软件工程
第五章 OO分析与设计
5-3 软件设计概论

刘铭

Monday, June 10, 2024

主要内容

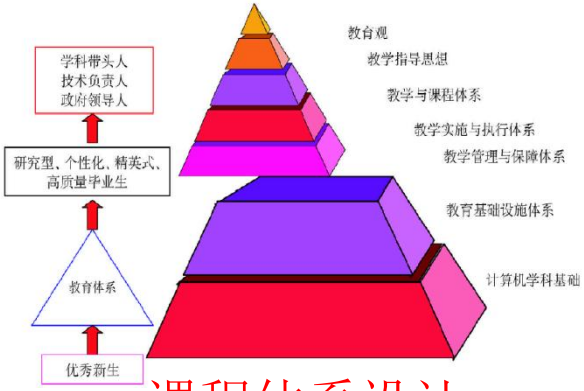
- 1.软件设计的背景
- 2.软件设计中的核心概念
- 3.软件设计模型

汽车设计

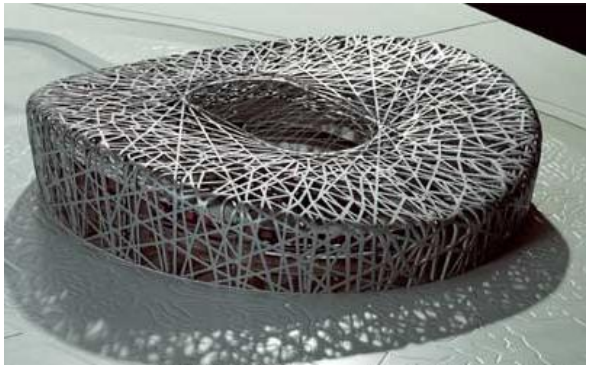
”？

服装设计

机械设计

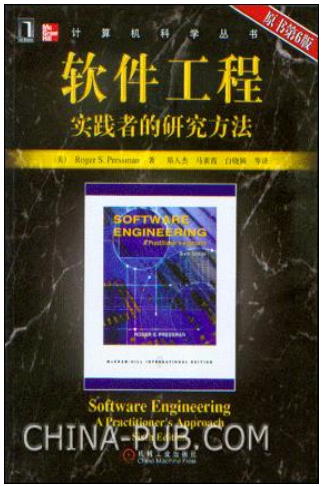


课程体系设计

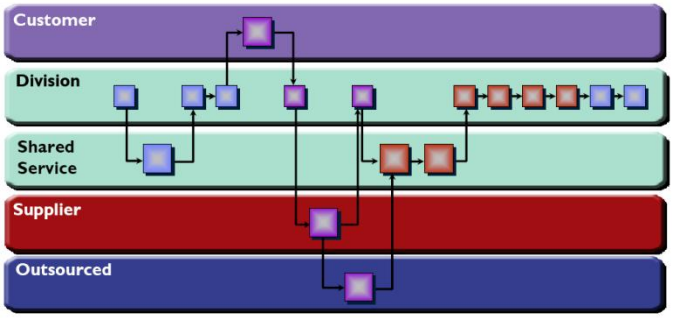


建筑设计

平面广告设计



封面设计



业务流程设计

设计=天才+创造力

- 每个工程师都希望做设计工作，因为这里有“创造性”——客户需求、业务要求和技术限制都在某个产品或系统中得到集中的体现。
- “设计”是最充分展现工程师个人价值的工作。

