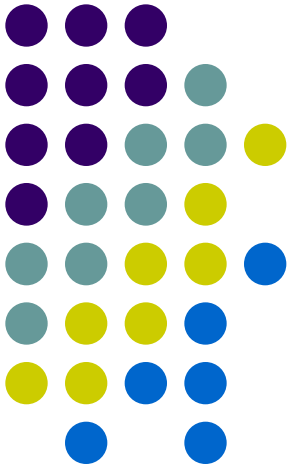
软件工程

Software Engineering

主讲人：张敏灵

Email: zhangml@seu.edu.cn

URL: http://cse.seu.edu.cn/people/zhangml/



Part I: 软件工程导论

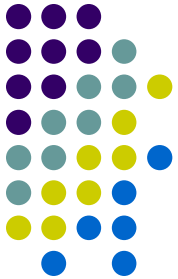
小结

- 为什么要研究软件工程(Why need SE?)
  - 软件危机
    - 计算机软件的**开发**和**维护**过程中遇到的一系列问题
  - 软件危机的具体表现
    - 成本过高、进度过长、软件质量差、文档缺失……
  - 软件危机产生的原因
    - 自身特点：不同于硬件、不同于一般程序
    - 软件开发和维护方法不正确：“重”编程、“轻”分析、维护

小结

- 什么是软件工程(What is SE?)
  - 指导计算机软件开发和维护的一门工程学科
    - 包括技术和管理两个方面
  - 本质特征/核心课题
    - 大规模、高复杂性、高动态性、效率要求高、关注用户需求、关注人员协作、克服背景差异
  - Boehm软件工程七原理
    - 原理“独立性”：任意一条原理不可被其余原理替代
    - 原理“完备性”：涵盖了绝大多数已有的原理或准则

小结



- 怎么进行有效的软件工程实践(How to SE?)
  - **软件工程方法学**：软件生命周期全过程中使用的一整套方案的集合
    - 涉及**软件过程**、**软件方法**、**软件工具**三个要素
  - 常用软件工程方法学
    - 传统方法学：“自顶向下、逐步求精”的方式完成软件开发的各阶段任务
    - 面向对象方法学：基于面向对象技术，保持问题域与求解域一致
  - 软件生命周期
    - 软件定义：问题定义，可行性研究，需求分析
    - 软件开发：总体设计、详细设计、编码实现、软件测试
    - 软件维护：改正性、适应性、完善性、预防性维护

小结

- 软件过程
  - 软件工程方法学**核心**
    - 没有一种软件过程适用所有软件项目
    - 应针对项目特点科学、有效地定义软件过程
  - 软件过程模型
    - 瀑布模型：需求分析定义清晰，顺序实现软件生命周期各阶段
    - 原型模型：用户的具体需求在开发初期难以清楚地界定
    - 快速开发模型：基于“构件(component)”的开发方法缩短开发周期
    - 螺旋模型：结合原型模型迭代特性与瀑布模型系统化特性
    - 其它模型

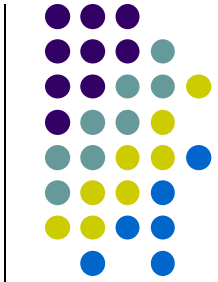
小结

- 问题定义
  - 3W：Who, What, Why
  - 问题定义报告
- 可行性研究
  - 目的，基本任务
    - 技术可行性，经济可行性，操作可行性，社会可行性
  - 研究过程
    - 复查系统规模目标➔刻画现有系统➔导出新系统逻辑模型➔进一步定义问题(可回溯)➔导出评价可选方案➔推荐行动方针➔撰写可行性报告

小结

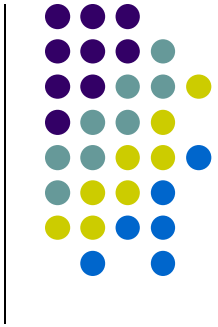
- 系统流程图
  - **物理数据流图**➔数据在系统各部件之间的流动情况
  - 基本符号+系统符号 (I/O+Processing+Flow)
- 数据流图 (DFD)
  - **逻辑数据流图**➔数据在软件中流动和被处理的逻辑过程
  - 基本符号
    - 数据源点/终点，数据处理，静态数据，动态数据
  - 附加符号
    - 逻辑“与”，逻辑“或”，逻辑“异或”
  - 命名规则

