文档介绍:

别管我这个命名，自己拿着安安静静复习就好
老师讲的考试内容过于模糊，故结合其他班老师+个人猜想总结如下
按我这个就行，不用看老师的 ☹

考试简介:

简答题（7*6'=42'）+ 综合设计题（4*n=58'）
卷面量较大;
内容涉及广度大;
好好看看老师上课所有的 PPT 习题;
面向对象概念少但分数多。。。

一 软件工程学概述

1.1 软件工程/软件定义

- 软件工程：指导计算机软件开发和维护的一门工程学科。采用工程的概念、原理、技术和方法来开发与维护软件，把经过时间考验而证明正确的管理技术和当前能够得到的最好的技术方法结合起来，以经济地开发出高质量的软件并有效地维护它，这就是软件工程。
- 软件（Software）是一系列按照特定顺序组织的计算机数据和指令的集合。
  程序+文档 两部分组成
- 产生原因：软件危机

1.2 软件危机定义 表现 原因 解决

定义：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

- 如何开发软件，满足需求
- 如何维护诸多的软件

具体表现:

- 对软件开发成本和进度的估计不准确
- 对已完成的软件不满意
- 软件质量不可靠
- 软件不可维护
- 没有文档资料
- 软件成本比上升
- 软件开发效率跟不上计算机的普及

原因:

- （客观）软件本身的特点：逻辑部件、 规模日益复杂庞大
- （主观）软件开发与维护：软件开发管理复杂 开发技术落后 生产方式落后 开发工具落后 软件费用日益增加 程序编写不规范 轻视软件维护

解决:

- 正确认识软件
- 科学的组织管理措施

- 有效的方法和技术
- 工具和软件工程支撑环境

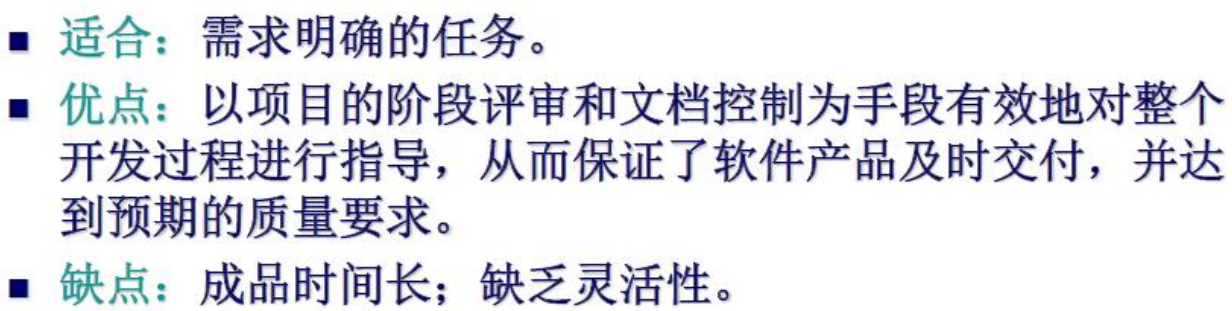
1.3 软件工程生命周期及各阶段工作

软件定义 + 软件开发 + 软件维护

- 问题定义：确定要解决的问题
- 可行性研究：用最小代价在尽可能短的时间内确定问题能否解决
- 需求分析：确定系统必须完成的工作，提出完整、准确、清晰、具体的要求
- 概要设计：系统设计，确定系统的具体实现方案；结构设计，确定软件结构
- 详细设计：确定应该怎样具体实现所要求的系统，得出对目标系统的精确描述
- 编码和单元测试：选定平台，将详细设计的结果翻译并编程，并测试每个模块
- 综合测试：通过各种类型的测试使软件达到预定的要求
- 软件维护：通过必要的维护活动使系统持久地满足用户需求