程序员→工程师





“设计”的本质

- 什么是设计？设计是你身处两个世界——技术世界和人类的目标世界——而你尝试将这两个世界结合在一起。

——Mitch Kapor(Lotus 1-2-3创始人), 《软件设计宣言》

Bill Gates, Mitch Kapor, Fred Gibbons, 1984

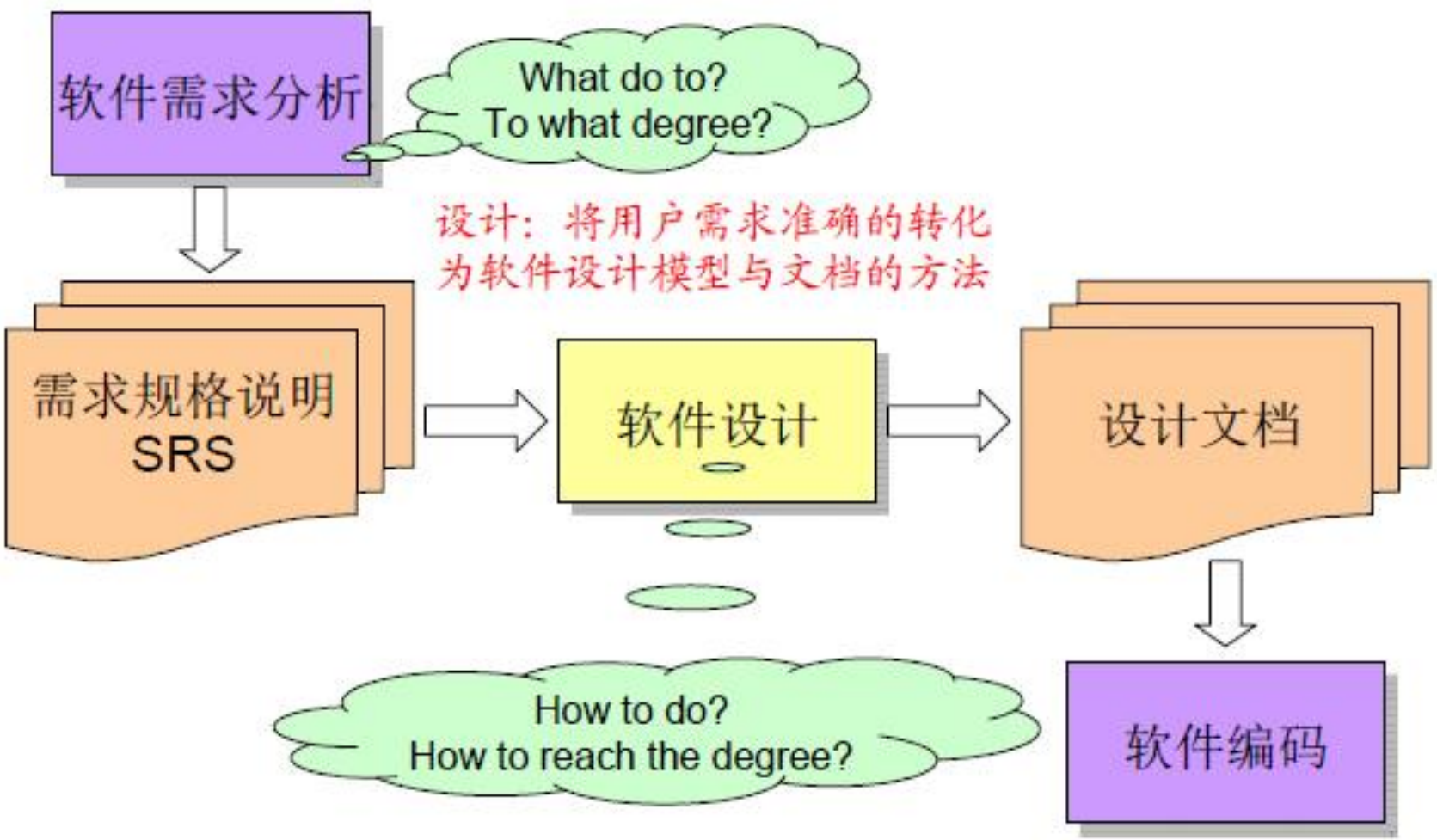
“软件设计”的定义

- **软件设计**：为问题域的外部可见行为的规约增添实际的计算机系统实现所需的细节，包括关于人机交互、任务管理和数据管理的细节。

——Coad/Yourdon

- 关于“软件设计”的几个小例子：
  - 需求1：教学秘书需要将学生的综合成绩按高到低进行排序
  - 设计1：void OrderScores (struct * scores[ ]) { 冒泡排序算法, step1; step2;… }
  - 需求2：数据字典“销售订单”
  - 设计2：关系数据表Order(ID, Date, Customer, …),
    OrderItem(No, PROD, QUANTITY, ..)
  - 需求3：“查询满足条件的图书”
  - 设计3：图形化web用户界面

软件设计在SE中所处的位置



从建筑设计看软件设计

- **“设计良好的建筑应该展示出坚固、适用和令人赏心悦目的特点。”**
- **对好的软件设计来说也是如此**
  - 坚固：软件应该不含任何妨碍其功能的缺陷；
  - 适用：软件要符合开发的目标，满足用户需求；
  - 赏心悦目：使用软件的体验应该是愉快的。

良好的软件设计的三个特征

- 目标：设计必须是实现所有包含在分析模型中的明确需求，满足利益相关者期望的所有隐含需求；
- 形态：对开发、测试和维护人员来说，设计必须是可读的、可理解的、可操作的指南；
- 内容：设计必须提供软件的全貌，从实现的角度去说明功能、数据、行为等各个方面。

设计的目标：质量

- “设计阶段”是软件工程中形成质量的关键阶段，其后所有阶段的活动都要依赖于设计的结果。
- “编写一段能工作的灵巧的代码是一回事，而设计能支持某个长久业务的东西则完全是另一回事。”

—— C. Ferguson

软件质量

- **外部质量：面向最终用户**
  - 如易用性、效率、可靠性、正确性、完整性等
- **内部质量：面向软件工程师，技术的角度**
  - 如可维护性、灵活性、可重用性、可测试性等

