SA OVERVIEW

- **世界虽变换万端，而理为一贯**
- **体系结构是指整个系统构成的基本和主体形态。**
  - 体系结构成为建立和考察系统的总体指导或基本出发点
  - 体系结构对一个应用系统的生命周期有着非常重要的影响
  - 软件的复杂性打破了以往硬件与软件的平衡，SA被提出了
- **软件体系结构是软件在设计构成上的基本、可供设计选择的形态和总体结构。**
  - 软件设计中可供选择的结构形态
  - 每个结构概念都传达了一些信息
  - 有关领域愈是成熟和发展，人们对结构的认识也就愈加深刻
  - SA概念的提出和应用，说明了软件设计在高层次上的发展并走向成熟

Main Contents

- Chapter 1 概述
- Chapter 2 软件体系结构的研究与发展
- Chapter 3 软件体系结构的层次性
- Chapter 4 软件体系结构的设计原理
- Chapter 5 部件和连接器
- Chapter 6 体系结构的一般描述
- Chapter 7 设计模式
- Others：
  - The “4+1” View Model of Software Architecture
  - 体系结构风格
  - 一些面向对象的设计法则

Chapter 1 概述

- 1.1 软件体系结构
  - 软件体系结构定义了软件的局部和总体计算部件的构成，以及这些部件之间的相互作用关系。
  - 总的来看，体系结构是由结构和功能各异、相互作用的部件集合，按照层次构成的
- 1.2 当前的软件设计
  - 体系结构的设计选择对于软件的长远成功是至关重要的
  - 体系结构描述的不规范性
  - 体系结构的理论和工具
- 1.3 软件设计的层次
  - 结构级
  - 代码级
  - 执行级
- 1.4 体系结构与软件的工程
- 1.5 软件体系结构的知识体系

Chapter 2 软件体系结构的研究与发展

- **2.1 软件工程设计和软件体系结构**
  - 便于维护和升级，因而应该是模块化的
  - 设计应该是便于移植的（移植比重新设计花费要小的多）
  - 设计过程应该受到理性的控制 Intellectual Control
  - 设计应该表现出概念的完整性
- **2.2 什么是软件体系结构**
  - **An architecture is the set of significant decisions about the organization of a software system, the selection of the structural elements and their interfaces by which the system is composed, together with their behavior as specified in the collaborations among those elements, the composition of these structural and behavioral elements into progressively larger subsystems, and the architectural style that guides this organization---these elements and their interfaces, their collaborations, and their composition .**
- **2.3 软件体系结构的意义和目标**
  - 软件体系结构是软件开发过程初期的产品，对于开发进度和软件质量的一切资金和劳务投入，可以获得最好的回报。
  - 体系结构设计是形成的投资高汇报的重要因素。
  - 正确有效的体系结构设计会给软件开发带来极大的便利。
- **2.4 软件体系结构的研究范畴**
  - 体系结构描述语言与工具
  - 产品线与标准：企业、需求、架构、系统
  - 软件体系结构风格及风格应用
  - 体系结构文档化

Chapter 3 软件体系结构的层次性

- **3.1 体系结构的基础和层次特性**
  - 从建筑学看软件的构成：基础、层次、模式、角色划分
- **3.2 软件体系结构的层次结构模型**
  - 6层
- **3.3 从层次模型看软件体系结构**
  - 数据库、网络、JavaEE、Spring ， etc.
  - 体系结构是关于软件的构成部件及其连接的分层的结构框架
  - 体系结构分析与设计涵盖并指导着从逻辑结构设计到运行实现的软件工程的全部过程

Chapter 4 软件体系结构的设计原理

- **体系结构设计中遵循的原理**
  - 抽象
  - 封装
  - 数据隐藏
  - 模块化
  - 注意点分离
  - 耦合和内聚
  - 充分性、完备性和原始性
  - 策略和实现的分离
  - 接口与实现的分离
  - 分而治之
  - 层次化

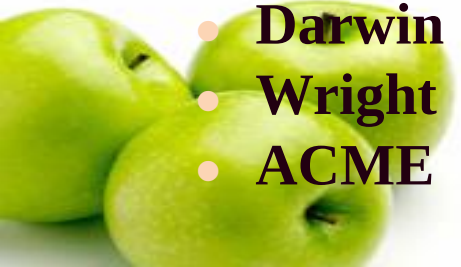
- **软件的非功能特性**
  - 可变性
  - 互操作性
  - 效率
  - 可靠性
  - 可测试性
  - 可重用性

Chapter 5 部件和连接器

- 部件和连接器被公认为体系结构的两大类构成部分
- 部件
- 连接
  - 计算机硬件提供了实现一切连接的基础
  - 过程调用、中断、I/O、DMA、事件、进程、线程、共享、同步、并/串行、并发， etc.
- 连接器

