**软件工程实践课程教学目标**

**课程的优势：**

1. **采用任务驱动的方式组织教学案例，将软件工程课程的内容融入案例教学之中，使学生在真实的任务环境下探索、学习。在这个过程中，学生可以不断地获得成就感、愉悦感，进一步激发学生的好奇心和求知欲，逐步形成一个认知活动的良性循环，从而培养出独立探索、勇于开拓进取的自学能力。**
2. **教学案例基于OBE教育理念，由易到难，为学生地深度思考搭脚手架，启发学生主动思考，激发学生的学习兴趣，调动学生学习的积极性、充分体现学生的主体作用。**
3. **培养学生自主学习能力。案例覆盖知识面全面完整，有针对性地设计案例，每个案例自成一体，聚焦到软件工程中系统分析、设计、实现、测试、维护等地某一特定知识点或知识单元。通过一个案例实践，使学生对软件过程有一个较全面的理解，培养学生的科学精神和工程设计能力。**
4. **训练学生的科学思维。使学生在掌握基本方法的基础上，自主探索，举一反三，触类旁通，从而训练学生的科学思维方式和创新能力。**
5. **前沿科学研究成果，可以开拓知识的深度和广度。**

# 概述

## 知识目标：

1. 了解软件危机发生的背景（两个时期）
2. 了解软件危机的几种常见表现
3. 了解软件危机产生的原因（可从客户角度、软件角度和开发者角度，亦可从客观原因和主观原因的角度）
4. 理解软件危机应对途径
5. 能够定义软件工程
6. 能够描述软件工程的7个本质特性
7. 了解软件工程的基本原理
8. 清楚知道软件软件工程方法学，理解传统方法学（结构化范型）和面向对象方法学（面向对象范型）
9. 理解软件过程的概念。
10. 了解软件生命周期的3个时期及其细分的8阶段（包括：问题定义、可行性研究与计划、需求分析、总体设计、详细设计、系统实现、系统测试、运行与维护）的内涵、基本任务
11. 理解软件生命周期模型（包括：瀑布模型、快速原型模型、增量模型、螺旋模型、喷泉模型、RUP、敏捷过程、微软过程）的定义、内涵、图形化表示、优缺点、适用场景

## 能力目标：

1. 能区分各种生命周期模型
2. 能根据项目描述判断该项目适用的生命周期模型

## 素质目标：

1. 树立正确的“危机观”，能够辩证看待“危”和“机”的关系。
2. 提升洞察力，能够透过软件危机的外在现象看到内在本质。
3. 提升定义问题、分析问题和解决问题的能力。

# 可行性研究

## 知识目标：

1. 了解可行性研究的定义、目的和意义
2. 掌握可行性研究的任务
3. 掌握可行性研究的内容
4. 掌握可行性研究的步骤
5. 掌握系统流程图的概念和组成元素
6. 掌握数据流图的概念和组成元素
7. 理解代码行技术、任务分解技术等工作量估算方法
8. 理解投资回收期、投资收益率等成本/效益分析方法
9. 理解法律方面的可行性分析和社会效益方面的可行性分析
10. 能用头脑风暴法、德尔菲法进行风险分析
11. 理解风险应对的四种策略

## 能力目标：

1. 能够撰写可行性研究报告
2. 能够针对一个项目分析其技术、经济和操作可行性
3. 能用头脑风暴法、德尔菲法进行风险分析

## 素质目标：

1. 具有公正心，能够对客观情况进行务实判断
2. 有全局观，能综合分析各个方案的优缺点
3. 具有发现问题、分析问题和解决问题的能力

# 结构化分析

## 知识目标：

1. 了解软件需求分析的定义、的任务和过程
2. 了解系统的8个综合要求（功能需求、性能需求、可靠性和可用性需求、出错处理需求、接口需求、约束、逆向需求、将来可能提出的要求）
3. 理解需求获取的4类方法
4. 会用访谈、面向数据流自顶向下求精、简易的应用规格说明技术、快速建立软件原型等4类需求获取方法，与用户沟通获取用户需求
5. 理解实体-联系图（E-R图）的概念，实体、联系、属性
6. 理解数据规范化的概念
7. 理解第一范式（1NF）第二范式（2NF）第三范式（3NF）
8. 了解层次方框图、Warnier图、IPO图等图形工具
9. 理解如何从一致性、现实性、完整性和有效性等4个方面验证软件需求的正确性
10. 理解Petri网的概念、Petri网包含的4种元素，理解Petri网的工作原理