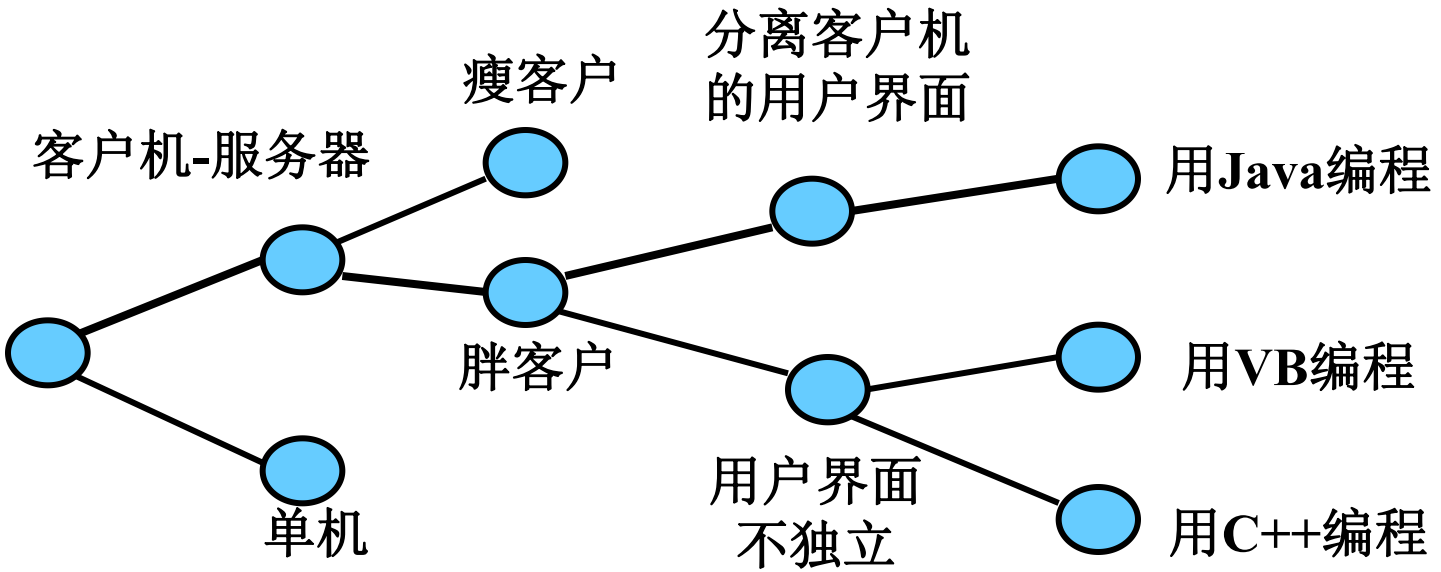
大多数软件设计师只关注外部质量忽视内部质量，导致软件变更代价大

软件设计的方式

- **软件设计的目标是创作出坚固、适用和赏心悦目的模型或设计表示**
- **方式：先实现多样化再进行聚合**
  - 多样化：找到各种可能的解决方案
  - 聚合：折中选取最适合的
- **多样化和聚合需要直觉和判断力，其质量取决于**
  - 构造类似实体的经验
  - 一系列指导模型演化方式的原则和（或）启发
  - 一系列质量评价的标准以及导出最终设计表示的迭代过程
- **软件工程缺少经典工程设计（如建筑）所具有的深度、灵活性和定量性，但是*软件设计的方法是存在的，设计质量的标准是可以获得的，设计表示法也是能够应用的。***

设计=不断的作出决策

- 解决“How to do”，就需要不断的做出各种“设计决策”，在各类需求之间进行“折中”，使得最终设计性能达到最优。



主要内容

- 1.软件设计的背景
- 2.软件设计中的核心概念
- 3.软件设计模型

设计概念

- **软件工程历史上，产生了一系列基本的软件设计概念**
- **每种概念的关注程度不断变化，但都经历了时间的考验**
  - 抽象
  - 体系结构
  - 模式
  - 关注点分离
  - 模块化
  - 信息隐蔽
  - 功能独立
  - 重构
  - 设计类

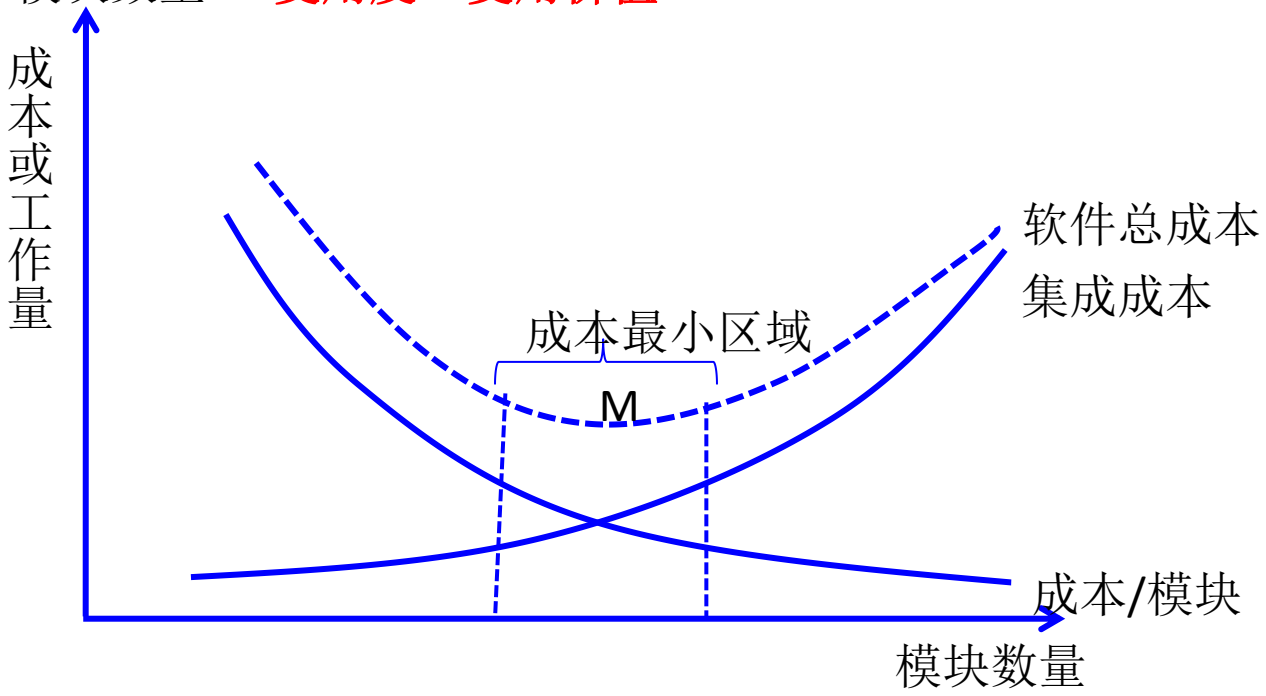
设计概念

- **抽象：强调关键特征，忽略实现细节**
  - 过程抽象：具有明确和有限功能的指令序列
  - 数据抽象：描述数据对象的冠名数据集合

- **体系结构：程序构件（模块）的结构、构件交互的形式、构件的数据结构**
  - 功能模型：表示系统的功能层次结构
  - 框架模型：解决某一类问题的处理流程
  - 结构模型：程序构件的一个有组织的集合

