ch1-绪论

- 软件
  - 定义：P3
  - 特性（P4磨损曲线）
    - 软件的生产和硬件不同，没有明显的制造过程。
    - 软件不会磨损。
    - 软件开发至今尚未完全摆脱手工开发方式。
    - 软件是一种逻辑实体，具有抽象性。
    - 软件的开发和运行受到计算机系统的限制，对计算机系统有不同程度的依赖性
  - 软件危机
    - 概念：是指落后的软件生产方式无法满足迅速增长的计算机软件需求，从而导致软件开发与维护过程中出现一系列的严重问题的现象。
    - 产生原因
      - 1)与软件的复杂性和本身的特点有关；
      - 2)由软件开发、维护、管理方法不正确有关
    - 表现 （ppt6）
      - Do not provide the desired functionality无法满足用户需求
      - Take too long to build开发时间长
      - Cost too much to build 花销大
      - Require too much resources (time, space) to run 资源需求大
      - Cannot evolve to meet changing needs无法随着需求变化而变化
      - Quality can not be guaranteed质量无法保证
    - 软件分类（P5）
    - 软件发展

	程序设计时代	程序系统时代	软件工程时代
时间	1946-1956	1956-1968	1968至今
生产方式	个体手工劳动	作坊式小集团合作	工程化的生产
使用工具	机器、汇编语言	高级语言	软件语言
开发方法	个人编程技巧	个人编程技巧，开始提出结构化方法	使用数据库、网络、分布式、面向对象等技术
硬件特征	价格贵，存储容量小，运行可靠性差	速度、容量及工作可靠性明显提高，价格降低	向超高速、大容量、微型化以及网络化方向发展
软件特征	只有程序设计概念，不重视程序设计方法	程序员数量猛增，但开发人员素质差，开发技术没有新突破，软件危机产生	开发技术有很大进步，但没有突破性进展，没有完全摆脱软件危机
软件定义	等于程序	等于程序加技术文档	完整定义

- 软件工程
  - 定义（P11）
  - 主要研究

    软件工程主要研究软件生产的客观规律，建立与系统化软件生产有关的概念、原则、方法、技术和工具，指导和支持软件系统的生产活动，以期达到降低软件生产成本、改进软件产品质量、提高软件生产率的目标。
  - 核心要素：P11，ppt10

图 2-1　软件工程层次图

    - 1. 质量：软件工程根基
    - 2. 过程，软件工程基础
      - 定义了解决特定问题或目标达成涉及的一系列步骤、活动以及交付的内容等。
      - 软件产品开发包括若干子过程：**开发过程、维护过程、管理过程等；**
      - 注意点
        - （1）过程与质量密切相关；
        - （2）不同过程适用于不同项目
    - 3. 方法：如何做的问题→定义“开发过程中每项活动，任务的实现方法”。
    - 4. 工具

        - 方法的实现需要支撑工具，如分析工具、设计工具、编译工具、测试工具等。如计算机辅助软件工程（CASE，工具集成，以被另一个工具调用）。
- 软件过程框架（p12, ppt15）
    - 概念：开发全部活动、动作、任务的结构框架。
    - 软件开发活动的层次结构：
        - 进程(process)→阶段(activity)→行动(action)→任务(task)
        - 一个项目采用的过程可能与另一个项目采用的过程有很大的不同。
    - 组成：
        - 框架活动 （ppt16, 书p12）：沟通、策划、建模、构建、部署
        - 普适性活动（书P13）
- 软件工程实践
    - 实践精髓：P14
    - 通用原则：P14
- 关键点
    - **增量开发和敏捷开发的区别**
        - 粒度不同，每一个增量的时长和规模 > 每一个Scrum迭代
        - 周期要求不同，增量开发以任务（如里程碑）为界限，Scrum以时间（如2~4周）为界限
        - 活动进行方式方法不同，Scrum以简化为核心（弱化了需求）
        - 交付频率不同，增 > Scrum （2~4周）

ch2-过程模型

- 定义过程模型（P23）
  - 步骤：定义过程流→定义每个活动(activity)的行动(action)→定义每个行动的任务(task)
  - 五种框架活动（P23）
    - 沟通、策划、建模、构建、部署

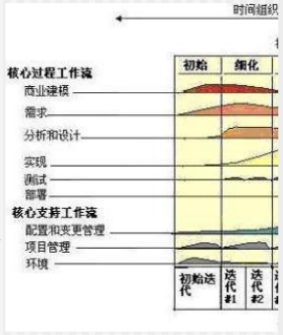
      沟通的目的：确定项目目标和了解需求
      策划的内容：资源、成本、进度估计和计划
      建模的本质：“做什么”，“怎么做”
    - 过程(process)→阶段(activity)→行动(action)→任务(task)
  - 四种过程流（书p24）
  - 任务集（书P24）
    - 里程碑：重要的时间节点。
- 惯用过程模型（P30）
  - 作用：提供有序的软件工程方法
  - 基本步骤：沟通、策划、建模、构建、部署
  - 瀑布模型（P30, ppt25）
    - 特点
      - 强调了每一阶段活动的**严格顺序**。
      - **推迟实现**：一个阶段没完成，将不进入到下一个阶段
      - **质量保证观点**：每个阶段均提供文档（里程碑），且需要通过技术评审，才能进行下一阶段的工作。
    - 适应情况：✓需求稳定的项目；✓开发团队对该应用领域非常熟悉
    - 问题（P31,ppt26）
  - 增量模型（P32, ppt27）
    - 概念（P31）：线性结构+并行结构
    - 优点：
      - 1.可提高对用户需求或市场需求的响应；
      - 2.增量开发在没有足够的实现完整需求的人员时特别有用；
      - 3.不确定的功能放在后面开发
    - 举例：ppt29
  - 演化模型（P33, ppt30）
    - 本质：通过迭代中一系列活动的重复应用，使软件持续改进
    - 1、原型开发范型
      - 原型类型

        - 抛弃型：通过原型工具快速构建假的界面，表明用户需求，如`Axure`，`墨刀`
        - 演化型：直接使用前端语言开发可优化迭代的界面
    - 迭代目标
        - 提高软件质量和长期可维护性
        - 使用更适合的操作系统和程序设计语言
    - 适合场景
        - 初步需求不明确的项目
    - 缺点
        - 更高的开发要求
        - 用户混淆原型系统和最终系统
        - 开发人员经常做出实现妥协，以使原型快速工作，如结构性较差，缺乏整体考虑
        - 需要和用户多次确认产品模型和需求，如果缺少客户的参与，原型开发容易失败
- 2、螺旋模型（P35， ppt34）
    - 结合了原型模型的迭代特征和瀑布模型的系统性和可控性，是一种风险驱动的过程模型
    - 概念和2个显著特点：P35
        - 最大的特点：具有**风险分析**，呈螺旋式开发
        - 基本特点（3点，P35-P36，ppt36）
    - 适合场景：大规模软件项目；高风险项目
- 并发模型（P36，PPT无）

