软件工程

by xd_zhu

笔记

零、考试

- 时间

> 7.11 9:00-11:00

- 题型

> 简答题5道（5*6）
> 分析题4道（4*10）
> 综合分析题（30）

- 考点

> ① 程序的含义、软件危机 → 软件工程、软件工程方法学、软件生命周期
> ② 面向对象方法：用例图、类图、对象图
> ③ 结构化方法：数据流图、结构图、数据流图 → 软件结构模型、程序流程图 → PAD图与盒图
> ④ 黑盒测试等价类划分、白盒测试路径概念
> ⑤ 软件工程完整流程：面向对象方法和结构化方法
> ⑥ 维护种类、软件质量保证、软件配置管理、能力成熟度模型

一、概述

- 软件

> 含义：程序、数据结构、文档

- 软件工程方法学

> 含义：方法、工具、过程
> 主要方法：结构化方法、面向对象方法

- 软件危机

> 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
> ① 如何开发软件，满足需求
> ② 如何维护诸多的软件
>
> 具体表现：
> ① 对软件开发成本和进度的估计不准确
> ② 对已完成的软件不满意
> ③ 软件质量不可靠
> ④ 软件不可维护
> ⑤ 没有文档资料
> ⑥ 软件成本比上升
> ⑦ 软件开发效率跟不上计算机的普及
>
> 原因：
> ①（客观）软件本身的特点：逻辑部件、规模庞大
> ②（主观）软件开发与维护的方法：忽视需求分析、程序编写不规范、轻视软件维护
>
> 解决：
> ① 正确认识软件
> ② 科学的组织管理措施
> ③ 有效的方法和技术
> ④ 工具和软件工程支撑环境

- 软件工程

> 定义：制导计算机软件开发和维护的一门工程学科
> 本质特征：大型程序、控制复杂性、软件常变、效率、合作、支持用户、文化背景
> 目标：低成本、及时、正确、有效、可靠、可维护、可重用、可追踪、可移植
> 主要问题：费用、可靠性、维护、生产率、重用、安全
> 基本原理：分阶段、阶段评审、产品控制、现代程序设计技术、结果可审查、人员少而精、不断总结改进

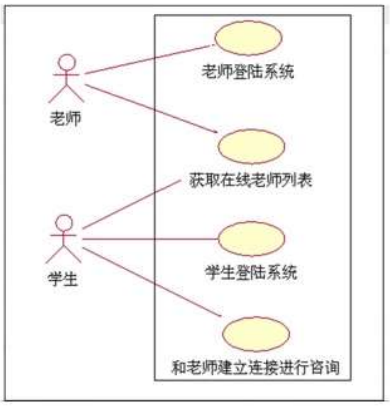
- 软件生命周期

> 软件定义①②③ + 软件开发④⑤⑥⑦ + 软件维护⑧
> ① 问题定义：确定要解决的问题
> ② 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
> ③ 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
> ④ 总体设计：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
> ⑤ 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
> ⑥ 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
> ⑦ 综合测试：通过各种类型的测试使软件达到预定的要求
> ⑧ 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