小结



- 数据字典 (DD)
  - 内容：静/动态数据，数据元素，数据处理，其它信息
  - 定义：“自顶向下"逐步分解
  - 用途与实现方式
- 成本-效益分析
  - 成本估计
    - 代码行技术，任务分解技术，自动估计成本技术
  - 分析方法
    - 货币的时间价值，投资回收期，纯收入，投资回收率

小结



- 需求分析
  - 目的：回答“系统必须做什么”，撰写**软件需求规格说明书(SRS)**
  - 准则
    - 理解并描述问题的信息域➔建立**数据模型**
    - 描述作为外部事件结果的软件行为➔建立**行为模型**
    - 定义软件应完成的功能➔建立**功能模型**
    - 对数据、行为及功能模型进行分解，分层次展示细节
  - 任务
    - 确定系统综合要求，分析系统数据要求
    - 导出系统逻辑模型
    - 修正系统开发计划

小结

- 用户需求获取
  - **访谈**：正式访谈/非正式访谈
  - **面向数据流求精**：“自顶向下” 细化数据流图
  - **简易应用规格说明**：摆脱用户 “被动”地位，与开发者密切合作
  - **快速建立软件原型**：尽快提供可运行且易修改的系统模型
- 实体-联系(E-R)图
  - 作用：对现实世界进行抽象，以用户观点对数据建模
  - 基本概念
    - 实体，属性，联系
  - 表示符号
    - 矩形(实体型)，椭圆(属性)，菱形(联系)

小结

- 状态转换图
  - 作用：描绘系统状态及引起状态转换的事件，表示系统行为
  - 基本概念
    - 状态(初始、最终、中间)，事件(引起系统动作/状态转换)
  - 表示符号
    - 实心圆(初始状态)，同心圆(最终状态)，圆角矩形(中间状态)
    - 箭头线(状态转换)，事件表达式(触发条件)
- 其他图形工具
  - 层次方框图，Warnier/Orr图，IPO图
- 验证软件需求
  - 一致性、现实性、完整性、有效性

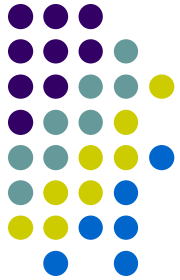
小结

- 形式化方法
  - 基于数学工具描述系统性质
  - 非形式化➔半形式化➔形式化
  - 非形化方法的缺点
    - 存在矛盾，二义性，模糊性，不完整性，抽象层次混乱
- 有穷状态机
  - 五元组：状态集，初始状态，终止状态集，输入集，转换函数
- Petri网
  - 四元组：位置集，转换集，输入函数，输出函数
  - 附加元素：位置集至非负整数映射（token）
  - 转换激活条件，token变化规则，禁止线

小结

- 总体设计
  - 目的：以抽象概括的方式确定系统如何完成预定任务
  - 任务
    - 系统设计：描述组成系统的物理元素（物理实现方案）
    - 结构设计：确定系统的程序模块，以及模块间的相互关系（软件实现方案）
  - 设计过程：系统设计阶段➔结构设计阶段
    - 系统设计：设想供选择方案➔选取合理方案➔推荐最佳方案
    - 结构设计：功能分解➔设计软件结构➔设计数据库➔制定测试计划➔书写文档➔审查和复查➔……

小结



- 软件结构设计：设计原理
  - 软件结构设计过程中应遵循的基本原理和相关概念
  - 思维方式：抽象+逐步求精
  - 求解方式：模块化+信息隐藏和局部化
  - 思维方式+求解方式➔模块独立
    - 意义：易于开发、测试和维护
    - 耦合(Coupling)：衡量模块彼此间相互依赖/连接的紧密程度
      - 耦合程度由低至高(越低越好)：无耦合，数据耦合，特征耦合，控制耦合，公用耦合，内容耦合
    - 内聚(Cohesion)：衡量模块内部各元素彼此结合的紧密程度
      - 内聚程度由高至低(越高越好)：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