- 专用过程模型（P38，PPT无）
- 统一过程，UP（P40, ppt37）
    - 项目周期的四个阶段
        - ppt:初始、细化、构造、交付
        - 书上是五个阶段
    - 特点
        - 每一个阶段由若干次迭代组成，只不过每一次迭代开发的侧重点不同（每个阶段集中力量做正确的事）。
        - 一个迭代构成一个完整的开发循环，产生一个可执行的产品版本，是最终产品的一个子集。
    - 初始阶段（ppt38）

在UP初始阶段的迭代中，项目组必须：

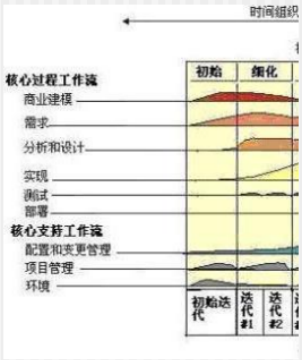
- ✓ 确定开发目标与范围
- ✓ 评估潜在的技术、商业风险
- ✓ 制定迭代实施计划
- ✓ 识别系统的关键用例(10%左右)
- ✓ 选择合适的软件架构
  - 例如编译软件适合的架构为管道-过滤器风格
- ✓ 如果需要构造原型，可以进行设计和实现



- 细化阶段（ppt39）

到了细化阶段的迭代，需要：

- ✓ 识别大部分用例？（至少80%）
- ✓ 通过多次迭代，每次选择当前最关注的用例进行体系结构设计，细化为对应的组件模型、实现模型、部署模型等。
- ✓ 核心是需求、分析和设计。
- ✓ 实现重要的部分用例，验证体系结构的稳定性。



- 构建阶段：核心是所有功能的实现、测试，为应用部署做好准备（ppt40）
- 交付阶段：核心是部署、系统上线、运行和维护（ppt40）
- UP裁剪

- UP是一个通用的过程模板，非常庞大。对不同的开发机构和项目，使用UP时需要裁剪，如：

1） 确定本项目需要哪些工作流。RUP的9个核心工作流并不总是需要的，可以取舍。

2） 确定每个工作流需要哪些制品。

3） 确定每个阶段内的迭代计划。规划RUP的4个阶段中每次迭代开发的内容。

ch3-敏捷开发

- 基本概念(ppt44)
  - 传统软件开发方法特点
  - “敏捷联盟”的价值观(P45)
  - 敏捷软件开发方法理念
  - Agile的典型方法
  - 敏捷软件开发方法特点（P47）
  - 敏捷12原则（P48）
- 极限编程（XP）
  - 4个框架活动：策划，设计，编码，测试（P49, ppt53）
    - 策划
      - 用户故事：P49（每一个用户故事描述了一个从用户角度出发的简单系统需求）
    - 设计(ppt55)
      - 保持简洁原则（KIS原则）
      - Spike solutions 探针方案
      - 重构
    - 编码（ppt59）
      - 先设计单元测试：从测试的角度来考 虑设计，考虑代码。
      - 结对编程
    - 测试
      - 测试自动化
      - 验收测试
  - 基本流程（PPT-49,50）
  - 现场客户：ppt61
  - 传统vs 敏捷 ppt62
- 敏捷DevOps开发方法 PPT420
  - 软件产业和交付模式发展趋势
    - 云成为软件的普遍承载方式
    - 新形势下企业面对多重挑战
    - 软件研发模式不断创新
  - 敏捷
    - 敏捷价值观 - 敏捷软件开发宣言 (1) (P45)
    - 敏捷原则- 敏捷软件开发宣言 (2） （P48）

    - 敏捷常用的工程方法
        - 全面视角的Scrum框架
        - 看板方法
- devops
    - 观念：DevOps打破开发与运维之间的孤岛
    - DevOps中一个完整的开发运维过程
    - DevOps生命周期过程
    -
- 敏捷和DevOps关系
    - DevOps覆盖端到端交付周期
- 华为云DevCloud HE2E DevOps框架及其主要服务

ch5-理解需求

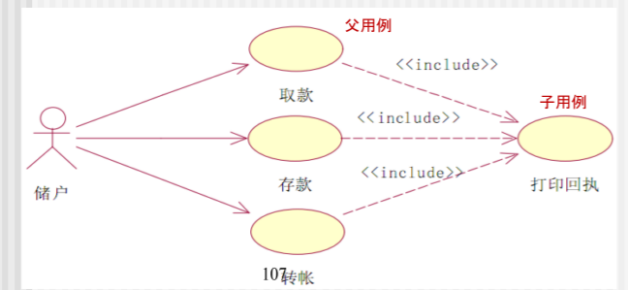
- 软件需求
  - 理解问题的需求是软件工程师所面对的最困难的任务之 一。（ppt64）
    - 客户不知道或无法全面的阐述需求；
    - 客户需求可能在项目实施过程中改变；
  - 软件需求：软件外部可见，软件所具有的、满足用户特点的功能及其他属性的集合（ppt67）
    - 做什么，而不是如何做
    - 清晰、简洁、一致、无二义性
  - 需求类型（ppt68）
    - 用户、视角不同，需求也必然不同
    - 🅰常见类型
      - 说法一：性能需求，可靠性需求，数据需求
      - 说法二：业务需求，用户需求，功能需求（也包括肺功能需求）
- 🅰需求过程：七项任务（ppt70，P74）

  起始，获取，细化，协商，规格说明，确认和管理
  - 起始（step：ppt71, 书P79→7.2建立根基）
  - 获取（points：ppt72）
    - 收集需求
      - 需求讨论会
      - 头脑风暴→适用于需求不确定的情况
      - 调查问卷→适合需求调研后期
      - 场景分析法
      - 实地考察（经验：ppt79）
      - 原型法
    - 质量功能部署QFD（P83）
    - 使用场景等剩余步骤（书P84）
  - 细化
    - 核心：开发一个精确的需求模型
    - 基本方法：ppt82
  - 协商
    - ppt86，具体做法见书P75
  - 规格说明（SRA）
    - 基本概念：ppt87

