开发过程并防止软件缺陷发生的标准和方法

### **5.4.4质量与可靠性(L)**

404

功能性、可靠性、可维护性、可用性、效率、可移植性

### **5.4.5调试与测试(L)**

404

两者都包含有处理软件缺陷和查看代码的过程

二者的区别在于：

测试的目标是发现软件缺陷的存在

调试的目标是定位与修复缺陷

### **5.4.6测试用例(H)**

404

测试用例是测试输入，执行条件，以及预期结果的集合，是为特定的目的开发的，例如执行特定的程序路径或验证与指定的需求相符合。

## **5.5测试技术**

### **5.5.1白盒测试**

冂冂马：

白盒测试：利用程序内部逻辑结构及有关信息，设计测试用例测试，也称结构性测试。

#### **5.5.1.1逻辑覆盖(H)**

冂冂马：

1. 语句覆盖：设计若干个测试用例，运行被测程序，使得每一可执行语句至少执行一次。
2. 分支覆盖：设计若干个测试用例，运行被测程序，使得程序中每个判断的取真分支和取假分支至少经历一次。也称为判定覆盖。
3. 条件覆盖：设计若干个测试用例，运行被测程序，使得程序中每个判断的每个条件的可能取值至少执行一次。
4. 条件组合覆盖：设计足够的测试用例，运行被测程序，使得每个判断的所有可能的条件取值组合至少执行一次。

#### **5.5.1.2控制流图覆盖(M)**

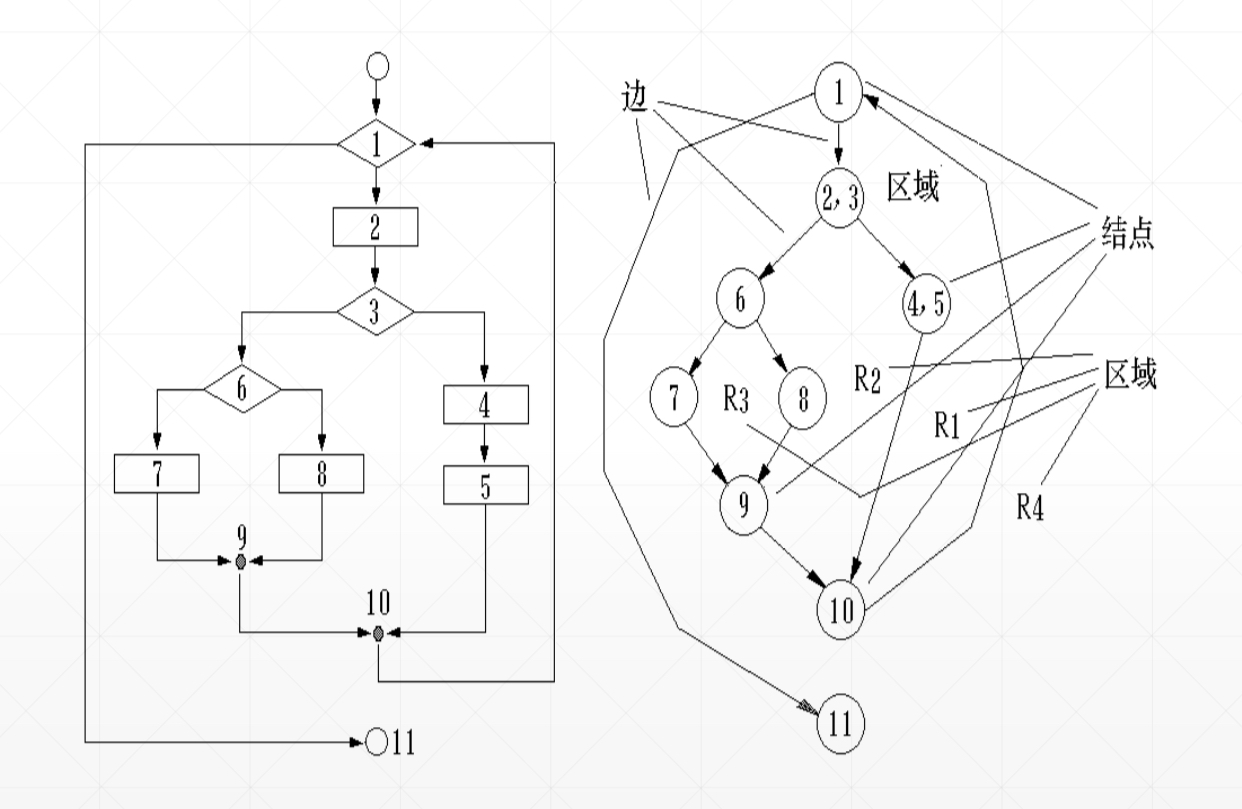
冂冂马：

1. 基本路径测试：

①基本路径测试方法把覆盖的路径数压缩到一定限度内，程序中的循环体最多只执行一次。

②它是在程序控制流图的基础上，分析控制构造的环路复杂性，导出基本可执行路径集合，设计测试用例的方法。设计出的测试用例要保证在测试中，程序的每一个可执行语句至少要执行一次。

③eg.