1.4 六大软件开发模型

- 瀑布模型



- 增量模型
- 快速原型模型
- 螺旋模型
- 喷泉模型
- Rational 统一过程

二 可行性分析研究

2.1 数据流图(data flow)概念

数据流图描绘系统的逻辑模型，图中没有物理元素，只是描绘信息在系统中流动和处理情况。该图只需考虑系统必须完成的基本逻辑功能，不需要考虑如何具体实现。

画数据流图的基本目的是利用它作为交流信息的工具，数据流图的另一个主要用途是作为分析和设计的工具。

2.2 画数据流图

见例题

2.3 数据字典(（DD，data Dictionary）)的定义和任务

数据字典是关于数据的信息的集合，也就是对数据流图中包含的所有元素的定义的集合。数据流图和数据

字典公共构成系统的逻辑模型。
数据字典由对下列四类元素的定义组成：
（1）数据流
（2）数据流分量（即：数据元素）
（3）数据存储
（4）处理
任务：
是对于数据流图中出现的所有命名元素，包括数据流，加工，数据文件，以及数据的源，终点等，在数据字典中作为一个词条加以定义，假设的每一个图形元素的名字都有一个确切的解释。
图形元素的名字：某一词条的名字，要求无二义性，为人们所公认；
- 别名或编号；
- 分类：加工，数据流，数据文件，数据元素，数据源，汇点等；
- 描述：功能，特点；
- 定义：该词条的组成，数据结构等；
- 位置：数据流的来源，去处，加工框的编号，输入输出；数据元素在哪个数据结构中等。
- 其他：数据流的数据量，流通量；数据文件的存储方式，存取要求；数据加工的加工顺序，以及外部实体的数量等。

三 需求分析

3.1 需求分析的工具
数据流图 数据字典 系统结构图 层次方框图

3.2 数据流图
见例题

3.3 需求分析过程（三个方面）
获取和理解需求：获取并理解用户需求，清除用户需求的不一致性，模糊性和歧义性，帮助用户发现潜在的需求
描述和分析需求：对用户需求进行建模，生成 SRS 和初步用户手册
评审用户需求：多方人员一起对 SRS 进行复核和评审，以确保用户手册和 SRS 全面、准确、一致地反映用户需求

需求分析是软件定义时期的最后一个阶段，它的基本任务不是确定系统怎样完成它的工作，而是确定系统必须完成哪些工作，也就是对目标系统提出完整、准确、清晰、具体的要求。并在需求分析阶段结束之前，由系统分析员写出软件需求规格说明书，以书面形式准确地描述软件需求。

第三章 需求分析与项目管理

典型的符合国家标准GB856D-88规定的规格说明格式

1、引言
- 1、1 编写说明
- 1、2 背景
- 1、3 定义
- 1、4 参考资料

2、任务概述
- 2、1 目标
- 2、2 用户的特点
- 2、3 假定与约束

3、需求规定
- 3、1 对功能的规定
- 3、2 对性能的规定

16

第三章 需求分析与项目管理

- 3、2、1 精度
- 3、2、2 时间特性要求
- 3、2、3 灵活性
- 3、3 输入输出要求
- 3、4 数据管理能力要求
- 3、5 故障处理要求
- 3、6 其他专门要求

4、运行环境规定
- 4、1 设备
- 4、2 支持软件
- 4、3 接口
- 4。4 控制

四 形式化说明技术（貌似不重要）

4.1 有穷自动机 Petri 网 Z 语言的定义及其使用

五 总体设计

模块化：就是把程序划分成独立命名且可独立访问的模块，每个模块完成一个子功能，把这些模块集成起来构成一个整体，可以完成指定的功能满足用户的需求。
抽象：现实世界中一定事物、状态或过程之间总存在着某些相似的方面(共性)。把这些相似的方面集中和概括起来，暂时忽略它们之间的差异，这就是抽象。
逐步求精：为了能集中精力解决主要问题而尽量推迟对问题细节的考虑。逐步求精是人类解决复杂问题时采用的基本方法，也是许多软件工程技术的基础。
信息隐藏：应该这样设计和确定模块，使得一个模块内包含的信息(过程和数据)对于不需要这些信息的模块来说，是不能访问的。
局部化：局部化的概念和信息隐藏概念是密切相关的。所谓局部化是指把一些关系密切的软件元素物理地放得彼此靠近。显然，局部化有助于实现信息隐藏。
模块独立：
模块独立的概念是模块化、抽象、信息隐藏和局部化概念的直接结果。