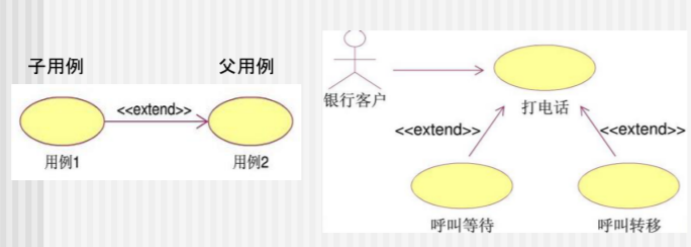
    - SRS(Software Requirement Specification)的作用 (ppt 88)
    - SRS主要内容(ppt 88)
        - 引言
        - 任务概述
        - 数据描述/数据要求
        - 功能需求
        - 性能需求
        - 运行需求/外部接口需求
        - 设计约束
        - 其他需求
- 确认
    - 参与者（ppt 90, P76）
    - 评审要点（ppt 90, P77(更全)）
    - 常见错误
        - 涉及到具体的技术说明
        - “各类文档” ——文档过多
        - 需求是否有归属，是否与其他需求有矛盾
        - 需求是否可实现、可测试?
- 需求管理
    - 基本概念：P77

ch6-需求建模：场景、信息与类分析OOA

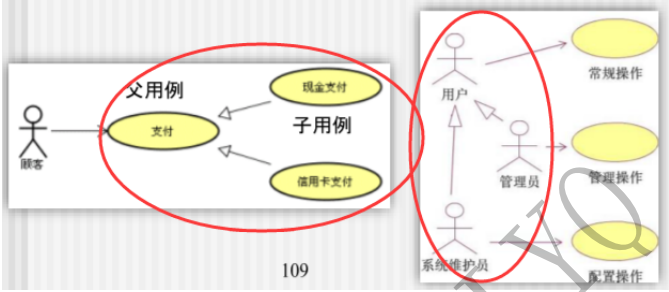
- 需求分析（ppt93）
  - 本质：通过建立概念化的分析模型，对收集的 需求进行提炼和审查。
  - 需求建模动作结果为以下一种或多种模型类型：书P96
    - 场景模型，面向类的模型，基于行为和模式的模型，数据模型，面向流的模型
- 分析方法
  - 结构分析法（Structured Analysis，SA）
  - 面向对象分析法（Object-oriented Analysis，OOA）
    - 基本概念
      - 现实世界（问题空间）：实体及其相互关系
      - 软件系统（解空间）：对象及其关系
      - 图像表示工具：统一建模语言，（Unified Modeling Language，UML）
    - 基本步骤
      - 用例模型 → 干什么 what ➡️ 功能需求, 用例图
      - 领域（静态）模型 → 有什么 who ➡️ 类图
      - 交互（动态）模型 → 怎么做 how ➡️状态图和时序图
- OOA
  - OOA & UML(Unified Modeling Language)
  - 分类：用例模型、（静态）领域 模型和（动态）行为模型
  - 分析过程：定义用例(**用例**) → 定义领域模型(**类图**) → 定义动态交互图(**时序图、状态图**)（ppt98）
  - 分析实例（ppt99）
- OOA-step1:基于场景的方法
  - 用例模型 人和事 （ppt102）
    - 用例表示从执行者的角度观察到系统的功能 和外部行为。
    - 主要描述用户的功能需求，强调谁在使用系统，系统可以实现哪些功能目标
      分析每个参与者如何参与系统
    - 如何识别用例：ppt103
    - 用例四元素：执行者、用例，执行者与用例间关联、用例间关系
      - 《include》关系 （ppt107）



- 《extend》关系（ppt108）



- 泛化关系（generalization）：父用例或者父参与者（ppt109）



- 用例场景描述（ppt110）
  - 用例场景是将用例发生的各种场景描述出 来，表示参与者与系统交互时双方的行为，即参与者做什么，系统做什么反应
  - 描述工具：用例模板或活动图
    - 用例模板（ppt111）
      - 内容：用例名、参与者、前置条件、后置条件、**事件流**等
      - 示例：（ppt112）
    - 活动图 事件的流程图 示例（ppt113）
      - 人+泳道，书P105
- OOA-step2:数据模型
  - 特点：ppt114
  - 数据对象
    - 概念：ppt115， P109
    - 数据对象和面向对象类的关系

INFO

数据对象和面向对象类——它们是同一个东西吗?

当讨论数据对象时会出现一个常见的问题：数据对象和面向对象[⊖]的类是同一个东西吗？答案是否定的。

数据对象定义了一个复合的数据项，也就是说合并一组独立的数据项（属性）并为数据项集合命名（数据对象名）。

一个面向对象类封装了数据属性，但对这些属性所定义数据的操作（方法）进行合并。另外，类的定义暗示了一个全面的基础设施，该基础设施是面向对象软件工程方法的一部分。类之间通过消息通信，它可以按层次关系组织，并为某个类的一个实例这样的对象提供继承特性。

- 数据属性 ppt116
- 关系 ppt117
  - 数据对象可以以多种不同的方式与另一个数据对象连接。
  - 实体-关系图（ERD）（示例 ppt118）

INFO

实体-关系图

对象/关系对是数据模型的基石。可以使用实体-关系图（ERD）[⊖]图形化地表示这些对象/关系对。ERD最初是由Peter Chen[Che77]为关系数据库系统设计提出的，并由其他人进行了扩展。ERD标识了一组基本元素：数据对象、属性、关系以及各种类型的指示符。使用ERD的主要目的是表示数据对象及其关系。

已经介绍过基本的ERD符号，用带标记的矩形表示数据对象，用带标记的线连接对象表示关系。在ERD的某些变形中，连接线包含一个带有关系标记的菱形。使用各种指示基数和模态的特殊符号来建立[⊖]数据对象和关系的连接。关于数据建模和实体关系图的更多信息，感兴趣的读者可以参考[Hob06]或[Sim05]。

- 数据建模总结

SOFTWARE TOOLS

数据建模

目的：数据建模工具为软件工程师提供表现数据对象、数据对象的特点和数据对象的关系的能力。主要用于大型数据库应用系统和其他信息系统项目，数据建模工具以自动化的方式创建全面的实体-关系图、数据对象字典以及相关模型。

机制：该类型的工具帮助用户描述数据对象及其关系。在某些情况下，工具使用ERD符号；有些情况下工具使用其他一些原理为关系建模。该类工具往往用来作数据库设计，还可通过为公共数据库管理系统（DBMS）生成数据库模式帮助创建数据库模型。

代表性工具：[⊖]