二、各种图

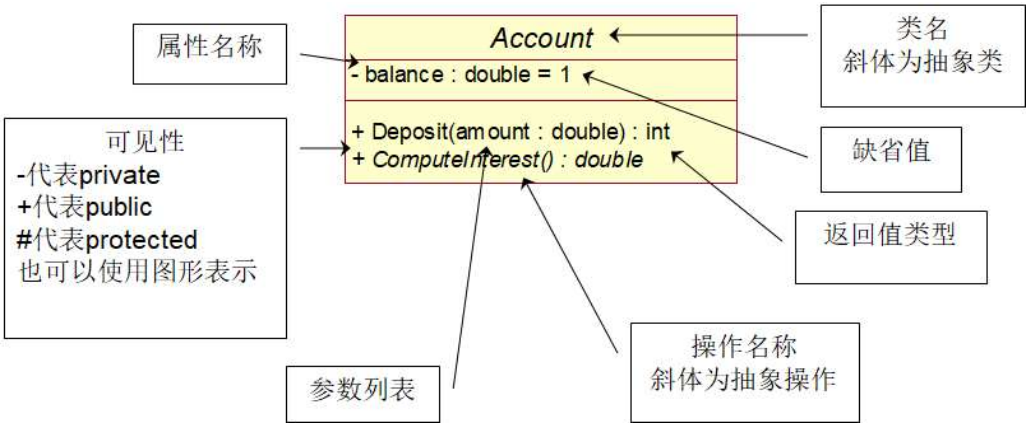
- 数据流图

- 用例图

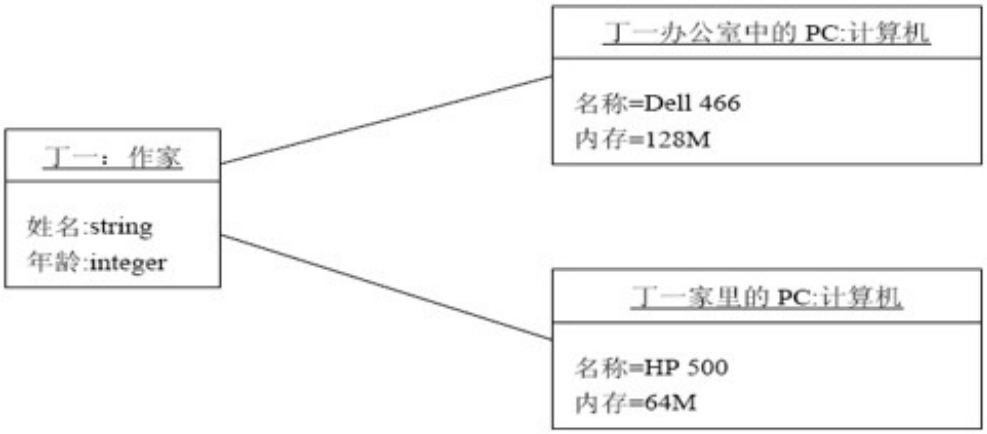


关系	功能	符号
关联	执行者与其参与的用例之间的通信路径	———
包含	在基用例上插入附加的行为，并且显示地描述了该插入	<<include>> --------->
扩展	在基用例上插入附加的行为，基用例并不知道	<<extend>> --------->
泛化	一般用例和特殊用例之间的关系 其中特殊用例继承了一般用例的特征并增加了新的特征	———▷