设计概念：模块化和软件成本

- **关注点分离**：关注点是一个特征或行为，软件需求模型的一部分；任何复杂问题如果被分解为可以独立解决和优化的若干块，该复杂问题能够更容易地被处理；
- **模块化**：关注点分离最常见的表现，分治策略，将软件划分为独立构件（模块）
  - 合理划分模块数量　　复用度？复用价值？



设计概念

- **信息隐藏：每个模块对其他所有模块都隐藏自己的设计决策**
  - 软件只通过定义清晰的接口通信
  - 每一个接口尽可能暴露最少的信息
  - 如果内部细节发生变化，外部的受到影响应当最小

- **功能独立：是关注点分离、模块化、抽象概念和信息隐藏的结果**
  - 功能专一，避免与其他模块过多交互
  - 独立性的评价：内聚性和耦合性

- **求精：**
  - 逐步求精
  - 分解细化
  - 层次结构

设计概念

- **重构**：是一种重新组织的技术，不改变外部行为而是改进内部结构

- **设计类**：精化分析类、创建新的设计类
  - 用户接口类（边界类）：人机交互所必需的抽象
  - 领域类（实体类）：分析类的精化
  - 过程类（控制类）：底层业务抽象
  - 持久类：持续存在的数据存储
  - 系统类：软件管理和控制功能

基于模式的软件设计

- 模式：特定问题的解决方案

- 体系结构模式：定义软件的整体结构，体现了子系统和软件构件之间的关系，并定义了说明体系结构元素（类、包、构件、子系统）之间关系的规则
- 设计模式：这些模式解决设计中特有的元素，例如解决某些设计问题中的构件聚集、构件间的联系或影响构件到构件之间通信的机制
- 框架（Framework）：已实现的特定的骨架基础设施，提供了基础功能，并规定了构件及构件的连接方式。开发者只需要将注意力集中于业务逻辑的实现