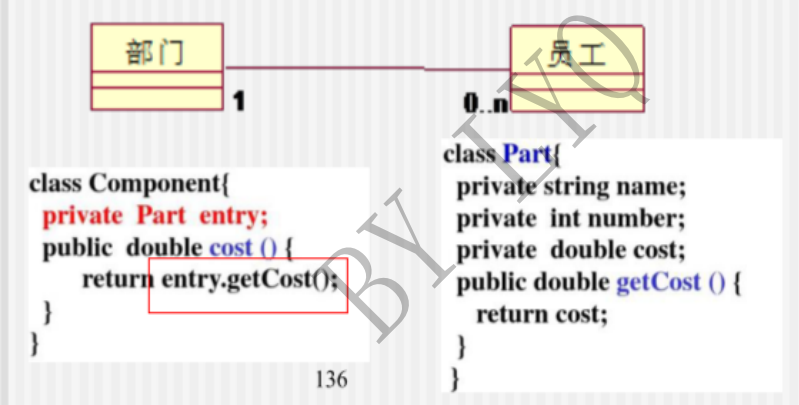
- **【关键】信息是否需要持久存储**

- OOA-step3:基于类的模型
  - 领域模型分析：以用例模型作为输入，将系统分解为 相互协作的概念(分析)类。
    - 概念分析类（ppt125）
      - 分析类是概念层次的东西，源于**问题空间**，与实现 方式无关。(ppt131)
  - 模型元素：(p108，ppt126)
    - 类和对象、属性、操作、CRC、协作图和包
  - 目标：对领域内的概念类进行分析，并抽 象出对应的类及其关系等。

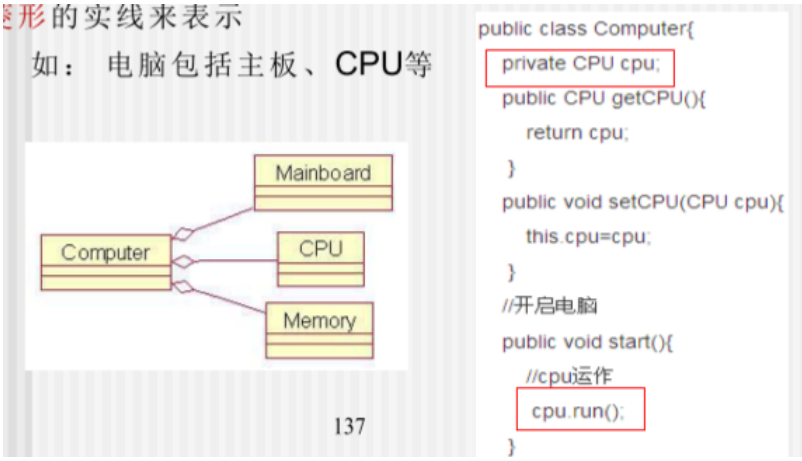
- 步骤（ppt127开始）
  - 从用例场景中寻找概念类/分析类（书p99：9.1 识别分析类，ppt128）
    - 2个步骤、7个表现方式、6个选择特征
  - 细化概念类，识别其类型（如边界类、实体类、控制类等）（ppt132，书P114）
  - 为概念类添加关联和属性 （ppt133）
    - 属性的概念（p111）
    - 定义操作的方法 ppt134
- 注意
  - （1）识别分析类

    最稳定，问题定了，分析了就定了

    系统管理持久化信息或贯穿应用的数据封装
  - （2）定义属性
  - （3）定义操作
  - （4）定义关系（**ppt136**，类图实例：ppt141, 书p116)
    - 关联关系，大类以小类为属性变量（ppt136）



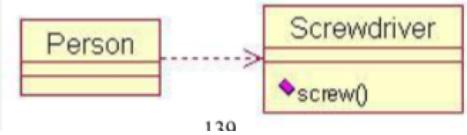
    - 特殊的关联关系：聚合关系。整体与部分（可分开，即Part为可以单独存在的实体）。



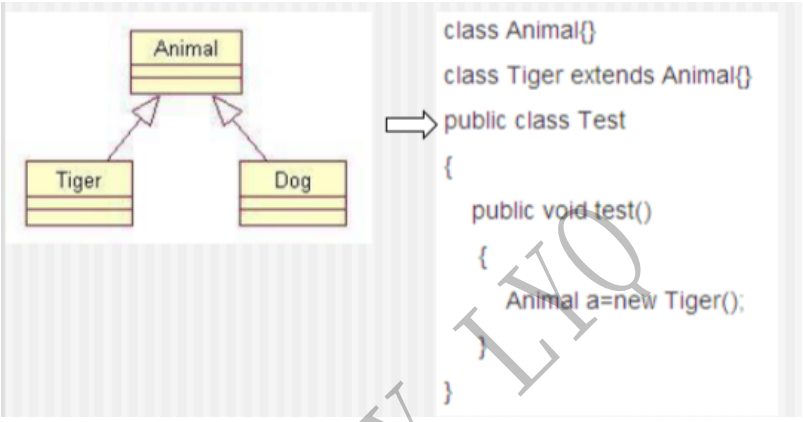
    - 特殊的关联关系：组合关系。整体与部分（不可分开），有相同的生命周期。

- 依赖关系

```
比如你要拧螺丝，要借助(也就是依赖)螺丝刀(Screwdriver)来完成
public class Person{
   public void screw(Screwdriver screwdriver){   /** 拧螺丝 */
     screwdriver.screw();
   }
}
```



  - 即一个类的实现需要另一个类的协助，多数情况下，依赖关系表现在某个类的方法使用另一个类的对象作为参数。
  - C++的前向声明（即依赖的那个类必须在其之前进行声明）
- 泛化关系



  - 表示类与类、接口与接口之间的继承关系

- CRC模型（Class-responsibility-collaborator，类-职责-协作者）

  作用：帮助补充、完善、优化分析类（属性、行为）

  - 职责的概念+分配职责的指导原则（ppt143,p115）
  - 协作 ppt144,p115
    - 类实现职责的两种方式
    - 协作的特点
    - 3种通用关系（书p116）
- 包管理 package
  - 好处和原则：ppt146
  - 包之间的主要关系是依赖关系。意味着两个 Package内的元素 之间存在着一个或多个依赖。
- OOA-step4:行为模型 （书p122）
  - 基本概念 ppt150
    - 行为模型显示了软件如何对外部事件或激励做出响应
    - 描述一个用例中，系统在接收到特定事件后，所涉及的对象以及这些对象之间的交互（消息传递）过程。

    - 主要包括：时序图、协作图、状态图
- 时序图
    - 建模步骤 ppt151
    - 系统时序图(一般不考)
        - 系统时序图中将系统作为一个黑盒子
        - 示例：ppt152
    - 对象时序图
        - 每个标注说明：ppt153
        - 主要步骤：ppt154
        - 为分析类分配职责 ppt157
            - 根据消息传递过程，将职责分配到相应的分析类中。
            - 到设计阶段，职责将被进一步细化，最终对应设计类中的成员方法。
            - 🅰消息和职责的关系：ppt157
        - 示例：ppt159
- 协作图
    - 表达对象间的关联关系（协作关系）
- 状态图

    - 基本概念：描述了一个对象生命周期内的所有可能状态，以及由于事件的发生而引起状态之间的转移。
    - 主要作用：帮助发现和定义**某个特定对象**的行为。
        - 针对**某个特定对象**，只关心单个对象及其状态改变的触发事件
    - 状态图元素 ppt164
    - 特殊状态：一个起始状态、多个终止状态、组合状态。
- 行为模型比较
    - 🅰状态图 vs 时序图 ppt163
    - 🅰状态图 vs 时序图 ppt163
- OOA建模总结（**详见ppt169**）
    - 用例建模
    - 静态建模（初步确定分析类）
    - 动态建模（确定类的行为/职责）
    - 完善分析类图

