1、逻辑架构

逻辑架构关注职责划分和接口定义。不同粒度的职责需要被关注，它们可能是逻辑层、功能子系统、模块、关键类等。不同通用程度的职责要分离，分别封装到专门模块、通用模块或通用机制中。

如果使用UML来描述架构的逻辑架构，则该视图的静态方面由包图、类图、对象图来描述，动态方面由序列图、协作图、状态图和活动图来描述。

核心设计任务是：模块划分、接口定义、领域模型细化。

2、开发架构

- 包括要编写的源程序
- 可以直接使用的第三方SDK和现成框架、类库。
- 目标代码文件的个数与形态（exe、war..）

如果使用UML来描述架构的开发架构，则该视图可能包括包图、类图和组件图等。

核心设计任务是：各种开发技术选型、程序文件划分到具体工程（Project）、程序文件之间的编译依赖关系。

3、物理架构

物理架构关注“目标程序及其依赖的运行库和系统软件”最终如何安装、烧写或部署到物理机器，以及如何部署机器和网络来配合软件系统的可靠性、可伸缩性等要求。

如果使用UML来描述架构的物理架构，则该视图可能包括部署图和组件图。

核心设计任务是：硬件分布、软件部署、方案优化。

4、运行架构

运行架构关注进程、线程、中断服务程序等运行时控制流，以及相关的并发、同步、通信等问题。

如果使用UML来描述架构的运行架构，则该视图的静态方面由包图、类图（其中主动类非常重要）和对象图（其中主动对象非常重要）等来说明关键运行时概念的结构关系。动态方面由序列图、协作图等来说明关键交互机制。

核心设计任务：并发技术选型、控制流划分、控制流间同步关系。

5、数据架构

数据架构关注持久化数据存储方案，不仅包括实体及实体关系的数据存储格式，还可能包括数据传递、数据复制、数据同步等策略。

核心设计任务是：持久化技术选型、数据存储格式、数据分布策略。