## 能力目标：

1. 能够根据需求画出E-R图
2. 能根据需求画出系统的状态转换图
3. 能用Petri网表示系统的规格说明
4. 能够制作软件需求规格说明书SRS

## 素质目标：

1. 具有换位思考的意识.
2. 懂得尊重人.
3. 能够用对方的立场思考问题。
4. 能够用心倾听，善解人意。

# 结构化设计

## 知识目标：

1. 了解模块以及模块化的基本思想，理解模块化的原因
2. 理解抽象、逐步求精、信息隐藏的概念，理解抽象与求精之间的关系，理解信息隐藏与模块化、抽象的关系
3. 理解Miller法则
4. 了解心理学家George A. Miller (February 3, 1920 – July 22, 2012)提出米勒法则的过程
5. 了解提出逐步求精思想的图灵奖得主尼古拉斯·沃斯（Niklaus Wirth，1934.2.15—）生平
6. 理解模块独立的概念
7. 掌握耦合的概念
8. 掌握数据耦合、控制耦合、特征耦合、公共环境耦合、内容耦合的概念
9. 掌握内聚的概念
10. 掌握偶然内聚，逻辑内聚，时间内聚，过程内聚，通信内聚，顺序内聚，功能内聚，能判断模块的内聚类型
11. 理解7条常用的启发规则。
12. 掌握层次图、HIPO图及结构图的概念，以及各图中符号的含义
13. 掌握根据DFD画出初始结构图的方法
14. 掌握面向数据流的设计方法，掌握数据流图映射到结构图的步骤
15. 理解变换流与事务流的区别
16. 理解软件详细设计时用到的各类工具的优缺点，包括：伪代码、程序流程图、盒图（N-S图）、PAD图、判定树、判定表等
17. 能够根据系统需要选择合适的结构化软件设计工具。
18. 理解Theo Mandel提出的人机界面设计三条“黄金原则”
19. 理解界面设计的4个问题：系统响应时间、用户求助机制、出错信息、命令方式
20. 掌握程序复杂程度的定量度量方法McCabe方法和Halstead方法
21. 理解软件设计评审的概念，评审目标、评审原则。

## 能力目标：

1. 会把控制耦合转换成数据耦合
2. 能够根据需求规格说明书中的数据流图画出结构图
3. 能把数据流图转换成初始的系统结构图
4. 能对初始的系统结构图进行优化
5. 能把变换流转换成变换型结构图
6. 能把事务流转换成事务型结构图
7. 能够根据伪代码画出程序流程图、盒图（N-S图）、PAD图
8. 会画判定表和判定树
9. 能用McCabe方法画出程序流程图的流图，并计算程序复杂程度
10. 能用Halstead方法预测程序长度和程序中包含错误的个数
11. 能够制作软件设计说明书
12. 能够用软件设计评审方法进行评审

## 素质目标：

1. 具有结构化、模块化的思想
2. 具有系统思维、全局意识和统筹能力
3. 具有独立思考的精神。
4. 在设计过程中具有细致、严谨的素质和精益求精的大国工匠精神
5. 有开发国产软件的自信和责任担当
6. 有创新精神、实践能力和团队协作精神

# 面向对象分析

## 知识目标：

## 能力目标：

## 素质目标：

# 面向对象设计

## 知识目标：

## 能力目标：

## 素质目标：

# 系统实现

## 知识目标：

1. 了解各类程序设计语言的发展历程
2. 理解各类程序设计语言的特点
3. 理解程序设计风格，掌握编程规范，包括：程序内部的文档、数据说明、语句构造、输入输出、效率等。
4. 了解面向对象语言的优势、技术特点
5. 理解面向对象编程时的编程规范
6. 理解可重用性、可扩充性、健壮性等概念
7. 掌握提高可重用性的准则
8. 掌握提高可扩充性的准则
9. 掌握提高健壮性的准则

## 能力目标：

1. 能根据项目特点选择合适的程序设计语言
2. 能设计出规范的命名
3. 能写出简洁合理的序言性注释和功能性注释
4. 能合理利用空格、空行、缩进等技巧，让源程序的视觉组织上清晰易懂，设计出整洁的布局
5. 能设计出规范的数据说明顺序
6. 能够设计出变量说明有序的语句
7. 能够对复杂的数据结构加上注释说明
8. 编写程序时能做到一行写1个语句，判定条件简单，循环嵌套少，表达式中使用括号
9. 对所有输入数据进行检验，保证数据的有效性
10. 检查输入项重要组合的合法性和有效性，报告必要的输入状态和信息及错误信息
11. 保持输入格式简单
12. 使用数据结束标记，不要要求用户指定数据的数目
13. 明确提示交互式输入的请求，详细说明可用的选择或边界数值
14. 当程序设计语言对格式有严格要求时，应保持输入格式一致
15. 设计良好的输出报表
16. 给所有输出数据加标志
17. **能**根据面向对象语言的优势及特点，为项目选择合适的面向对象编程语言
18. 能编写出可重用性高、可扩充性强、健壮的面向对象程序