- 建立包图（可选）
- 建立数据模型，连接数据库 （可选）
- 状态图（解决复杂生命周期模型）（可选）

ch7-需求建模：面向流的建模SA

- **8th没有这章**
  - SA：面向功能,把系统看成一组功能，它将数据和转换数据的流程视为单独的实体
  - OAA：类是系统的基础，系统由类组成
- 结构化分析方法（Structured analysis） ppt172
  - **SA，又称面向数据流的分析方法**
  - 原理 ppt172
  - 四种分析模型：数据流图（DFD，功能视角），数据字典，实体关系图（ER，数据视角），状态迁移图（行为视角）→四者的关系ppt172
- 数据流图（Data Flow Diagram） ppt173
  - 数据流图将系统看作由数据流联系起来的各种功能的组合
    - 分析问题域中数据如何流动
    - 在各个流动过程中的加工、变化
  - 四种符号 （相关解释：ppt174起）
    - 外部实体、加工、数据流、数据存储
  - 建模方法 ppt179
    - 方法：见ppt
    - 注意事项：见ppt
    - 顶层数据流图(含义+作用：ppt181)
    - DFD子图分解（方法+注意事项：ppt182）
    - 保持父图与子图平衡
    - 看图方法：三次遍历法
      - 1.数据流方向，画出主要业务流程
      - 2.异常情况的分支处理
      - 3.依次确认需要存储的数据是否进行有效加工过程
  - 教材购销系统（ppt184）
- 数据字典
  - 用于精确严格地定义每一个与系统相关的数据元素（包括数据流、数据存储和数据项），并以字典顺序将它们组织起来，使得用户和分析员有共同的理解 ==> DFD的补充和完善
  - 四要素
    - 数据流、数据项、数据存储、加工处理（包括输入数据、输出数据、加工逻辑）
  - 内容（不知道在哪儿抄的）
    - 1.名称：数据对象或控制项，数据存储或外部实体名字

    - 2.别名或编号
    - 3.分类：数据对象，加工，数据流，数据文件，外部实体，控制项（事件/状态）
    - 4.描述：描述内容或数据结构等
    - 5.何处使用：使用该词条（数据或控制项）的加工
    - 6.注释：数据量，峰值，限制，组织方式等
- 实体关系图（ER）→数据视角
- 状态迁移图→ 行为视角 / 控制流模型 ppt191
    - 特点+适用情况 ppt191
    - 一种控制流模型：State Diagram（状态迁移图）（ppt192）
    - 状态迁移图 vs 状态图
        - 状态迁移图对象是系统
        - 状态图对象是类
- 数据流和用例图的关系
    - 用例图强调任务的include、extend、abstract的关系；
    - 数据流图强调任务的先后顺序（任务流程）

ch8-设计概念

- 设计(ppt194)
  - 🅰分析 vs 设计 (ppt194)
  - 设计举例
- 软件设计 ppt195
  - 定义 （ppt195，书p134）
  - 分两阶段（体系结构设计、构建设计或详细设计）
  - 设计的重要性 （ppt197,p133）
  - 良好设计的三大特征 ppt199,**p134**
  - 设计的工作集（ppt200,P133）
    - 数据/类设计：将分析类模型转化为设计类以及软件所需要的数据结构（即书上第九章）
    - 体系结构设计：定义软件的主要元素（构件）以及元素之间的联系
    - 接口设计：定义软件与协作系统之间、软件与用户之间的通信
    - 构件设计或详细设计：定义软件元素（构件）的内部细节，如内部数据结构、算法等
    - 部署设计：定义软件元素在物理拓扑结构中的分布
  - 需求建模 to 设计模型 (ppt201, 书p132)

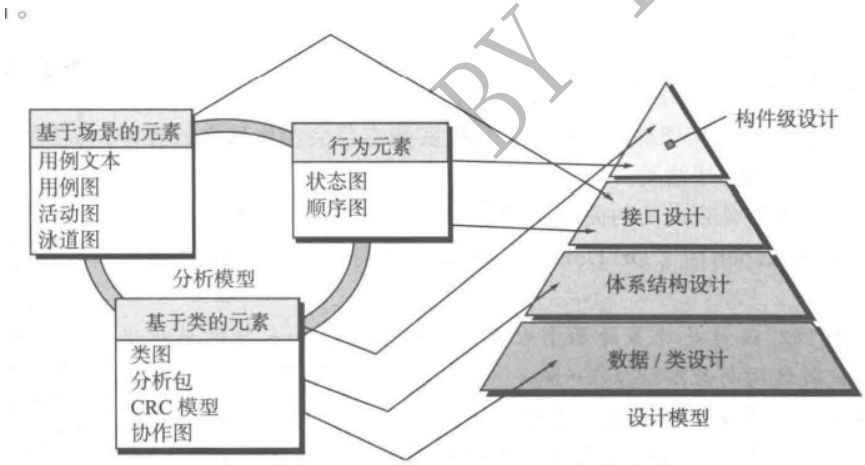


图 11-1　从需求模型到设计模型的转换

    - 举例 ppt202
  - 软件质量指导原则：p134
  - FURPS质量属性（p135）
- 设计概念**（注意ppt和书结合来看，汇总ppt205， 书P137）**
  - 抽象 — 数据、程序、控制
    - 数据抽象、结构抽象（p137，ppt208）
    - 抽象的本质：强调关键特征，忽略无关细节
  - 体系结构 — 软件的整体结构 ppt208