小结

- 软件结构设计：启发式规则
  - 具有重要参考价值，有助于改进设计，提高软件质量
    - 改进软件结构提高模块独立性；模块规模需适中；深度、宽度、扇入、扇出均应适当；模块作用域在控制域之内；力争降低模块接口的复杂程度；设计单入口单出口的模块；模块功能应可以预测
  - 既非设计原理，亦非设计目标
- 软件结构设计：图形工具
  - 层次图：描述模块间调用层次关系
  - HIPO图：层次图+IPO图
  - 结构图：表征模块间传递的数据或控制信息
    - 数据信息：尾部空心圆箭头
    - 控制信息：尾部实心圆箭头
    - 附加符号：菱形尾部（选择调用），圆弧箭头（循环调用）

小结

- 软件结构设计：面向数据流图的方法
  - 目的：为软件结构设计提供系统化的途径
  - 基本策略：面向信息(数据)流，通过“映射(mapping)”的方式将DFD变换成软件结构
  - 信息(数据)流类型
    - 变换流(transformation flow)
    - 事务流(transaction flow)
  - 映射方法
    - 变换分析

    复查基本系统模型➔复查并精化DFD➔确定DFD变换/事务特性➔确定输入/输出流边界➔“第一级”分解➔“第二级”分解➔进一步精化

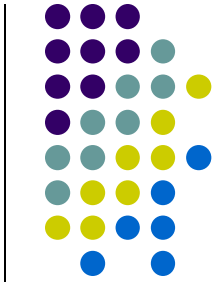
    - 事务分析

    接收分支+发送分支(调度模块，活动模块)

小结

- 详细设计
  - 目的：得出对目标系统的精确描述
  - 任务：设计程序的“蓝图”，而非具体地编写程序
  - 关键技术：结构程序设计
    - 三种基本控制结构：顺序+选择+循环
    - 经典/扩展/修正的结构程序设计
- 人机界面设计
  - 设计问题
    - 系统响应时间、用户帮助设施、出错信息处理、命令交互
  - 设计过程
    - 创建设计模型➔原型实现设计模型➔用户试用评估➔修改设计模型➔……
  - 设计指南
    - 一般交互指南，信息显示指南，数据输入指南

小结



- 过程设计工具
  - 作用：描述程序处理过程
  - 基本要求：能提供对设计的无歧义的描述
  - 类型
    - 图形工具：程序流程图、盒图(NSD)、PAD图
    - 表格工具：判定表、判定树
    - 语言工具：过程设计语言(PDL)
- 面向数据结构的设计方法
  - Jackson图
    - 用于描绘数据结构/程序结构：顺序、选择、重复
  - 五基本步骤

小结

- 程序复杂度定量度量
  - 目的：定量(quantitative)度量软件的性质
  - McCabe方法
    - 使用流图(亦称程序图)表示程序的控制流：圆、箭头线、区域
    - 程序流程图/PDL描述➔流图
    - 三种复杂度计算方法(环形复杂度)
  - Halstead方法
    - 运算符(operator)、操作数(operand)
    - 基本度量：N_1, N_2, n_1, n_2
    - 八种Halstead度量
- 编码
  - 如何选择程序设计语言：理想标准/实用标准
  - 编码风格：内部文档、数据说明、语句构造、I/O、效率

Part II: 面向对象分析与设计

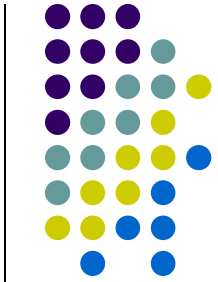
小结

- OO方法学
  - 出发点与基本原则
    - 问题域与求解域在结构上尽可能一致，将数据与处理相结合
  - OO方法学思维方式
    - 减少问题域与求解域之间存在的“语义鸿沟(semantic gap)”
  - OO方法学要点
    - OO=Objects + Classes + Inheritance + Encapsulation
  - OO方法学优点
    - 与人类习惯的思维方法一致，稳定好，可重用性好，较易开发大型软件产品，可维护性好
- OO方法学概念
  - 对象(object)、类(class)、实例(instance)、消息(message)、方法(method)、属性(attribute)、封装(encapsulation)、继承(inheritance)、多态性(polymorphism)、重载(overloading)