建议同学们选择成熟的框架进行开发

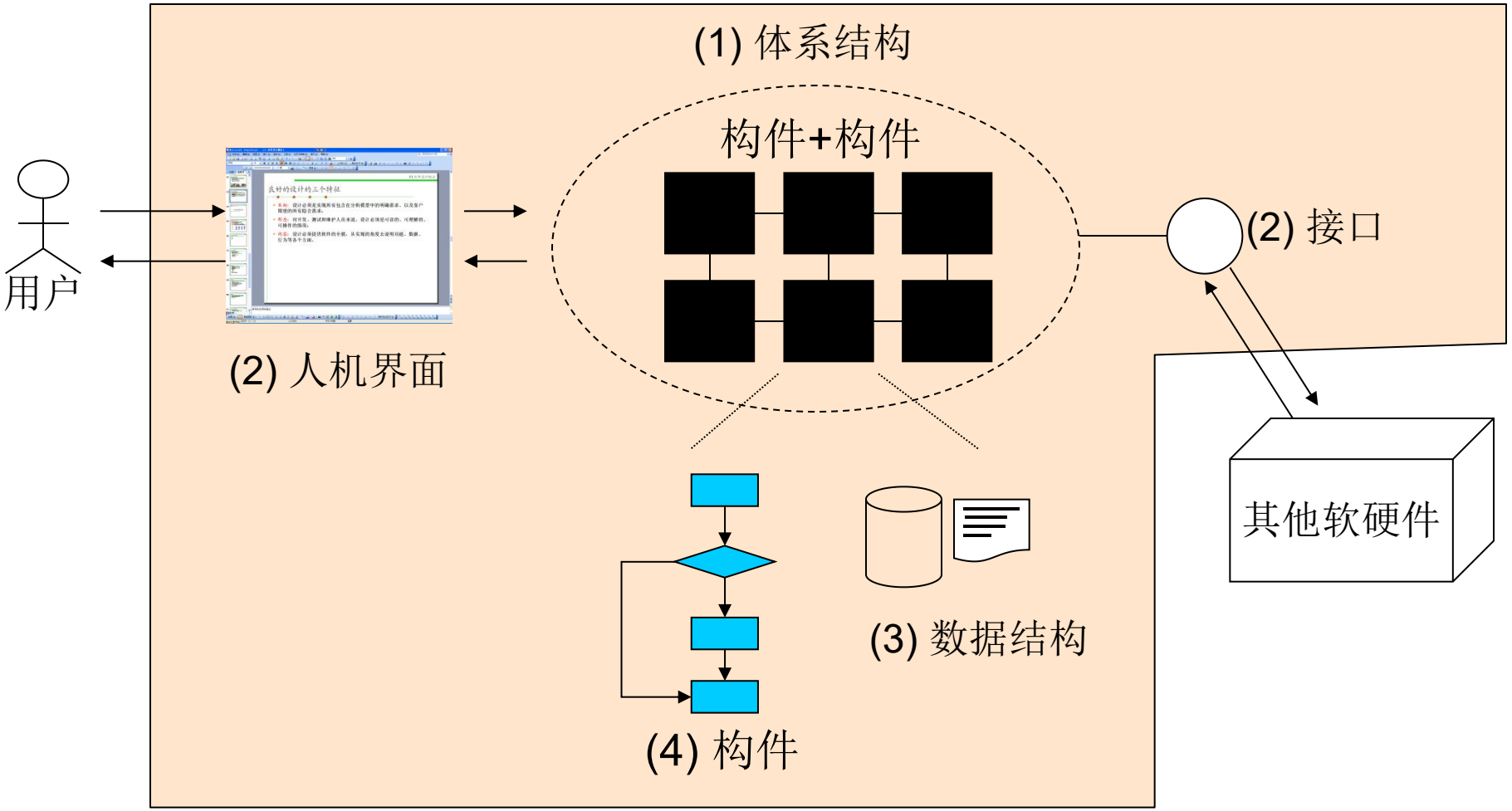
设计建模原则

- 设计可追溯到分析模型
- 经常关注待建系统的架构
- 数据设计与功能设计同等重要
- 必须设计接口（包括内部接口和外部接口）
- 用户界面设计必须符合最终用户要求
- 功能独立的构件级设计
- 构件之间以及构件与外部环境之间松散耦合
- 设计表述（模型）应该做到尽可能易于理解
- 设计应该迭代式进行。每一次迭代，设计者都应该尽力简化

主要内容

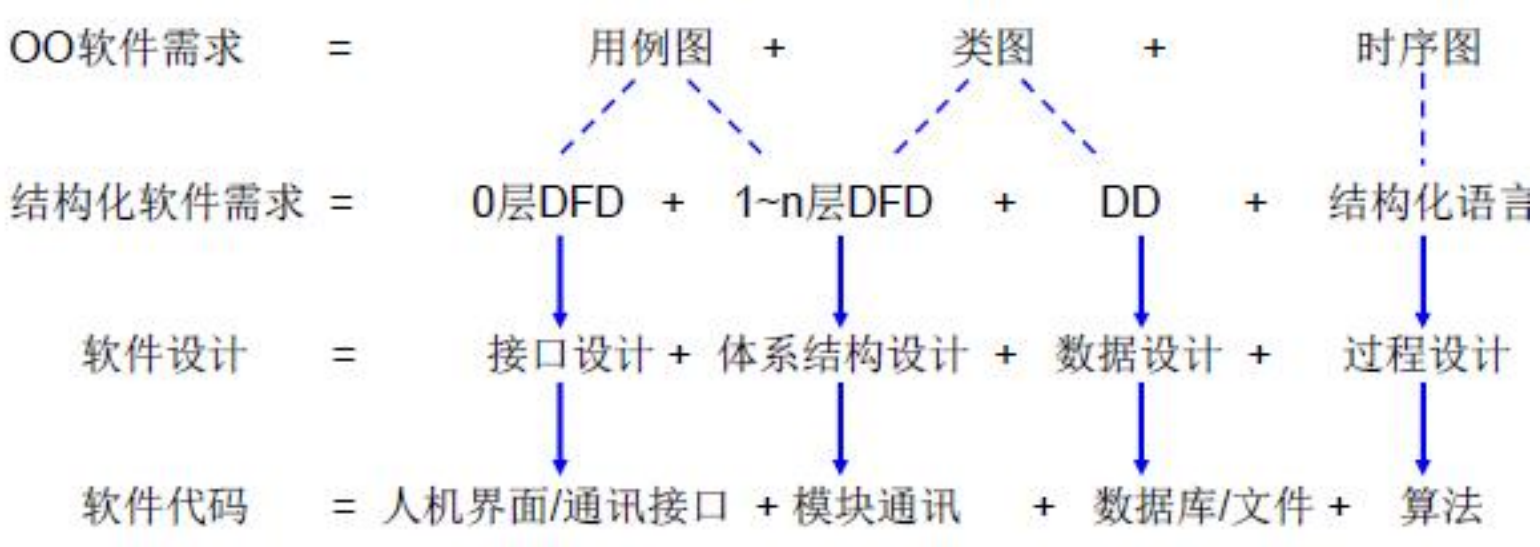
- 1.软件设计的背景
- 2.软件设计中的核心概念
- 3.软件设计模型

软件设计的四方面内容



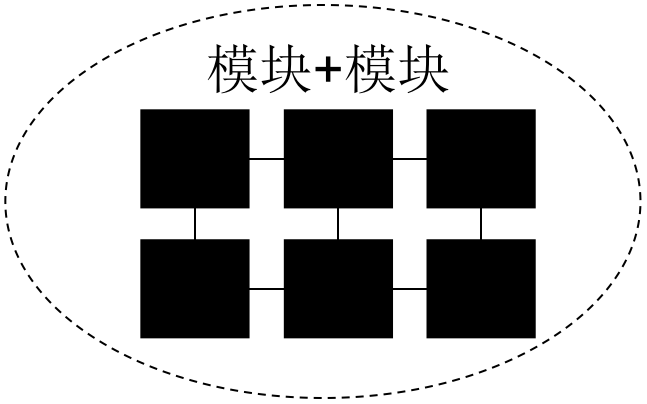
软件设计的元素

- 体系结构设计：定义了软件的主要结构元素之间的联系，也用于达到系统所定义需求的体系结构风格和设计模式以及影响体系结构实现方式的约束
- 接口设计：描述了软件和协作系统之间、软件和使用人员之间是如何通信的
- 数据/类设计：将分析类模型转化为设计类的实现以及软件实现所要求的数据结构
- 构件级设计：将软件体系结构的结构元素变换为对软件构件的过程性描述