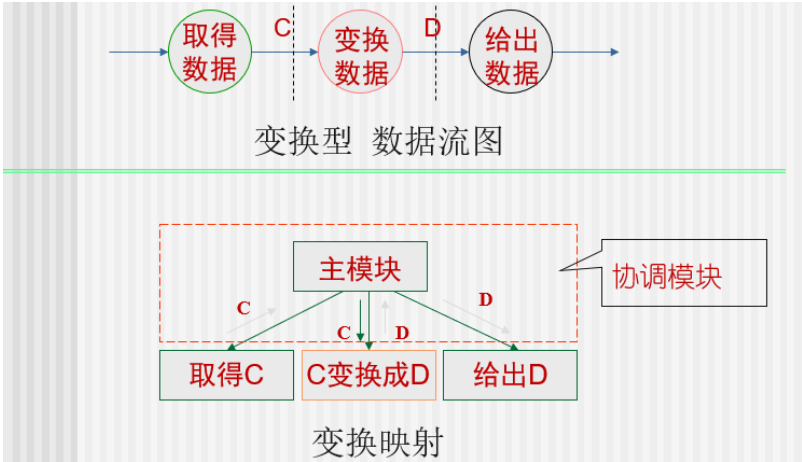
- 关注点分离 — 如果将任何复杂问题细分为多个部分，则可以更轻松地处理
  - 分治
- 模块化 — 数据和功能的划分 ppt209
  - 分治
  - 关注点分离最常见的表现
- 信息隐蔽 — 受控接口
  - 模块内部修改对外部影响最小
- 🅰功能独立—功能独立设计，耦合度低
  - 每个模块尽可能专一独立，避免彼此交互过多
  - 开发的独立和并行性
  - 🅰高内聚，低耦合
    - 内聚：模块内相关功能的强度，内聚性强的模块通常只完成一件事情
    - 耦合：显示了模块之间 的相互依赖性，取决于模块之间接口的复杂性、引用方式、传递数据的复杂性
  - 优势 ppt213
- 求精 — 为所有抽象详细阐述
- 方面 — 一种了解全局需求如何影响设计的机制 ppt214
  - AOP（面向方面编程）：核心内容剖开封装的对象内部，将那些影响了多个类的公共行为 封装到一个可重用模块中，并将其命名为Aspect，即方面。
  - AOP横切关注点 → OOP（纵向关注点）的补充和完善
  - Spring AOP
- 重构 — 一种简化设计的重组技术
  - 接口不改变，内部结构优化
- 设计类 — 提供设计详细信息，使分析类得以实现**(ppt218)**
  - 分析类面向业务领域，设计类面向实现
  - 定义方式
    - 为分析类提供细节设计：成员属性和成员方法
    - 创建新的设计类，该设计类实现了软件的 基础设施以支持业务解决方案。
  - 良好设计类的四个特征（p143）
    - 完整性与充分性、原始性、高内聚性、低耦合性

ch9-体系结构设计

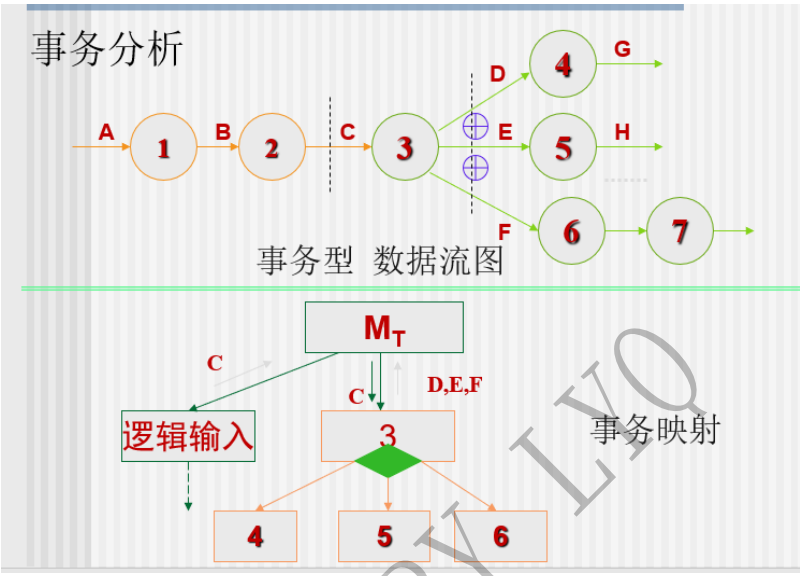
- 1、理解体系结构及设计
  - （1）什么是软件体系结构 ppt224
    - 定义：软件体系结构是软件的一种划分形式，从较高层次 定义系统的构件、构件之间的连接，以及由构件与构件连接形成的拓扑结构。
    - 软件体系结构设计质量评估：结构稳定性、可扩展性和可复用性
    - 从计算机看“体系结构” ppt225
      - 软件系统划分
        - 处理数据存储、处理业务逻辑、处理页面交互、处理安全
      - 硬件系统划分
        - 控制器、运算器、内存、外存、输入设备
        - 硬件模块之间的连接关系
    - 软件体系结构实例 ppt226+p154
  - （2）为什么要做体系结构设计 ppt227
  - （3）做好软件体系结构设计的关键决策 ppt229
- 2、体系结构风格
  - 体系结构模式 vs 体系结构风格 p157
  - 概念 p156+ppt231
  - 简单分类：⬛ 数据中心体系结构 ⬛ 管道-过滤器体系结构 ⬛ 调用返回体系结构 ⬛ 面向对象体系结构 ⬛ 层次化体系结构（ppt231）
    - 数据中心体系结构
    - 管道-过滤器体系结构
    - 调用-返回体系结构
      - 主程序/子程序结构：层级关系 call return
      - 远程过程调用：主程序/子程序中的构件分布在网络中的多台计算机，如分布式计算系统Hadoop
      - 缺点 ppt237
    - 面向对象体系结构
      - 构造单元：类和对象
      - 连接：基于消息机制
      - 拓扑结构：平级（无主次）
    - 消息总线体系结构
      - 不直接调用构建，而是广播一个或多个事件，构件进行事件监听并自动调用对应事件注册的所有构建

        - 消息中间件：
            - 场景描述 ppt240
            - 应用：系统解耦（降低系统的耦合性）、异步、削峰
            - 优点
    - 层次化体系结构
        - 基本概念
    - 客户-服务器体系结构
        - 客户机 `front-end`：业务逻辑、服务器通讯
        - 服务器 `back-end`：与客户机通讯接口、业务逻辑、数据管理
        - 演变
    - MVC体系结构
        - Model-View-Controller
    - 混合风格
        - 大规模系统往往包括多种体系结构风格
- 3、体系结构描述 （ppt247+书p154）
    - 概念
    - 逻辑架构→功能
    - 物理架构→基础设施和拓扑结构
    - 开发架构→程序包
- 4、面向对象设计方法 ppt249
    - 概念 (**书p158**)
        - 识别系统外部环境和交互特性
        - 定义原型
        - 设计顶层构件结构
        - 细化构件结构
- 5、面向对象设计应用
- 6、结构化设计方法 ppt258
    - 目标：按照DFD的不同类型，通过对系统中模块的 合理划分，得到软件体系结构图
    - 设计概念：ppt259
    - 分析方法