小结

- 面向对象建模
  - 对象模型＋动态模型＋功能模型
  - 统一建模语言UML
    - 发展过程，UML模型图，UML应用领域
- 对象模型
  - UML工具：类图(class diagram)
    - 定义类：类名＋属性＋服务
    - 定义属性：可见性 属性名：类型名=初值{性质串}
    - 定义服务/操作：可见性 操作名(参数表)：返回值类型{性质串}
    - 类与类之间的关系：
      - 关联(association)：关联名称，关联重数，关联角色，限定关联，关联类
      - 聚集(aggregation)：共享聚集，组合聚集
      - 泛化/继承(generalization)：普通泛化，受限泛化
      - 依赖和细化(dependency & realization)

小结



- 动态模型
  - 状态，事件，行为
  - UML工具：状态图(state machine diagram)
- 功能模型
  - UML工具：用例图(use case diagram)
    - 系统(system)，行为者(actor)，用例(use case)
    - 用例间的关系：扩展关系，使用关系
  - 用例建模
    - 定义系统➔寻找行为者和用例➔描述用例➔定义用例关系➔确认模型
    - 如何寻找行为者
    - 如何寻找用例
- 三种模型间的关系

小结

- 面向对象分析(Objected-Oriented Analysis, OOA)
  - 目的
    - 抽取和整理用户需求并建立问题域精确模型
  - 基本过程
    - 分析需求陈述文件➔深入理解用户需求，用模型准确表示
  - OOA三个子模型
    - 对象模型：基础模型
    - 动态模型：交互、实时系统
    - 功能模型：大规模运算(处理)问题
  - 对象模型五个层次
    - 主题层、类与对象层、结构层、属性层、服务层
    - OOA对象建模主要活动

小结

- 需求陈述
  - 陈述内容
    - 问题范围，功能需求，性能需求，应用环境等
  - 陈述要点
- 建立对象模型
  - OOA首要工作
  - 典型工作步骤
    - 确定对象类和关联
    - 对大型复杂问题划分出若干主题
    - 确定类和关联的属性
    - 基于继承关系进一步合并和组织类
    - 确定类的操作(基于动态模型和功能模型)

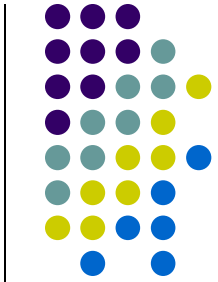
小结

- 建立动态模型
  - 交互式系统OOA的重要工作
  - 基本步骤
    - 编写脚本➔设想用户界面➔画事件跟踪图➔画状态图➔审查动态模型
- 建立功能模型
  - OOA描述数据处理的必要工作
  - 方法
    - 基于DFD：基本系统模型图➔功能级DFD➔描述处理框功能
    - 基于UML，如用例图(use case diagram)等
- 定义服务

小结

- 面向对象设计(Objected-Oriented Design, OOD)
  - 目的
    - 基于OOA结果，以面向对象观点建立求解域精确模型
  - OOD设计准则
    - 模块化、抽象、信息隐藏、可重用、弱耦合、强内聚
  - OOD启发规则
- 软件重用
  - 软件重用层次、软件成分重用级别
  - 典型可重用软件成分
  - 类构件
    - 实例重用、继承重用、多态重用

小结

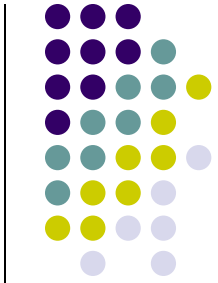


- 系统分解
  - 典型OOD模型：5个层次+4个子系统
  - 设计问题域子系统、设计人机交互子系统、设计任务管理子系统、设计数据管理子系统
  - 设计优化：提高效率+调整继承关系+委托
- 面向对象实现(Objected-Oriented Programming, OOP)
  - 任务：编码+测试
  - 选择OO语言需关注的技术特点
  - 面向对象程序设计风格
    - 可重用性、可扩充性、健壮性

Part III:

软件测试、维护及项目管理

小结



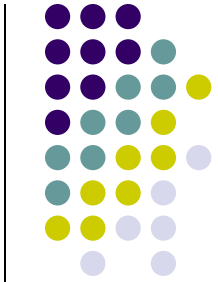
- 软件质量
  - 狭义理解：3A标准
  - 广义理解：产品质量，过程质量，商业化表现
- 软件缺陷
  - 种类：致命(fatal)，严重(critical)，一般(major)，微小(minor)
  - 产生原因：技术问题，团队工作，软件本身
  - 构成：规格说明书，设计，代码，其它

  软件测试≠找代码中的错误
- 软件测试的基本方法
  - 黑盒 vs 白盒
  - 静态 vs 动态
  - 自动化测试 vs 手工测试
  - 验证 vs 确认

小结

- 软件测试的分类
  - 测试范围
  - 测试目的
  - 测试对象
  - 测试过程
- 软件测试阶段
  - 自底向上，逐步集成
  - 单元测试➔集成测试➔功能测试➔确认测试➔系统与验收测试➔安装测试
- 软件测试的工作范畴
  - 技术工作+组织和管理工作

小结

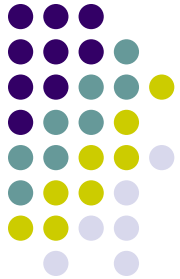


- 什么是单元测试
  - 对软件基本组成单元进行的测试，是最早期的测试活动
- 单元测试的目标
  - 单元模块被正确编码
- 单元测试的任务
  - 接口测试，局部结构测试，边界条件测试，独立执行通路测试，错误处理通路测试
- 静态测试技术的运用
  - 走查，审查，评审
- 动态测试技术的运用
  - 白盒测试为主，黑盒测试为辅
- 单元测试工具

小结

- 集成测试
  - 增量式(Incremental Test)集成测试
    - 自顶向下，自底向上，三明治
  - 非增量式(Non-Incremental Test)
    - 大棒集成
- 功能测试
  - 功能测试内容
  - 功能测试方法
    - 等价类划分，边界值分析，因果图，错误推断，组合分析
- 系统测试
  - 压力测试，容量测试，性能测试
  - 安全性测试，可靠性测试，容错性测试

小结

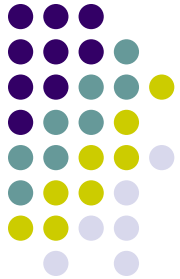


- 软件测试用例重要性
  - 有效性，易组织性，可评估性，可复用性，可管理性
- 软件测试用例书写标准
  - ANSI/IEEE 829-1983
- 白盒测试用例设计
  - 逻辑覆盖法
    - 语句覆盖，判定覆盖，条件覆盖，判定-条件覆盖，条件组合覆盖，路径覆盖
- 黑盒测试用例设计
  - 等价类划分法，边界值分析法，因果图法，错误推断法，功能图法

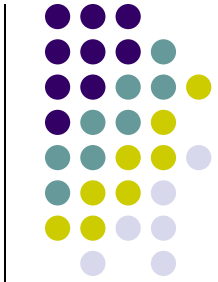
小结

- 软件维护
  - 目的
    - 软件交付后，为改正错误或满足新需要而修改软件的过程
  - 任务
    - 改正性维护、适应性维护、完善性维护、预防性维护
  - 工作量模型
    - Belady & Lehman Model
  - 软件维护过程
    - 建立维护组织➔制定维护报告➔确定维护事件序列➔保存维护记录➔评价维护活动
  - 软件可维护性
    - 影响因素，文档，可维护性复审

小结



- 软件项目管理
  - 目的
    - 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
  - 任务
    - 项目计划
      - 软件规模估计：代码行技术，功能点分析等
      - 工作量估计：单变量模型，多变量模型，构造性成本模型等
      - 进度估计：Gantt图，关键路径法等
    - 项目组织
      - 民主制程序员组，现代程序员组
    - 项目控制
      - CMM(软件能力成熟度模型)：初始级(initial)，可重复级(repeatable)，已定义级(defined)，已管理级(managed)，优化级(optimizing)



考试时间：

1月20日(周四)下午2:00-4:00

考试地点：

教六-202

总成绩构成：

平时成绩(5%)+Project (20%)+期末考试(75%)

考试方式：

闭卷考试

题型：

判断题：10*1.5=15分

选择题：10*1.5=15分

简答题：4*5=20分

问答题：15+15+20=50分