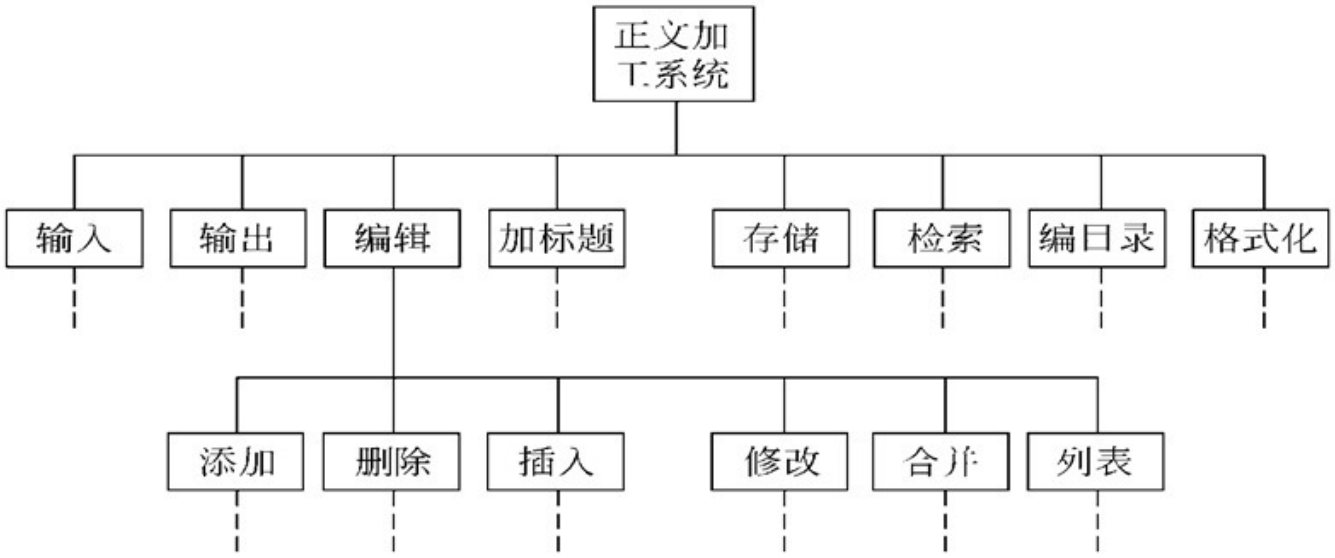
- 类图



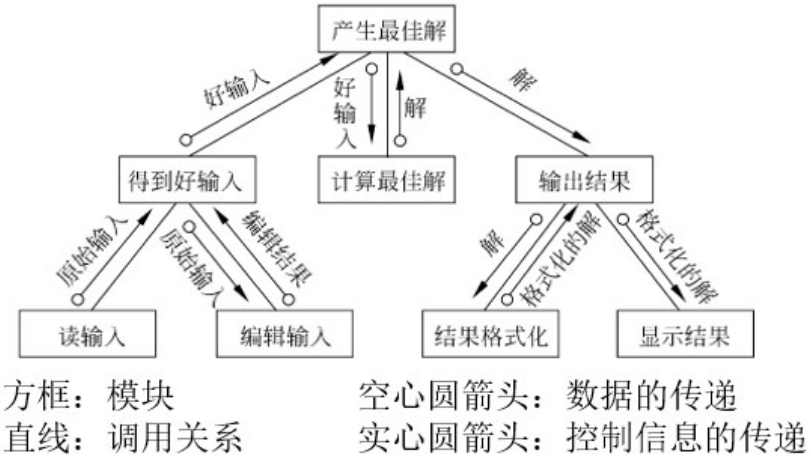
- 对象图（类图的一个实例）



- 层次图

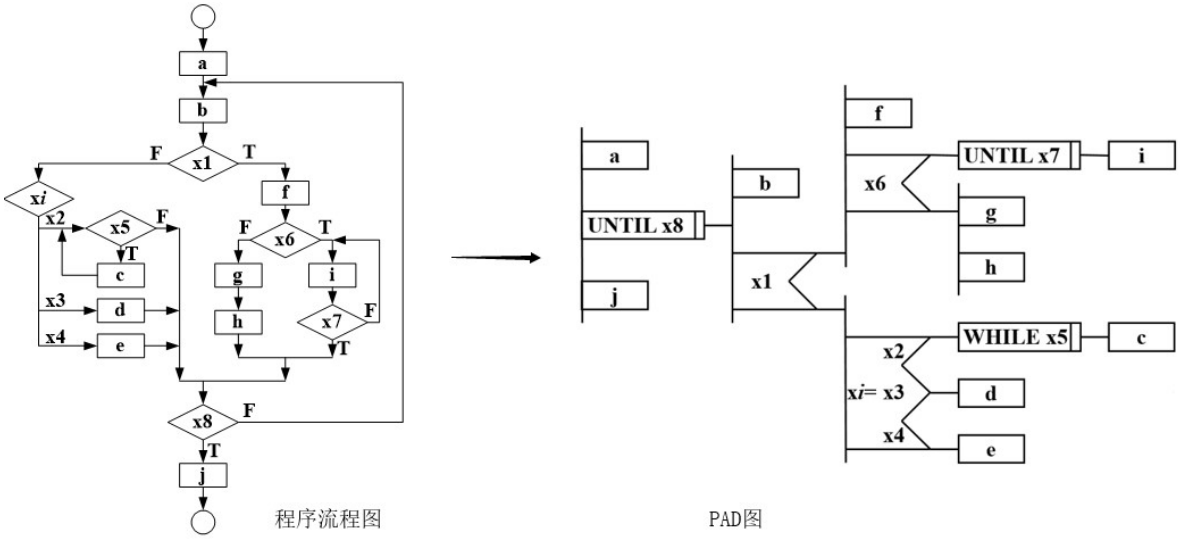


- 结构图



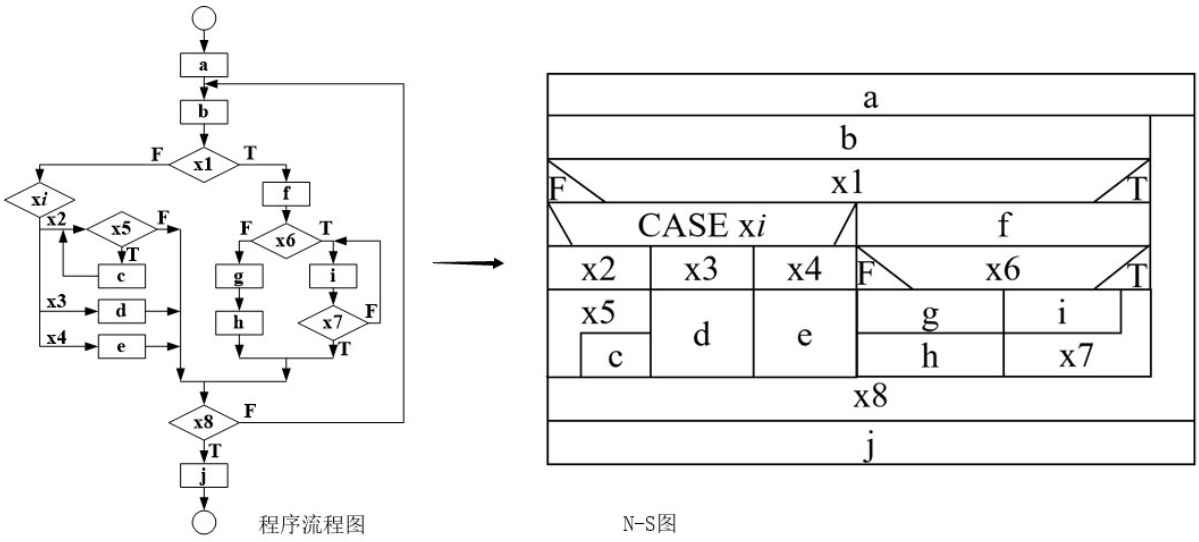
方框：模块　　空心圆箭头：数据的传递
直线：调用关系　　实心圆箭头：控制信息的传递

- PAD图



程序流程图　　PAD图

- 盒图



程序流程图　　N-S图

- ~~状态图、活动图、顺序图、协作图~~（懂）
- ~~Jackson图、HIPO图、判定树、判定表~~（懂）

三、可行性研究

- 包括

> 技术可行性、经济可行性、操作可行性、法律可行性

四、需求分析

- 用户需求 → 系统需求

> ① 功能需求
> ② 非功能需求：性能需求、可靠性和可用性需求、安全性需求、隐私需求、用户界面需求等
> ③ 领域需求

- 过程

> ① 获取和理解需求
> ② 描述和分析需求 → 需求规格说明书
> ③ 评审用户需求 → 基线

五、总体设计

- 耦合

> 概念：对一个软件结构内不同模块之间互连程度的度量
> 分类：完全独立、数据耦合、控制耦合、特征耦合、公共环境耦、内容耦合

- 内聚

> 概念：标志一个模块内各个元素彼此结合的紧密程度
> 分类：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

- 数据流图 → 软件结构

> 区分变化流、事物流
>
> 启发规则：
> ① 改进软件结构提高模块独立性
> ② 模块规模应该适中
> ③ 深度、宽度、扇出和扇入都应适当
> ④ 模块的作用域应在控制域之内
> ⑤ 力争降低模块接口的复杂程度
> ⑥ 设计单入口单出口的模块
> ⑦ 模块功能应该可以预测

六、详细设计

- 程序复杂度

> 程序流程图 $\xrightarrow{\text{退化}}$ 流图
> 复杂度 = 区域数 = 边数-节点数+2 = 判定节点数+1

七、结构化分析法、面向对象分析法

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

八、编码、单元测试、综合测试

- 概念

> Alpha测试：开发环境下，开发者对用户的指导下
> Beta测试：实际使用环境下，开发者不在现场
> 可靠性：时间段内，成功运行的概率
> 可用性：时间点上，成功运行的概率