      数据流图分类
        - 变换分析：一个大的接口，针对某一数据，进行特定的变换（内部协调处理）



- 事务分析→不同的数据选择不同的操作（外部逻辑判断分流）



  - 创建顶层控制结构
  - 对输出分支进行细化

- 基本步骤：ppt263
  - (1) 分析确定数据流图类型。
  - (2) 标识输入、输出流边界。
  - (3) 依据变化或事务分析规则，将DFD图映射为顶层和第一层模块结构。
  - (4) 对第一层中的模块继续细化和分解。
  - (5) 利用启发式规则优化初始系统结构图。
- 变换分析实例 ppt264
- 事务分析实例 ppt268

ch10-构件设计

- 1、构件设计基本概念 （ppt272）
  - 基本概念
  - 构件设计 vs 体系结构设计
  - 1）什么是构件
  - 2）结构化视角下构件设计
    - 功能模块
  - 3）面向对象视角下构件设计
    - 类
  - 4）面向对象构件特征
    - 构件接口与实现需要严格分离
    - 基于设计框架，一个构件需要分解为多个设计类。
  - 5）面向对象构件的4个关键问题
- 2、构件设计指导原则 （ppt279，书p180）
  - 1）开闭原则（OCP）
    - 对扩张开放，对修改关闭
    - 实现途径：构建子类
  - 2）LSP: Liskov Substitution Principle
    - 子类可以替换父类，而不影响功能的使用。
  - 3）依赖倒置原则：Dependency Inversion Principle（DIP）
    - 依赖于抽象，而非具体实现。
  - 4）接口分离原则 Interface Segregation Principle（ISP）
    - 多个客户专用接口比一个通用接口要好
    - 即：接口尽量细化，同时接口中的方法尽量少
- 3、面向对象构件设计过程 （ppt288,书p185）
  - ppt 五个步骤
  - 书 7个步骤
- 4、构件设计实例 ppt297
- 5、传统构件设计过程
  - ⊠ 确定每个模块的算法，用工具描述算法逻辑。
  - ⊠ 确定每一模块的数据结构。
  - ⊠ 确定模块接口细节。
  - 常用算法描述工具

- 程序流程图
- 问题分析图(Program Analysis Diagram, PAD)AAA ppt301
- 盒图(N-S图)AAA ppt303
- 伪代码(PDL)

ch11-用户界面设计

- 为什么需要界面设计（ppt305）
- 三条黄金规则（ppt305，书P198→之后有每一条的设计原则）
  - 把控制权交给用户
    - PPT5条设计原则，6条设计原则
  - 减轻用户的记忆负担
    - 5条设计原则
  - 保持界面一致
    - PPT2条设计原则，3条设计原则
- 四条补充规则（ppt318）
  - 反馈、宽容性、个性化、美观性
- 用户界面的设计 （ppt320,建议结合书来看）
  - **分析步骤P203**
    - 用户分析（交互的人） p203
    - 任务分析和建模（执行的任务） p204
    - 呈现内容分析 p207
    - 执行环境 p207
  - **设计步骤P208**

    交互设计和视觉设计
    - 1）交互设计
      - 交互流程（四步，见P208，ppt326）
        - 定义界面对象和动作
        - 导致界面发生变化的事件
        - 每个状态的表示形式
        - 如何向用户解释界面
      - 设计考虑（p210）
        - 响应时间、帮助措施、错误处理、菜单和命令标记、应用的可访问性、国际化
    - 2）视觉设计
      - 概念：针对眼睛的主观形式的表现手段和 结果。
      - 设计考虑（ppt329）
        - 主要考虑界面的布局、数据的呈现方式、色彩搭配 等
      - 设计工具：visio，axure
  - **设计评估（ppt336,P212）**

1. 系统及其界面的需求模型或书面规格说明的长度和复杂性在一定程度上体现了用户学习系统的难度。
2. 指定用户任务的个数以及每个任务动作的平均数在一定程度上体现了系统的交互时间和系统的总体效率。
3. 设计模型中动作、任务和系统状态的数量体现了用户学习系统时所要记忆内容的多少。
4. 界面风格、帮助设施和错误处理协议在一定程度上体现了界面的复杂度和用户的接受程度。

ch14~ch16

- **ch14-质量概念**
  - 软件质量定义：ppt338
  - McCall质量维度 （ppt339, 书p219）

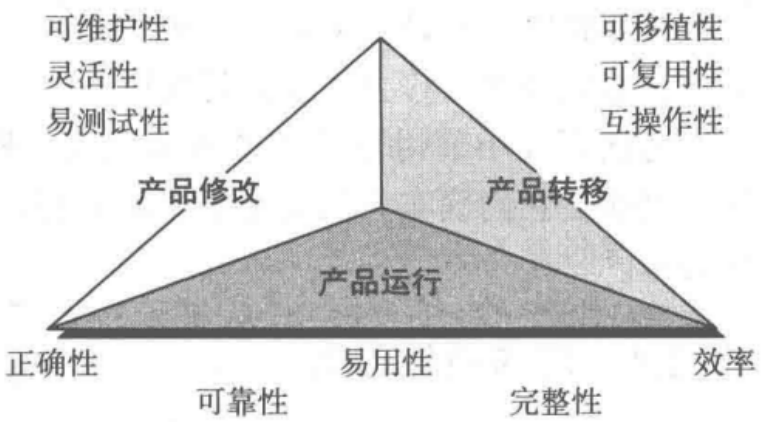


图 15-1　McCall 的软件质量因素

  - 质量因素中的冲突 ppt340
  - 质量模型——国标GB/T ppt341
  - 质量成本（ppt342, 书p223）
    - 预防费用成本
    - 评估成本
    - 过程失效成本
    - 外部失效成本
  - 关键成功因素 **(ppt344, p226)**
    - 软件工程方法
    - 项目管理技术
    - 质量控制
    - 质量保证
- ch15-评审技术 （8th没有这章节）
  - 质量控制（Quality Control， QC）
    - 基本概念：ppt346
    - 两种实现方式：软件测试、软件评审
  - 评审
    - 基本概念 （ppt347）
    - 价值（ppt348）

- 分类
  - 非正式评审 （ppt350）
    - 优缺点
    - 评审方式分类（desk check, 结对编程，walk through走查，轮查）
  - 正式评审（ppt352）
    - 基本概念
    - 评审参与者：生产者、审阅负责人、审阅者、记录者
    - 评审流程
    - 评审报告

      在FTR期间，由一名评审员（记录员）主动记录所有提出的问题。在评审会议结束时要对这些问题进行汇总，并生成一份“评审问题清单”。此外，还要完成一份“正式技术评审总结报告”。评审总结报告中要回答以下3个问题：

