软件工程导论

- 软件危机：在计算机软件开发和维护过程中，所遇到的一系列严重问题。
- 软件工程各部分工作量：
  - 编写程序10~20%
  - 测试40~50%
  - 维护占软件总费用55~70%
- **软件是程序、数据及相关文档的完整集合。**
  - 程序是能够完成预定功能和性能的可执行指令序列。
  - 数据是使程序能够适当地处理信息的数据结构。
  - 文档是开发、使用和维护程序所需要的图文资料。
- 软件工程是从**管理**和**技术**两方面研究如何开发和维护计算机软件的学科。
- 软件工程：采用工程的概念、原理、技术和方法来开发与维护软件。
- 软件工程的本质特性：
  1. 关注于大型程序的构造。
  2. 中心课题是控制复杂性。
  3. 软件经常变化。
  4. 开发软件的效率非常重要。
  5. 和谐的合作是开发软件的关键。
  6. 软件必须有效地支持它的用户。
  7. 在软件工程领域中，通常由具有文化背景的人替具有另一种文化背景的人创造产品。
- **软件工程方法学三个要素：方法、工具、过程。**
  - 传统方法学：采用结构化技术
  - 面向对象方法学：把数据和行为看成同等重要，以数据为主线，把数据和对象的操作紧密地结合。
    - 四个要点：对象、类、继承、用消息通信。
- 软件生命周期：
  - 软件定义：
    - **问题定义**：要解决的问题是什么。
    - **可行性研究**：有行得通的解决办法吗，系统分析员需要一次压缩和简化的系统分析和设计过程。
    - **需求分析**：确定目标系统具备哪些功能。，撰写规格说明书。
  - 软件开发：
    - **总体设计**：概括地说如何实现系统，确定程序由哪些模块组成及模块之间的关系。
    - **详细设计**：如何具体实现该系统，详细设计每个模块，确定需要的算法和数据结构。
    - **编码和单元测试**：写出正确，容易理解、容易维护的程序模块；仔细测试编写的每一个模块。
    - **综合测试**：
      - 集成测试：模块装配过程中进行测试。

        - 验收测试：由用户对目标系统进行验收。
    - 软件维护：
        - **运行维护**：使系统持久地满足用户的需要。
- 软件过程：通常使用**软件生命周期**模型描述软件过程。生命周期模型规定了把生命周期划分成哪些阶段及各个阶段的执行顺序，也称过程模型。
    - 瀑布模型（文档驱动）
        - 阶段间具有顺序性和依赖性：
            - 必须完成前一阶段的工作，才能开始后一阶段的工作。
            - 前一阶段的输出文档就是后一阶段的输入文档。
        - 清楚地区分逻辑设计和物理设计，尽可能推迟程序的物理实现。
        - 质量保证观点：
            - 每个阶段都要完成规定的文档，没有交出合格的文档就是没有完成该阶段的任务。
            - 每个阶段结束前都要对完成的文档进行评审。
        - 缺点：开发过程一般不能逆转，代价太大，可能最终产品不能满足用户需要。
    - 快速原型模型
        - 不带负反馈环，频繁变化，然后废弃。
    - 增量模型
        - 将软件产品作为一系列增量构建来设计、编码、集成和测试。
        - 优点：
            - 能够在较短时间内向用户提交可完成部分工作的产品。
            - 逐步增加产品功能，使用户有较充裕的时间学习新产品。
    - 螺旋模型（风险驱动）
        - 每个阶段之前都增加了风险分析过程的快速原型模型。
    - 喷泉模型
        - 体现面向对象软件开发过程迭代和无缝的特性。
        - 在分析、设计、和编码等开发活动之间不存在明显的边界。
    - 极限编程
        - 客户作为开发团队成员。
        - 使用用户素材。
        - 短交付周期。
        - 验收测试。
        - 测试驱动开发...
    - RUP（强调迭代和渐增）
        - 二维的生命周期模型。
        - 四个阶段的工作目标：
            - 初始阶段：建立业务模型，定义最终产品视图，确定项目的范围。
            - 精化阶段：设计并确定系统的体系结构，制定项目计划，确定资源需求。

        - 构建阶段：开发出所有构件和应用程序，将它们集成为客户需要的产品，并详尽地测试所有功能
        - 移交阶段：把开发出的产品提交给用户使用。

第二章 可行性研究（重点：数据流图）

- 确定问题是否值得去解决。
- 系统流程图是概括地描绘物理系统的传统工具。
- 数据字典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共同构成系统的逻辑模型。

第三章 需求分析（重点：ER图，状态转换图）

- 需求分析阶段的任务包括**确定对系统的综合要求**和**分析系统的数据要求。**
    - 综合要求包括：功能需求、性能需求、可靠性和可用性需求、出错处理需求、约束、接口需求、逆向需求和将来可能提出的要求。
- 从四个方面验证需求的正确性：一致性、完整性、现实性、有效性。
- 在需求分析阶段结束之前，系统分析员应该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，该过程中，分析员和用户都起着关键作用。
- 访谈是迄今为止仍广泛使用的需求分析技术。
- 使用ER图建立数据模型，使用数据流图建立功能模型，使用状态图建立行为模型。