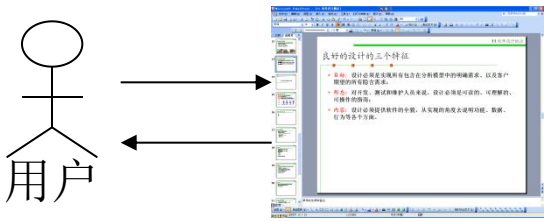
1.软件体系结构设计

- 选择适合于需求的软件体系结构风格，软件架构设计；
- 如何以最佳的方式划分一个系统，如何标识组件，组件之间如何通信，信息如何沟通，系统的元素如何能够独立地进化
- 例如：基于功能层次结构建立系统：
  - 采用某种设计方法，将系统按功能划分成模块的层次结构
  - 确定每个模块的功能
  - 建立与已确定的软件需求的对应关系
  - 确定模块间的调用关系
  - 确定模块间的接口
  - 评估模块划分的质量

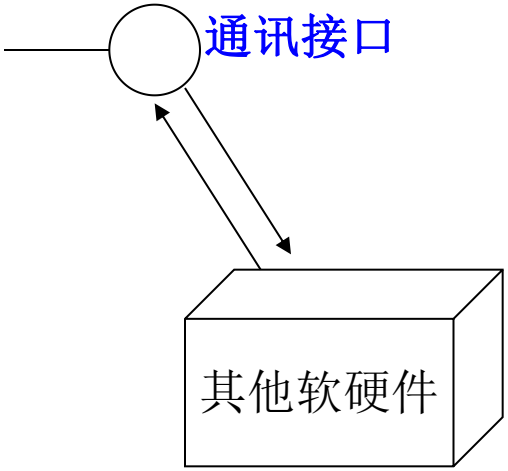


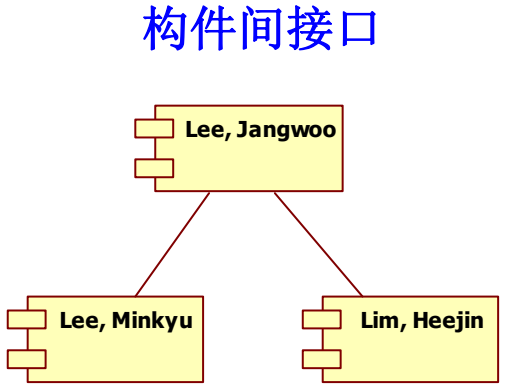
2.接口设计

- 接口是类、构件或其他分类（包括子系统）的外部可见的（公共的）操作说明，而没有内部结构的规格说明
  - 用户界面（UI）
  - 与其他系统、硬件的外部接口
  - 各种构件之间的内部接口



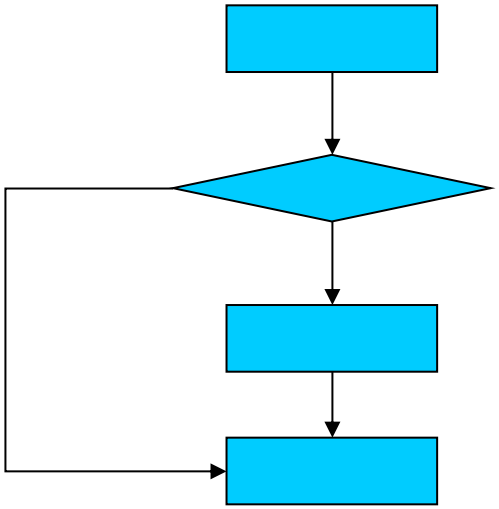
人机界面

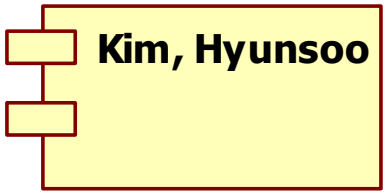




3.构件级设计

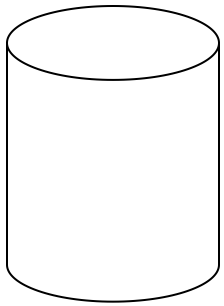
- **构件是面向软件体系架构的可复用软件模块**
- **完整地描述了每个软件构件的内部细节**
  - 为本地数据对象定义数据结构，为构件内的处理定义算法细节
  - 定义允许访问所有构件操作（行为）的接口