      1. 评审的产品是什么？
      2. 谁参与了评审？
      3. 发现的问题和结论是什么？

      评审总结报告通常只是一页纸的形式（可能还有附件）。它是项目历史记录的一部分，有可能将其分发给项目负责人和其他感兴趣的参与方。

      评审问题清单有两个作用：（1）标识产品中存在问题的区域；（2）作为行动条目检查单以指导开发人员进行改正。通常将评审问题清单附在总结报告的后面。

- ch16-软件质量保证（ppt356）
  - 软件质量保证（Software Quality Assurance， SQA）
    - 基本概念
    - 主要职责
  - QA vs QC （ppt357）
  - 软件质量保证活动（6点，ppt358，详见书p232）

ch17-软件测试策略🅰️

- 测试
  - 概念：ppt363
  - 测试目的：书p246
- 测试策略
  - 一般特征（5点，ppt364, P247→书上更详细）
  - 注意事项 （ppt365）
    - 测试不能取代评审
    - 测试人员：开发者和独立测试人员的优缺点分析（ppt366）
    - 在项目确定时就需要制定测试计划，开始软件测试
      - 需求分析→测试计划→测试设计→测试环境 搭建→测试执行→测试记录。
- 测试进程（ppt368， P249）
  - 我们从"小测试"开始，向"大测试"迈进
  - 宏观步骤：单元测试→集成测试→确认测试→系统测试
  - 细分
    - 传统软件测试
    - 面向对象软件测试
- 传统软件测试（conventional software）（ppt369，书P251）
  - 单元测试——验证开发人员编写的模块或单元是否符合预期
    - 单元测试问题（书P252）
      - 接口、局部数据结构、边界条件、独立路径、错误处理路径
    - 单元测试过程（ppt373, 书p252）
      - **驱动模块**：用来模拟被测模块的上级调用模块。
      - **桩模块**：模拟被测试模块所调用的模块，返回被测模块所需的 信息。
  - 集成测试——调用接口是否出错 （ppt374，书p253）
  - 非增量式集成（大爆炸/一步到位方法）

    优缺点分析见P253
    - 增量式集成（ppt375）
      - 自顶向下集成（具体步骤见书P254）
      - 自底向上集成（具体步骤见书P255）
      - 回归测试 ppt378
        - 定义和优缺点（书P255, ppt378）
        - 手工进行：重新执行所有测试用例的子集

        - 自动进行：捕获-回放工具（如WinRunner）能够捕获在测试过程中 传递给软件的输入数据，生成执行的脚本，并且在回归测试中，重复执行该过程。
        - 回归测试套件（P255）
    - 冒烟测试
        - 概念和目的：（ppt379）
        - 测试方法：P256
        - 好处： P256
- 面向对象的测试（OO software）（ppt380,书P257）
    - 单元测试
        - 和传统的单元测试的异同：P257
    - 集成测试
        - 基于类之间的协作关系进行集成
        - 两种策略（ppt380，书P257）
            - 基于线程的测试
            - 基于使用的测试
        - **驱动模块和桩模块，和传统集成的变化（P258）**
- 确认测试（ppt381，书p258）
    - 概念：检验所开发的软件是否满足需求规格 说明中的各种功能和性能需求，以及软件配置是否完全和正确。
        - 侧重于需求级的错误
        - 确认测试方法的基础：需求规格说明书中的确认准则
    - 分类：

      α测试是由用户在开发环境下进行的测试；
      β测试是由软件的多个用户在实际使用环境下进行的测试
- 系统性测试（ppt382，书P260）
    - 概念：将软件与其它系统部分（如外部系统，硬件设计 等）结合进行的一系列集成测试和确认测试。
        - 关注非功能性需求
    - 系统测试方法
        - 恢复测试
            - 强制软件以多种方式失败，并验证恢复是否正确执行
        - 安全测试
        - 压力测试
        - 性能测试
- 调试过程（ppt387，书P264）

- 调试目标：定位错误的原因和位置
- 测试策略（**蛮干法、回溯法、原因排除法**）
  - 蛮干法：逐步调试
  - 回溯法：根据错误征兆回溯定位
  - 原因排除法：
    - 归纳法：根据线索导出假设→证明假设
    - 演绎法：假设错误→验证错误

- **关键点**
  - **回归测试是减少“副效应”的重要方法**

ch18-测试传统的应用软件

- 关于测试的2个问题（ppt393）
  - （1）经过测试没有发现错误的软件就一定正确 吗?
    - 错误。测试只能证明软件是有错的,但不能证明软件是 没有错误。
  - （2）可以通过穷举方式测试软件吗?
- 好的测试的特征（书P269）
- 测试用例设计方法 （ppt396, 书p270）
  - 了解已设计的产品要完成的指定功能 → 黑盒法
  - 了解产品的内部工作情况 → 白盒法
- 白盒测试（P397,书p271）
  - 概念
    - 按照程序内部逻辑结构，设计测试数据并完成测试的一种测试方法。
    - 目标是使得程序中各元素（如语句、判定）尽可能被覆 盖到。
    - 白盒测试一般在测试过程的早期执行。
  - 方法
    - 语句覆盖法（ppt398）
    - 基本路径测试（ppt399,书p271）
      - 基本集合
        - 独立路径（概念： ppt402， P273）
      - 环域复杂度 = |基本集合| = Edge-Node+2 = 判定Node+1
      - 测试步骤 （ppt399, P274）
        - 使用设计或代码作为基础，绘制相应的流图。
        - 确定生成的流图的循环复杂性。
        - 确定独立路径的基本集合。
        - 准备测试用例，强制执行基本集合的每个路径
    - 控制结构测试 (ppt404, P276)
      - 条件测试
      - 数据流测试
        - 变量的定义-使用链条
        - 对于关键变量适用
      - 循环测试
        - 简单循环、嵌套循环、串接循环、非结构循环
        - 具体的步骤见书（p276）

- 黑盒法（ppt409，P277）
  - 概念
    - 测试者把测试对象看做一个黑盒子，测试人员 完全不考虑程序内部的逻辑结构，只依据需求规格说明书，检查程序的功能是否符合它的功能说明。
    - 一般用于测试的后期阶段
  - 测试数据设计
    - 等价类划分（ppt410,p278）
      - 将程序的输入域划分为若干个数 据类，从中生成测试用例。
      - 2个指导原则（ppt412）
      - 4个分情况讨论（书）
    - 边界值分析法（Boundary Value Analysis, BVA）
      - 边界值应该尽可能选择等于边界，略小于边界， 略大于边界的值
      - 边界值法可与等价类划分法结合，作为对其的 一种补充。
      - 4个分情况讨论（书）
  - 测试方法
    - 🅰️正交数组测试（Orthogonal Array Testing, OAT） ppt415
      - 当输入参数的数量较少且每个参数可能采用的值有明确界限时使用
      - **均匀分散、整齐可比**
      - $\sum_{i=1}^{n}(x_i-1)+1$，其中共有n个因子，第i个因子的取值数为x_i
      - 案例分析🅰️🅰️
    - 基于模型的测试（Model-Based Testing, MBT）（ppt481,p278)
      - 测试用例设计基础：基于模型的测试技术使用UML状态图
      - 5个步骤（见书P278）
- **关键点**
  - **只有在构件级设计（或源代码）存在之后，才设计白盒测试**
  - **可以描述程序逻辑的图：盒图（NS图）、问题分析图（PAD图）、流图（F图）**
  - **流图的基本集合不是唯一的**