5.1 六大设计原则

模块化 抽象 逐步求精 信息隐藏和局部化 模块独立

5.2 两种设计方法（结构化设计 面向对象设计）

阶段	结构化分析法	面向对象分析法
可行性研究	数据流图、数据字典	数据流图、数据字典
需求分析	细化数据流图、数据字典	用例图、类图、对象图
设计阶段	数据流图 → 软件结构：层次图、HIPO图、结构图 过程设计：PAD图、盒图、判定树、判定表、伪代码 面向数据结构设计：Jackson图、Jackson方法	类图的细化 状态图、活动图、顺序图或协作图

教改班讲的三种程序设计方法：

- 面向对象
- 面向数据流
- 面向程序结构

5.3 信息隐藏 局部化

信息隐藏：应该这样设计和确定模块，使得一个模块内包含的信息(过程和数据)对于不需要这些信息的模块来说，是不能访问的。
局部化：局部化的概念和信息隐藏概念是密切相关的。所谓局部化是指把一些关系密切的软件元素物理地放得彼此靠近。显然，局部化有助于实现信息隐藏。

5.4 耦合 内聚

耦合：是对一个软件结构内不同模块之间互连程度的度量。

完全独立、数据耦合 控制耦合 特征耦合 公共环境耦合 内容耦合

内聚：标志一个模块内各个元素彼此结合的紧密程度，它是信息隐藏和局部化概念的自然扩展。
偶然内聚 逻辑内聚 时间内聚 过程内聚 通信内聚 顺序内聚 功能内聚

尽量使用数据耦合， 少用控制耦合和特征耦合，限制公共环境耦合的范围， 完全不用内容耦合。
设计时力争做到高内聚，低耦合，并且能够辨认出低内聚的模块。

5.5 面向数据流的方法（分解）
见例题

六 详细设计

6.1 结构化设计定义

- 经典定义：如果一个程序的代码块仅仅通过顺序、选择和循环这 3 种基本控制结构进行连接，并且每个代码块只有一个入口和一个出口，则称这个程序是结构化的。
- 更全面定义：结构程序设计是尽可能少用 GOTO 语句的程序设计方法。最好仅在检测出错误时才使用 GOTO 语句，而且应该总是使用前向 GOTO 语句。

6.2 程序伪码 程序结构图 盒图 PAD 图 的转化（画图需要📏）

- 伪码->盒图->PAD 图
- 程序结构图->PAD 图

6.3 McCabe 方法判断复杂度（三种方法）
环形复杂度定量度量程序的逻辑复杂度。
计算环形复杂度 V(G):

- V(G)=流图中的区域数
- V(G)=E-N+2 其中 E 是流图中的边数，N 是结点数
- V(G)=P+1 其中 P 是流图中判定结点的总数目

（判定结点=多分支的数目-1，结点上的 2 个输出为 1，3 个为 2…）

6.4 三种最基本的控制结构
顺序 选择 重复

七 实现(编码和测试)

7.1 编码的任务
编码就是把软件设计结果翻译成用某种程序设计语言书写的程序。

7.2 测试方法
黑盒测试(功能测试):

- 把程序看作一个黑盒子;
- 完全不考虑程序的内部结构和处理过程;

- 是在程序接口进行的测试。

白盒测试(结构测试):

- 把程序看成装在一个透明的盒子里;
- 测试者完全知道程序的结构和处理算法;
- 按照程序内部的逻辑测试程序，检测程序中的主要执行通路是否都能按预定要求正确工作。

集成测试:

综合测试:

7.3 白盒测试 五种方法

- 语句覆盖：使被测程序中每个语句至少执行一次。
- 判定覆盖：不仅每个语句必须至少执行一次，而且每个判定的每种可能的结果都应该至少执行一次。
- 条件覆盖：不仅每个语句至少执行一次，而且使判定表达式中的每个条件都取到各种可能的结果。
- 判定/条件覆盖:使判定表达式的每个条件都取到各种可能值,每个判定表达式都取到各种可能的结果。
- 条件组合覆盖:选取足够多的测试数据,使得每个判定表达式中条件的各种可能组合都至少出现一次。