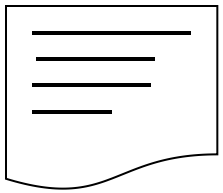




4.数据设计

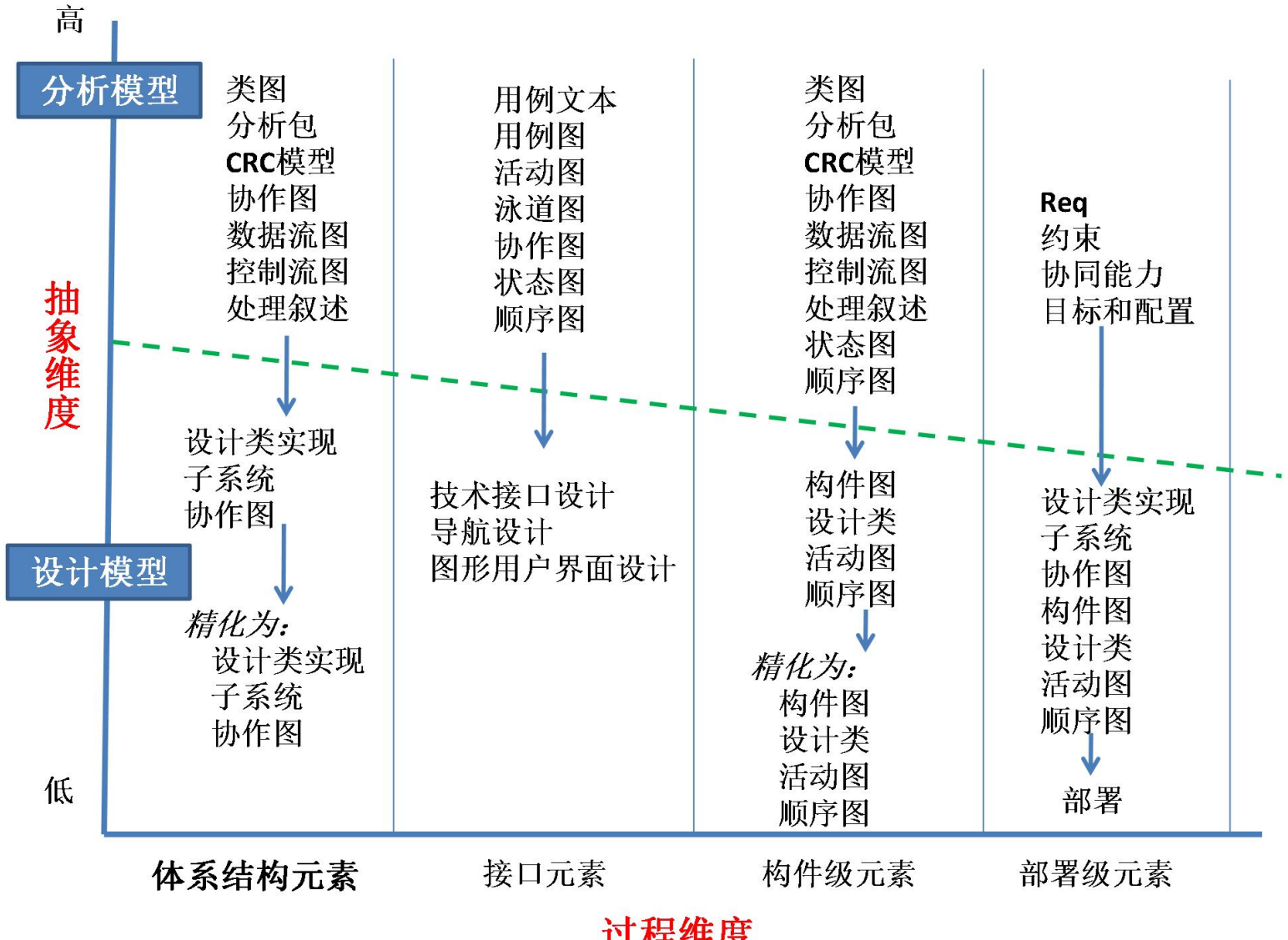
- **体系结构级数据设计**
  - 确定软件涉及的文件系统的结构以及数据库的模式、子模式，进行数据完整性和安全性的设计
  - 确定输入、输出文件的详细的数据结构
- **构件级数据设计**
  - 结合算法设计，确定算法所必需的逻辑数据结构及其操作
  - 确定对逻辑数据结构所必需的那些操作的程序模块(软件包)
  - 限制和确定各个数据设计决策的影响范围
  - 确定其详细的数据结构和使用规则
  - 数据的一致性、冗余性设计