第五章 总体设计

- 两个阶段组成：
    - 系统设计阶段：确定系统的具体实现方案。
    - 结构设计阶段：确定软件结构。
- 包含的过程：
    - 设想并选取合理的方案
    - 推荐最佳方案
    - 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
    - 审查和复审
- 耦合度从高到底：**内容耦合**、公用耦合、控制耦合、印记耦合、**数据耦合**
- 内聚从高到低：**功能内聚**、顺序内聚、通信内聚、过程内聚、时间内聚、逻辑内聚、**偶然内聚**
- 尽量低耦合，高内聚。即数据耦合，功能内聚。
- 启发规则：
    - 深度：软件结构中控制的层数
    - 宽度：软件结构内同一个层次上的模块总数

    - 扇出：一个模块直接控制的模块数目
    - 扇入：一个模块有多少个上级模块调用它
    - 良好的软件结构应该是：顶层扇出比较高，中层扇出较少，底层模块有高扇入。
- 层次图和结构图是描绘软件结构的常用工具。

第六章 详细设计（重点：盒图、程序流程图、PAD图、判定表）

- 结构化程序：代码块仅仅通过顺序、选择和循环3种基本控制结构进行连接，每个代码块只有一个入口和一个出口，则称该程序是结构化的。
- 经典的结构程序设计：只允许顺序、选择、循环
- 扩展的结构程序设计：还允许do-while，switch case
- 修正的结构程序设计：再允许leave/break
- 盒图（N-S图）
- 问题分析图（problem analysis diagram, PAD图）
- 判定表：算法中包含多重嵌套的条件时。
- 过程设计语言：伪码

第七章 实现

- 编码和测试统称为实现。
- 程序内部文档：
    - 序言性注解：每个模块开始处有一段序言性的注解。
    - 解释性注解：插在程序中间与一段程序代码有关的注解。
- 测试的目标：
    - 为了发现程序中的错误而执行程序的过程。
    - 好的测试方案是极可能发现迄今为止尚未发现的错误的测试方案。
    - 成功的测试是发现了至今为止尚未发现的错误的测试。
- 测试只能查找出程序中的错误，不能证明程序中没有错误。
- 测试方法：
    - **黑盒测试（验收测试）**：在程序接口进行的测试，又称功能测试，数据驱动。
        - **等价划分**：
        - **边界值分析**：
    - **白盒测试（单元测试）**：测试者完全知道程序的结构和处理算法，又称结构测试，逻辑驱动。
        - 语句覆盖：每个语句执行一次。
        - 判定覆盖：每个判定的每个分支执行执行一次。
        - 条件覆盖：每个条件都取到各种可能的结果。
        - 条件组合覆盖：每个判定表达式中条件的各种可能组合都至少出现一次。
- 测试步骤：
    - 系统测试（集成测试）：验证系统确实能提供需求说明书中指定的功能。
        - 自顶向下测试：

        - 优点：不需要测试驱动程序，能够在测试阶段的早期实现并验证系统的主要功能。
        - 缺点：要存根程序，低层模块中错误发现较晚。
    - 自底向上测试：
        - 优点：可复用模块得到充分测试。
        - 缺点：主要涉及错误发现迟。
    - 三明治集成测试：对软件结构中较上层采用自顶向下，较下层使用自底向上。
  - 验收测试：发现系统说明书中的错误。
  - 确认测试：
    - Alpha测试：由用户在开发者的场所进行，且在开发者对用户的指导下进行测试。
    - Beta测试：开发者不在测试现场。

第八章 维护

- 软件维护定义：在软件已经交付使用后，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：诊断和改正错误（17~21%）
  - 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（18-25%）
  - 完善性维护：使用软件过程中，用户提出增加新功能或修改已有功能的建议（50~66%）
  - 预防性维护（4%）

第九章 面向对象方法学引论

- 面向对象的概念要点：对象、类、继承通信
  - **描述系统数据结构的对象模型**
  - **描述系统控制结构的动态模型**
  - **描述系统功能的功能模型**
- 类图：
  - 重数：多少个对象与对方的一个对象关联。
    - 0..1
    - 0..*或*，0到多个对象
    - 1+或1..*，1到多个对象
    - 1..15，1到15个对象
  - 限定关联：
    - 如[目录][文件名]---[文件]
  - 聚集：
    - 共享聚集（白菱形）：一个课题组含多个成员，每个成员可以是另一个课题组的成员。
    - 组合聚集（黑菱形）：部分类完全隶属于整体类。
  - 泛化/继承（白三角）
- 用例图（带小人）
- 顺序图：
  - 一条竖线代表一个对象，每个事件用一条水平的箭头表示。

第十一章 面向对象设计

- 分类：系统设计、对象设计