7.4 黑盒测试（重点等价划分）

见例题

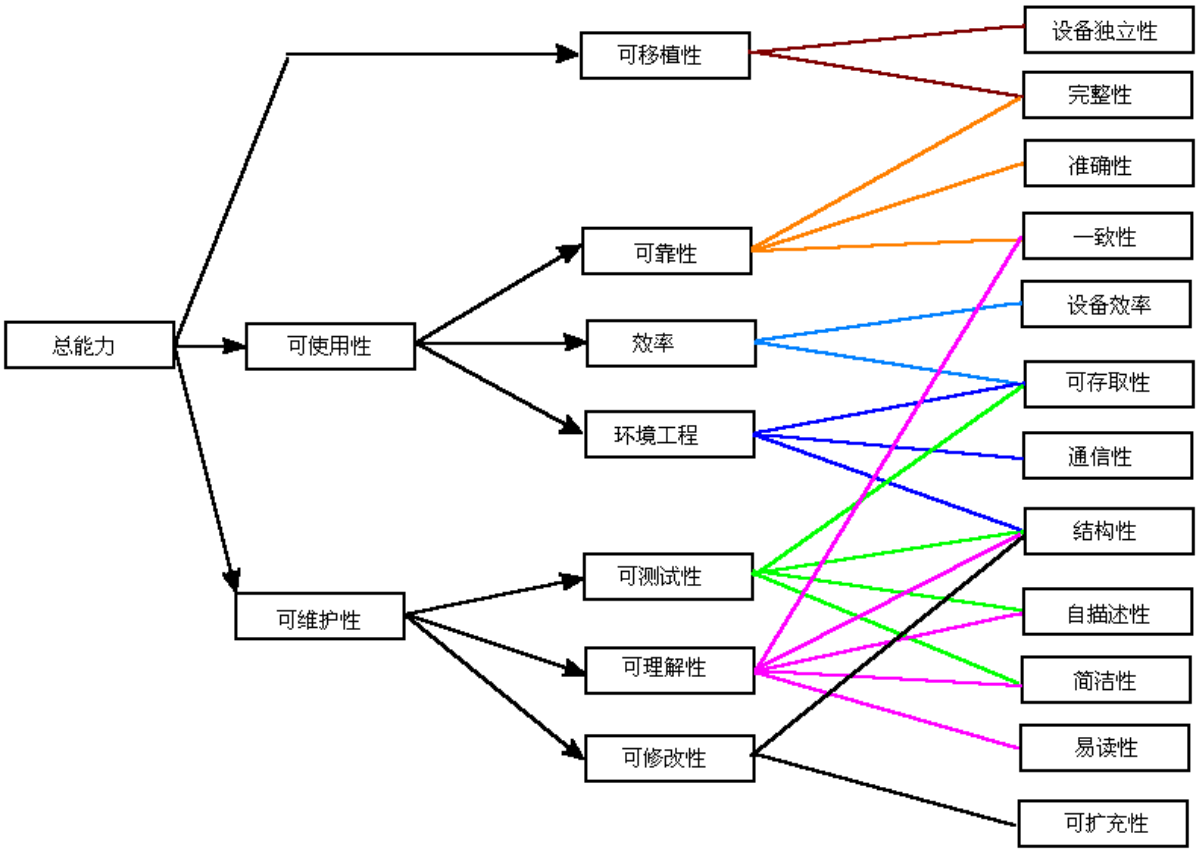
模块测试 子系统测试 系统测试 验收测试 平行运行

八 维护

8.1 维护的定义 作用

定义：所谓软件维护就是在软件已经交付使用之后，为了改正错误或满足新的需要而修改软件的过程。

- 改正性维护：把诊断和改正错误的过程
- 适应性维护：为了和变化了的环境适当地配合而进行的修改软件的活动
- 完善性维护：为了满足在用户提出的增加新功能或修改已有功能的要求和一般性的改进要求
- 预防性维护：采用先进的软件工程方法对需要维护的软件或软件中的某一部分，主动性维护。