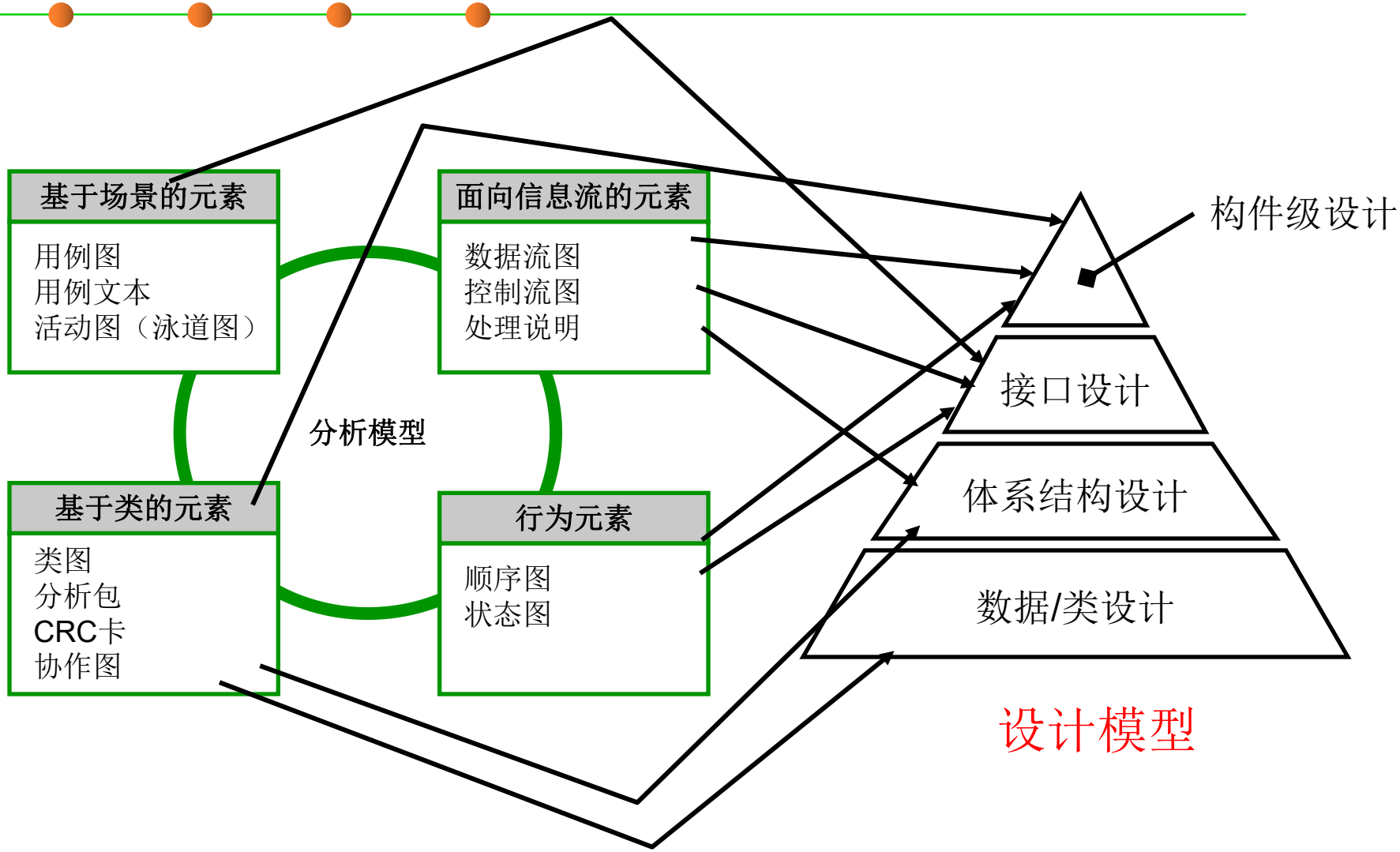


数据结构和数据库课程详细介绍

设计模型的维度

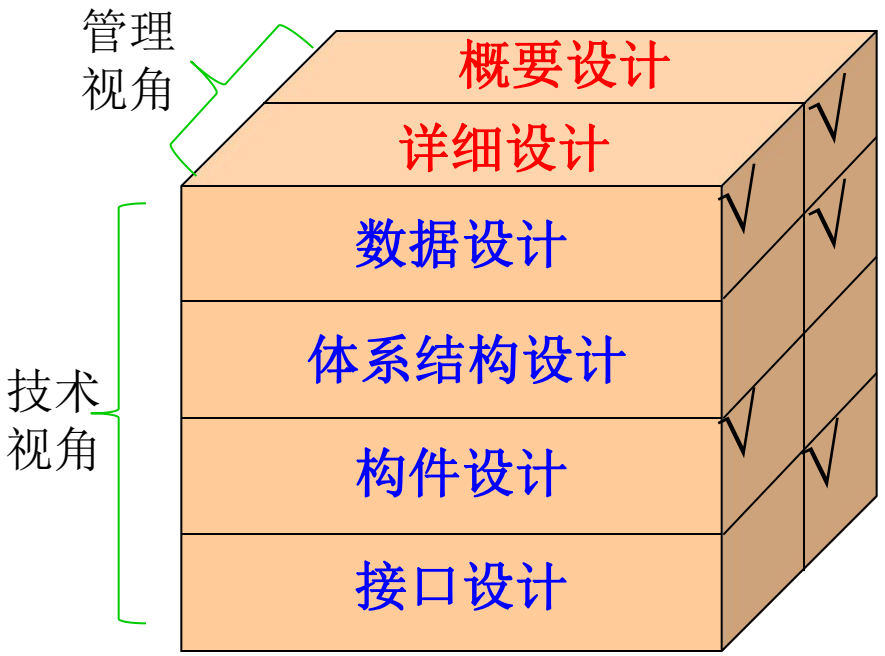

高
分析模型
类图
分析包
CRC模型
协作图
数据流图
控制流图
处理叙述
抽象维度
设计类实现
子系统
协作图
设计模型
精化为:
设计类实现
子系统
协作图
低
用例文本
用例图
活动图
泳道图
协作图
状态图
顺序图
技术接口设计
导航设计
图形用户界面设计
类图
分析包
CRC模型
协作图
数据流图
控制流图
处理叙述
状态图
顺序图
构件图
设计类
活动图
顺序图
精化为:
构件图
设计类
活动图
顺序图
Req
约束
协同能力
目标和配置
设计类实现
子系统
协作图
构件图
设计类
活动图
顺序图
部署
体系结构元素
接口元素
构件级元素
部署级元素
过程维度

从分析模型到设计模型的转化



软件设计的两大阶段

- **从工程管理的角度看，软件设计包括：**
  - 概要设计：将软件需求转化为数据结构和软件的系统结构
  - 详细设计：即构件设计，通过对软件结构表示进行细化，得到软件的详细的数据结构和算法



结束

2024年6月10日

面向对象设计

- **Jacobson：“当实现的细节开始显现，那就是设计”**
- **对象关注点转移到解决域**
  - 对象、语义和关系被确定
  - 贯彻需求，不断迭代
  - 注重质量和原则

面向对象的设计的两个阶段

- **系统设计(System Design)**
  - **相当于概要设计(即设计系统的体系结构)；**
  - 选择解决问题的基本途径；
  - 决策整个系统的结构与风格；
- **对象设计(Object Design)**
  - **相当于详细设计(即设计对象内部的具体实现)；**
  - 细化需求分析模型和系统体系结构设计模型；
  - 识别新的对象；
  - 在系统所需的应用对象与可复用的商业构件之间建立关联；
    - 识别系统中的应用对象；
    - 调整已有的构件；
    - 给出每个子系统/类的精确规格说明。