Chapter 6 体系结构的一般描述

- 通用的一般的描述方法
  - 主子程序
  - 数据抽象和面向对象
  - 层次结构
- 理论的形式化方法
  - Z Notation
  - CSP
  - 类属理论
  - 化学抽象机模型
- 软件体系结构集成环境
  - **UniCon**
  - **Darwin**
  - **Wright**
  - **ACME**



Chapter 7 设计模式

- 概述
  - 美可以客观评价么？
- 设计模式的特点与应用
  - MVC
- 设计模式的方法

		目的		
		创建型	结构型	行为型
范围	类	Factory Method	Adapter	Interpreter Template Method
	对象	Abstract Factory Builder Prototype Singleton	Bridge Composite Decorator Façade Flyweight Proxy	Chain Of Responsibility Command Iterator Mediator Memonto Observer State Strategy Visitor

创建型模式(5)

- **Singleton**模式解决的是实体对象个数的问题。除了Singleton之外，其他创建型模式解决的都是new所带来的耦合关系。
- **Factory Method, Abstract Factory, Builder**都需要一个额外的工厂类来负责实例化“易变对象”，而Prototype则是通过原型（一个特殊的工厂类）来克隆“易变对象”。
- 如果遇到“易变类”，起初的设计通常从Factory Method开始，当遇到更多的复杂变化时，再考虑重构为其他三种工厂模式（ Abstract Factory，Builder，**Prototype** ）。

结构型模式(7)

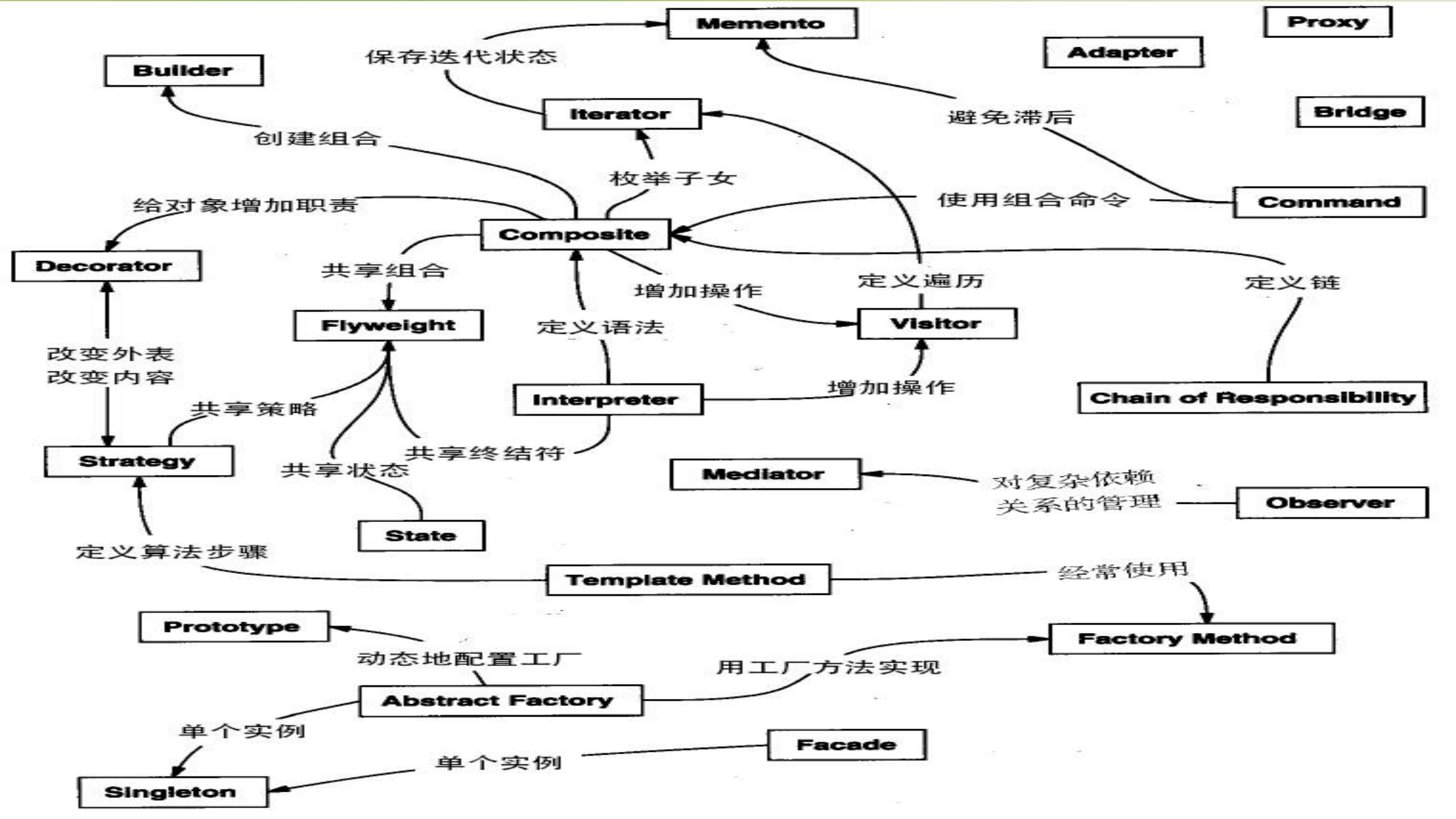
- **Adapter**模式注重转换接口，将不吻合的接口适配对接。
- **Bridge**模式注重分离接口与其实现，支持多维度变化。
- **Composite**模式注重统一接口，将“一对多”的关系转化为“一对一”的关系。
- **Decorator**模式注重稳定接口，在此前提下为对象扩展功能。
- **Façade**模式注重简化接口，简化子系统与外部客户程序的依赖关系。
- **Flyweight** 模式注重保留接口，在内部使用共享技术对对象存储进行优化。
- **Proxy** 模式注重假借接口，增加间接层来实现灵活控制。