例如，在图示的控制流图中，一组独

立的路径是：

path1：1 - 11

path2：1 - 2 - 3 - 4 - 5 - 10 - 1 - 11

path3：1 - 2 - 3 - 6 - 8 - 9 - 10 - 1 - 11

path4：1 - 2 - 3 - 6 - 7 - 9 - 10 - 1 - 11

路径 path1，path2，path3，path4组成了控制流图的一个基本路径集。

### **5.5.2黑盒测试**

冂冂马：

黑盒测试：忽略程序内部机制，测试程序的每项功能是否符合要求，也称为功能性测试。

#### **5.5.2.1等价类划分(H)**

冂冂马：

1.等价类划分方法：

▪ 等价类划分是一种典型的黑盒测试方法，使用这一方法时，完全不考虑程序的内部结构，只依据程序的规格说明来设计测试用例。

▪ 等价类划分方法把所有可能的输入数据，即程序的输入域划分成若干部分，然后从每一部分中选取少数有代表性的数据做为测试用例。

▪ 使用这一方法设计测试用例要经历划分等价类（列出等价类表）和选取测试用例两步。

2.等价类的划分有两种不同的情况：

▪ 有效等价类：是指对于程序的规格说明来说，是合理的、有意义的输入数据构成的集合。

▪ 无效等价类：是指对于程序的规格说明来说，是不合理的、无意义的输入数据构成的集合

3.划分等价类的原则

(1) 如果输入条件规定了取值范围，或值的个数（范围），则可以确立一个有效等价类和两个无效等价类。

(2) 如果输入条件规定了输入值的集合，或者是规定了“必须如何”的条件，这时可确立一个有效等价类和一个无效等价类。

(3) 如果输入条件是一个布尔量，则可以确定一个有效等价类和一个无效等价类。

(4) 如果规定了输入数据的一组值，而且程序要对每个输入值分别进行处理。这时可为 每一个输入值确立一个有效等类,此外针对这组值确立一个无效等价类，它是所有不允许的输入值的集合。

(5) 如果规定了输入数据必须遵守的规则，则可以确立一个有效等价类（符合规则）和若干个无效等价类（从不同角度违反规则）。

#### **5.5.2.2边界值分析(M)**

### **5.5.3静态分析(L)**

## **5.6其他**

1. **软件维护**

## **6.1软件维护的基本概念(M)**

软件维护是指：由于软件产品出现问题

或需要改进，而对代码及相关文档进行

修改，其目的是对现有软件产品进行修

改的同时保持其完整性。

## **6.2软件维护的四个基本类型(H)**

165:

[软件维护](https://www.baidu.com/s?wd=%E8%BD%AF%E4%BB%B6%E7%BB%B4%E6%8A%A4&tn=SE_PcZhidaonwhc_ngpagmjz&rsv_dl=gh_pc_zhidao)活动类型大概有四种：纠错性维护（校正性维护）、适应性维护、完善性维护或增强、[预防性维护](https://www.baidu.com/s?wd=%E9%A2%84%E9%98%B2%E6%80%A7%E7%BB%B4%E6%8A%A4&tn=SE_PcZhidaonwhc_ngpagmjz&rsv_dl=gh_pc_zhidao)或再工程。

除此四类维护活动外，还有一些其它类型的维护活动，如：支援性维护（如用户的培训等）。

### **6.2.1纠错性维护(M)**

### **6.2.2适应性维护(M)**

### **6.2.3完善性维护(M)**

占比最大

### **6.2.4预防性维护(M)**

占比最小

## **6.3可维护性的决定因素(L)**

## **6.4软件维护过程模型(L)**

## **6.5软件再工程(L)**

## **6.6逆向工程(L)**

## **6.7其他**

1. **项目管理**

## **7.1项目管理四要素(H)**

蓝老师：人员、产品、项目、过程

## **7.2软件度量**

### **7.2.1度量**

#### **7.2.1.1基于规模(H)**

165:

p338

面向规模的软件度量是通过对质量和生产率的测量进行规范化而得到的，而这些测量都是根据开发过的软件的规模得到的。

面向规模的度量可以选择代码行（LOC）作为规范化值（需要注意的是LOC的度量不仅仅是代码的量多与少来评判，而是包含了错误数，缺陷数，成本和文档页数等但反对LOC来评判的人认为LOC是依赖于程序设计语言，说白了就是大牛们写的代码非常精简但效率极高，用LOC的度量来评判有点不合适）

#### **7.2.1.2基于功能点(M)**

### **7.2.2估算**

#### **7.2.2.1算法成本模型(L)**

#### **7.2.2.2COCOMO模型(M)**

## **7.3项目计划**

## **7.4其他**

|