## 素质目标：

1. 勤奋务实，精益求精，低调谦虚
2. 具有规则意识，责任意识，有敬业精神
3. 具有规则意识，责任意识，有敬业精神

# 软件测试

## 知识目标：

1. 理解软件测试的目标、准则、步骤和方法。
2. 理解黑盒测试和白盒测试
3. 理解单元测试、子系统测试、系统测试、验收测试、平行运行
4. 理解单元测试的5个方面内容
5. 理解代码审查
6. 理解驱动程序和存根程序
7. 了解非渐增式测试
8. 理解渐增式集成测试及其两种集成策略
9. 理解回归测试
10. 理解确认测试（验收测试），理解确认和验证的内涵
11. 理解Alpha和Beta测试
12. 理解逻辑覆盖
13. 理解语句覆盖、判定覆盖、条件覆盖、判定/条件覆盖、条件组合覆盖、点覆盖、边覆盖、路径覆盖的内涵
14. 理解等价划分
15. 理解等价划分的6条启发规则
16. 理解软件调试的过程
17. 了解hopper女士及其发现的第一个“bug”
18. 理解蛮干法、回溯法
19. 理解软件可靠性、可用性概念
20. 能够估算平均无故障时间MTTF，能够用平均无故障时间MTTF和平均维修时间MTTR计算系统的稳态可用性

## 能力目标：

1. 能够对他人编写的程序进行代码审查
2. 能够编写驱动程序和存根程序
3. 能够进行自顶向下集成测试和自底向上集成测试
4. 能够根据模块的内部结构设计测试用例，实现语句覆盖、判定覆盖、条件覆盖、判定/条件覆盖、条件组合覆盖、点覆盖、边覆盖、路径覆盖
5. 能够根据等价划分设计测试用例，对常见程序进行测试
6. 会用边界值分析
7. 会用原因排除法进行软件调试
8. 会用植入错误法和分别测试法

## 素质目标：

1. 具有细致、严谨的观察能力，能否发现细微处的bug
2. 具有洞察力，能窥一斑而知全貌
3. 具有精益求精的工匠精神
4. 理解“是人都会犯错”，尽力避免错误，有反思精神，勇于自我批评；乐于接受批评，懂得感恩
5. 有团队精神，任何代码的修改都要经过协商
6. 低调谦虚，尊重比自己弱的“新手”
7. 能够独立思考，有独立人格
8. 实事求是，用事实说话，不要用自己的地位强迫别人接受自己的观点
9. 对事不对人，可以批评代码（code），同时要友好对待编写代码的人（coder）
10. 有敬业精神和法治意识，有良好的职业操守

# 维护

## 知识目标：

1. 理解系统部署的概念
2. 理解软件维护的定义和4类软件维护活动
3. 理解软件维护的3个特点
4. 理解软件维护工作量模型
5. 理解软件维护过程
6. 理解软件维护组织的结构
7. 理解软件维护的事件流
8. 了解软件维护记录的内容
9. 了解维护评价活动的7个方面工作
10. 理解软件的可维护性的定义
11. 了解90-10测试
12. 理解可理解性
13. 了解可理解性度量的检查表
14. 理解可测试性
15. 了解可测试性度量的检查表
16. 理解可修改性
17. 了解可修改性度量的检查表
18. 理解可移植性
19. 了解可移植性度量的检查表
20. 理解可重用性
21. 了解提高可重用性的方法
22. 理解文档的定义和结构
23. 理解用户文档和系统文档的定义和内容
24. 理解可维护性复审
25. 理解预防性维护
26. 理解软件再工程范型
27. 理解软件再工程范型的6类活动
28. 理解软件再工程过程

## 能力目标：

1. 能够在真实环境下部署一个系统
2. 能够用软件维护工作量模型计算软件维护工作量
3. 能够写软件维护要求表
4. 能够进行改正性维护
5. 能够进行适应性维护
6. 能够进行完善性维护
7. 能够针对部分模块进行文档重构
8. 能够对可疑模块进行代码重构

## 素质目标：

1. 有洞察力，能够发现软件的问题
2. 能换位思考，团结用户，做好良好沟通
3. 具有协调能力，能够平衡原则性和灵活性
4. 有自我反思精神，能够做到持续改进
5. 有法制意识，能够遵守软件相关法律法规
6. 有敬业精神和责任意识，对发现的问题能够担当

# 软件项目管理

## 知识目标：

## 能力目标：

## 素质目标：

# 前沿技术

## 知识目标：

## 能力目标：

## 素质目标：