行为型模式(11)

- **Template Method**模式封装算法结构，支持算法子步骤变化。
- **Strategy**模式注重封装算法，支持算法的变化。
- **State**模式注重封装与状态相关的行为，支持状态的变化。
- **Memento**模式注重封装对象状态变化，支持状态保存/恢复。
- **Mediator**模式注重封装对象间的交互，支持对象交互的变化。
- **Chain Of Responsibility**模式注重封装对象责任，支持责任的变化。
- **Command**模式注重将请求封装为对象，支持请求的变化。
- **Iterator** 模式注重封装集合对象内部结构，支持集合的变化。
- **Interpreter**模式注重封装特定领域变化，支持领域问题的频繁变化。
- **Observer模式**注重封装对象通知，支持通信对象的变化。
- **Visitor模式**注重封装对象操作变化，在运行时为类层次结构动态添加新的操作。

设计模式总结

- 设计模式建立在对系统变化点的基础上，哪里有变化点，哪里应用设计模式。
- 设计模式应该以演化的方式来获得，系统的变化点往往是经过不断演化才能准确定位。
- 不能为了模式而模式，设计模式是一种软件设计的软力量，而非标准。不应夸大设计模式的作用。
- GOF 23 Pattern :简单的、复杂的；易理解的、不易理解的。
- 共性：抽象、接口、继承、委托、解耦、复用。
- 设计模式将带来什么？
  - 一套通用的设计词汇。
  - 书写文档和学习的辅助手段。
  - 现有方法的一种补充。
  - 重构目标。


Memento
Proxy
保存迭代状态
Adapter
Builder
Iterator
避免滞后
Bridge
创建组合
枚举子女
给对象增加职责
使用组合命令
Command
Composite
Decorator
共享组合
定义遍历
定义链
增加操作
Flyweight
Visitor
定义语法
改变外表
改变内容
Interpreter
增加操作
Chain of Responsibility
共享策略
Strategy
共享终结符
共享状态
Mediator
对复杂依赖
关系的管理
Observer
State
定义算法步骤
Template Method
经常使用
Prototype
Factory Method
动态地配置工厂
用工厂方法实现
Abstract Factory
单个实例
Facade
单个实例
Singleton

Others

- **The “4+1” View Model of Software Architecture**
  - Logical view、Process view、Development view、Physical view、Scenario
- **体系结构风格**
  - 数据流、调用和返回、独立构件、虚拟机、数据中心
- **一些面向对象的设计法则**
  - 优先使用（对象）组合，而非（类）继承
  - 针对接口编程，而非（接口的）实现
  - 开放－封闭法则（OCP）
  - Liskov替换法则（LSP）
  - LOD
  - etc.

考试内容

- **题型**
  - 名词解释（5×2=10）
  - 填空 （10×1=10）
  - 选择 （30×1=30）
  - 简答（叙述）（5×6=30）
  - 设计题 （2×10=20）
- **成绩构成**
  - **期末**： 试卷 50%
  - **平时**： 到课+作业（实验） 50%

考试安排

- **对象：** 2018级本科生
- **科目：** 软件体系结构
- **时间：** 2021-06-18（星期五）10:30-12:30
- **地点：** 海韵教学楼302（61）
  海韵教学楼307（57）
  海韵教学楼306（30）

Welcome and Enjoy!

好好学习，天天向上