8.2 软件再工程过程 反推 再生

8.3 冗余 软件/硬件冗余

- 硬件冗余：增加一些硬件设备，使系统具有某些硬件作为备份，出现故障时，系统仍能工作，无故障时这些设备是多余的，但可以提高系统的可靠性。
- 软件冗余：增加一些程序，可以检查和避免错误，还可以消除由于错误的程序而造成的影响。

8.4 软件文档(GB 13 种)作用：
提高软件开发过程的能见度、提高开发效率.作为开发人员工作成果和结束标志、记录开发过程的有关信息便于使用和维护、提供软件运行、维护和培训的资料、便于用户了解软件功能和性能

九 面向对象

9.1 用例图 类图
见例题

9.2 面向对象优点
与人类习惯思维方法一致
稳定性好
可重用性
较易开发较大型的软件
可维护性好

9.3 面向对象原则
1. 模块化：对象就是模块。它是把数据结构和操作这些数据的方法紧密地结合在一起所构成的模块。
2. 抽象：面向对象方法不仅支持过程抽象，而且支持数据抽象。
3. 信息隐藏：在面向对象方法中，信息隐藏通过对象的封装性实现。
4. 弱耦合：耦合指不同对象之间相互关联的紧密程度。
5. 强内聚：内聚衡量一个模块内各个元素彼此结合的紧密程度
6. 可重用：软件重用是提高软件开发生产率和目标系统质量的重要途径。

9.4 活动图（带泳道）时序图 协作图
- 基本概念
- 习题

9.5 UML 中建立的模型（9 个->13 个）
对需求建模
- 用例图：从用户角度描述系统的行为
- 类图：显示一组类、接口、协作及它们之间的关系
- 对象图：显示一组对象及它们之间的关系

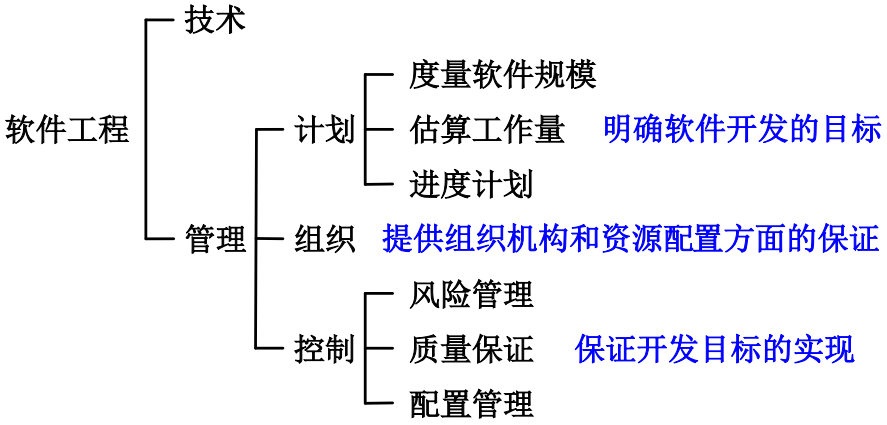
对结构建模
- 组件图：显示一组组件之间的组织和依赖关系
- 部署图：显示对运行时处理节点以及其中的组件的配置
- 类图、对象图

对行为建模
- 顺序图：注重于消息的时间次序
- 协作图：注重于收发消息的对象的结构组织
- 状态图：注重于由事件驱动的对象状态变化序列
- 活动图：注重于从活动到活动的控制流程
-

十 软件项目管理

10.1 软件配置管理内容
管理：就是通过计划、组织和控制等一系列活动，合理地配置和使用各种资源，以达到既定目标的过程。

软件项目管理过程：从一组项目计划活动开始，而制定计划的基础是工作量估算和完成期限估算。



10.2 目标 配置项 过程

- 目标：使变化更正确且更容易被适应，在必须变化时减少所需花费的工作量
- 软件配置项：程序、文档、数据
- 过程：标识配置对象，版本控制，变化控制，配置审计，状态报告

IEEE 把基线定义为：已经通过了正式复审的规格说明或中间产品，它可以作为进一步开发的基础，并且只有通过正式的变化控制过程才能改变它。

简而言之，基线就是通过了**正式复审的软件配置项。**