-软件架构概念

-模型与建模方法 （4+1视图模型：逻辑、进程、物理、开发）

-***风格***与模式

-***基于架构***的软件开发

-架构演化与维护

***软件架构***

定义：软件架构包括**构件（component）**，连接件，约束（基本元素）端口和角色

构建和交互关系的集合（组成派）、重要设计决策的集合（决策派）

软件系统设计图，包含软件设计决策（**结构组织**）

特征：**重用性**、利益相关者多、关注点分离（**模块化**，分治）、质量驱动（**非功能性需求**）、概念完整性、循环风格

发展阶段：

1、基础研究阶段：Modular（模块化，逻辑切分）

2、概念体系核心技术形成：Software architecture，component（**组件化**，功能分隔）

3、理论体系丰富：基于组件软件架构（架构的通用描述方法，演化和重用）

4、软件架构理论完善与实践

架构描述：ADL、形式化方法、4+1架构、UML、IEEE软件架构描述

架构分析：结构分析（开发风格选择），功能分析，非功能分析

设计方法：工作，用例，模式，领域，属性驱动

测试：仿真系统测试

***架构模型***

对架构设计决策具象化和文档化

建模方法：非规范图形（盒线图），UML，形式化建模（符号化），基于UML形式化

基于图形的可视化建模：树形结构，树地图（Tree map），旭日图，双曲树

UML：表示法统一，支持多视图，**建模能力不强，语义精确性不足**，软件架构风格无法体现 建模阶段：当作架构语言直接建模，扩展机制约束丰富元模型，扩充元模型

形式化建模：Z语言，Petri网，B语言，CSP（新语言，难学难用）

UML形式化：用符号语言对UML进行规约（类，用例，状态，顺序）

文本语言建模：文档描述整体架构，无图形结构不易理解

**M**odel **D**riven **A**rchitecture（模型驱动架构建模）：构造平台无关模型（PIM）验证可行并移植至平台相关模型（PSM）

***风格与模式（style & pattern）***

架构风格：描述特定领域中系统组织方式的惯用模式

设计元素（管道，过滤器，对象，服务），配置规则，语义解释

设计代码重用性，组织结构易理解，内部互操作性，需求和风格分析

**风格分类**：**数据流**、**调用返回**、**独立组件**（进程通信）、**虚拟机**（解释器）、**仓库**

1. 管道过滤器：输入-管道-过滤器(功能模块)-管道-输出（高内聚，低耦合）
2. 主，子程序风格：逐步分解细化，调用返回（数据访问效率高，可复用性低）
3. 面向对象：对象维护自身完整性，对象之间**相互隐蔽**，类标识调用**独立性低**
4. 层次化：系统组织为若干层次，层间存在接口，通过接口形成服务（问题分解，支持扩展，重用 OS）
5. 事件驱动：监听广播事件触发过程（独立，可复用性强 监听响应组件无控制权，无数据交换，正确性验证困难）
6. 解释器风格（Interpreter）：逐条边翻译边执行（程序可移植，语言**跨平台，**效率低 JVM）
7. 基于规则的系统：提取频繁变化的业务逻辑形成**规则库**（跨场景）
8. 仓库风格：两种组件：**中央数据存储**组件，相对独立组件（可扩展性好 需有**同步机制**防止数据死锁）
9. 黑板系统：知识源-控制器-**黑板**（数据结构）-控制器-知识源（知识共享，黑板数据结构可扩展重用 对黑板数据结构要求高，需同步和加锁机制）
10. C2风格：通过**连接件绑定**在一起的**组件网**，通信和处理分开（分层结构，易重用和替换，不需共享地址空间，**多用户** 大规模流式风格时不适用）
11. 客户机/服务器：两层CS（客户机-服务器） 三层CS（客户机-**应用服务器（部分数据与应用处理逻辑）**-数据服务器）