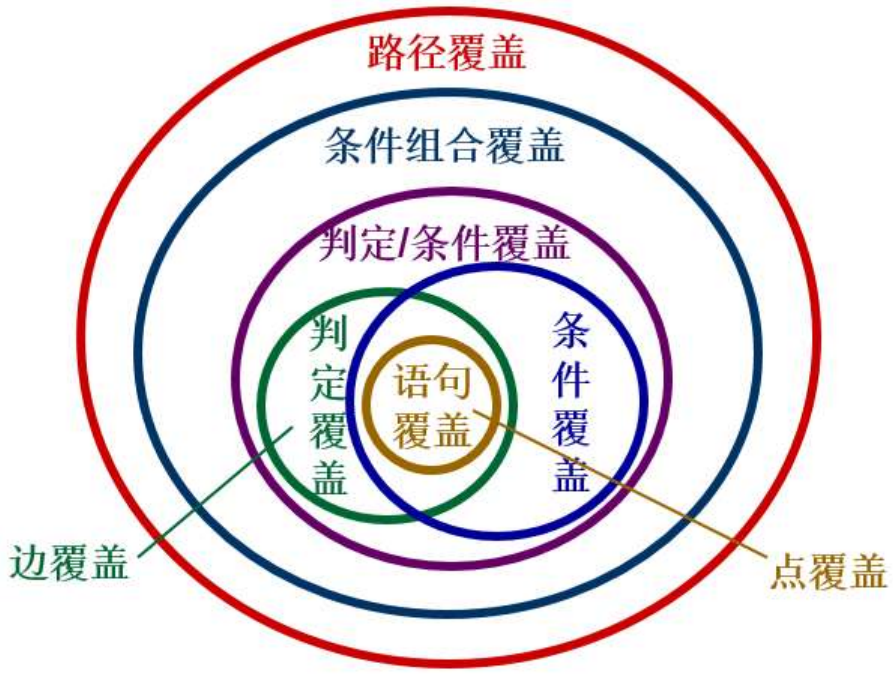
- **测试**

> 方法：黑盒测试（功能）、白盒测试（结构）
> 步骤：单元测试、子系统测试、系统测试、确认测试、平行运行

测试阶段	主要依据	测试人员	测试方式	主要测试内容
单元测试	系统设计文档	开发小组	白盒测试	接口测试 路径测试
子系统测试	系统设计文档 需求文档	独立测试小组	白盒测试 黑盒测试	接口测试 路径测试 功能测试 性能测试
系统测试	需求文档	独立测试小组	黑盒测试	功能测试 健壮性测试 性能测试 用户界面测试 安全性测试 压力测试 可靠性测试 安装/反安装测试
确认测试 验收测试	需求文档	用户	黑盒测试	

- 白盒测试

> ① 语句覆盖（点覆盖）：每个语句至少执行一次
> ② 判定覆盖（边覆盖）：语句覆盖+每个判定的每种结果至少执行一次
> ③ 条件覆盖：语句覆盖+每个判定表达式的每个结果至少取到一次
> ④ 判定条件覆盖：语句覆盖+判定覆盖+条件覆盖
> ⑤ 条件组合覆盖：每个判定框中的判定条件的每种组合至少取到一次
> ⑥ 路径覆盖：每条路径至少执行一次



- 黑盒测试：等价类划分

  - （1）划分等价类。
  - （2）设计新的测试方案以尽可能多地覆盖尚未被覆盖的有效等价类，重复这一步骤直到所有有效等价类都被覆盖为止。
  - （3）设计新的测试方案，使它覆盖一个而且只覆盖一个尚未被覆盖的无效等价类，重复这一步骤直到所有无效等价类都被覆盖为止。

九、维护

- 分类

> ① 改正性维护：诊断和改正错误的过程
> ② 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、
> ③ 完善性维护：为了满足增加新功能或修改已有功能的要求和一般性的改进要求
> ④ 预防性维护：主动性维护

十、管理

- 软件质量保证（SQA）

> 软件质量：软件与明确地和隐含地定义的需求相一致的程度
> ① 复审或评审
> ② 软件测试
> ③ 程序正确性证明

- 软件配置管理

> 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：①标识配置对象，②版本控制，③变化控制，④配置审计，⑤状态报告

- 能力成熟度模型（CMM）

> ① 初始级：无序、混乱。
> ② 可重复级：建立了基本的项目管理过程，可跟踪成本、进度、功能和质量
> ③ 已定义级：定义了完整的软件过程，软件过程已经文档化和标准化
> ④ 已管理级：所有项目的重要的过程活动都是可度量的
> ⑤ 优化级：软件过程是可优化的