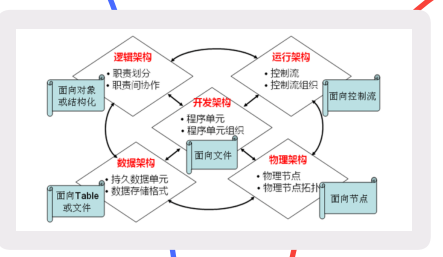
软件架构五视图法



架构解决什么问题？

- 开发之初，逻辑设计阶段的问题--系统如何开发
- 即将进入开发时的问题--数据是如何持久化的，数据库如何选型、存储方案等
- 开发过程中的问题--如何保证开发质量、如何分层、代码可扩展性、使用的设计模式、依赖哪些框架、开发语言等
- 开发完成后的问题--运行阶段如何保证运行器件的质量属性、性能、可伸展行等
- 系统的部署问--运行在什么硬件上、服务器的性能、操作系统选型、网络拓扑等

数据架构

- 着重考虑数据需求
- 关注点
  - 持久化数据的存储方案
  - 数据存储格式
  - 数据传递
  - 数据复制
  - 数据同步
- 用E-R图和数据流图表示

逻辑架构

- 着重考虑功能需求
- 系统应该向用户提供什么样的服务
- 关注点是行为或职责的划分
- 关注用户可见的功能
- 提供辅助功能模块
- 它们可能是逻辑层，功能模块和类

运行架构

- 着重考虑运行期重量属性
  - 性能
  - 可伸缩性
  - 持续可用性
  - 安全性
- 关注点是系统的并发与同步问题
- 关注进程，线程，对象等运行时概念
- 考虑并发，同步，通信等问题
- 偏重程序包在编译时期的静态依赖关系
- 解决运行时各单元的交互问题

开发架构

- 着重考虑开发期质量属性
  - 可扩展性
  - 可重用性
  - 可移植性
  - 易理解性
  - 易测试性
- 关注软件模块实际组织方式
  - 源程序文件
  - 配置文件
  - 源程序包
  - 现成框架，类库
  - 提供中间件
  - 编译后目标文件
  - 第三方库文件
- 逻辑层会映射到程序包

物理架构

- 关注软件如何安装或部署到物理机器
- 部署机器和网络配合软件系统的可靠性，可伸缩性等要求
- 重视目标程序的静态位置问题
- 考虑整个软件系统之间是如何互相影响的
- 着重考虑安装和部署需求
- 关注点是软件的目标单元如何映射到硬件
- 关注相关的质量届性
  - 可靠性
  - 可伸缩性
  - 持续可用性
  - 性能
  - 安全性

表达方式

- 逻辑架构：通常使用 UML 类图、活动图等进行表达。
- 开发架构：常用工具包括组件图和包图。
- 运行架构：通常用序列图、协作图等进行描述。
- 物理架构：通常用序列图、协作图等进行描述。
- 数据架构：E-R图和数据流图。