## 软件体系结构风格

软件体系结构风格是经过实践证明，可复用的优秀技术，是面向一类给定环境的架构设计决策的集合，这些通用的设计决策形成了一种特定的模式，为一族系统提供粗粒度的抽象框架。每一个软件系统都有一种或多种软件体系结构风格。以下是在本系统中涉及的主要风格。

#### 层次体系结构风格

###### 组成

**构件**：层和层中的构件

**连接件**：层间交互协议

图片包含 图表

描述已自动生成

结构图

###### 工作方式

系统被组织成若干个层次，每个层次由一系列构件组成。层次之间存在接口，通过接口形成调用/返回关系。每一层担任两个角色：为上层提供服务、调用下层服务。层间的通信是显示的和松耦合的。比如，表现层负责界面实现，需要调用控制层为这个界面提供的数据和事件处理方法，本项目中表现层（View层）和控制层（ViewModel层）间的交互协议是WPF的数据双向绑定和指令机制。再比如，控制层（ViewModel层）负责实现业务逻辑，需要使用服务层（Services层）和模型层（Models层）提供的功能函数和域对象，交互协议是控制层和下面两层的调用返回关系。

图形用户界面, 应用程序

描述已自动生成

项目目录结构

###### 优缺点

**优点**：

1. 支持逐层抽象的系统设计，有利于设计者对一个复杂系统进行分解；
2. 支持更新，因为每一层至多和相邻的上下层交互，因此功能的改变通常只影响相邻的上下层；
3. 支持复用，如果某独立层保证了功能的完整性并且提供了文档化的接口，便可在多个语境中复用。一组标准接口也可以使用各种不同的实现方法。
4. 支持测试。具有定义明确的层接口以及交换层接口的各个实现的能力提高了可测试性。

**缺点：**

1. **并不是每个系统都可以很容易地划分为分层的模式，甚至即使一个系统的逻辑结构是层次化的，出于对系统性能的考虑，不得不把一些低级或高级的功能综合起来；**
2. **效率的降低：由分层风格构成的系统，运行效率往往低于整体结构。在上层中的服务如果有很多依赖于最底层，则相关的数据必须通过一些中间层的若干次转化，才能传到；**
3. **很难找到合适的、正确的层次抽象方法：层数太少，分层不能完全发挥这种风格的可复用性、可更改性和可移植性上的潜力。层数过多，则引入不必要的复杂性和层间隔离冗余以及层间传输开销。目前，没有可行的广为人们所认可的层粒度的确定和层任务的分配方法。**

#### ****2. 客户机/服务器风格****

###### 组成

**构件**：客户机软件（使用WPF实现）、服务器软件（使用WCF实现）

**连接件**：建立在网络协议（传输层）之上，驻留在服务器和客户机两端，主要作用是提供透明的网络连接与服务，让应用及开发人员感觉不到网络的存在，以更直观的方式使用网络。遵循公共的通信协议：请求的表达；返回结果的表达；连接关系和状态的表达。例如：RPC(远程过程调用)；HTTP；SOAP(简单对象访问协议)；

###### 工作方式

在本项目中，主机进程就是服务端，在其中需要创建服务，然后将服务通过EndPoint（终结点）与客户端进行通信。客户端在使用服务时，需要首先创建一个代理服务，然后调用这个代理服务。EndPoint（终结点）包含了服务地址（Address）、绑定格式（Binding）和契约（Contract）。其中，地址定义了服务的位置，绑定定义了服务通信的方式，契约定义了服务所提供的功能。

图示

描述已自动生成

在服务端程序的App.config文件中配置的EndPoint信息

图形用户界面, 应用程序

中度可信度描述已自动生成

###### 优缺点

1. **本项目中客户端属于胖客户端，胖客户端的优缺点如下**

优点：

* 1. 较低的服务器需求：胖客户端自己就可以完成很多业务处理
  2. 可以离线工作：通常不需要和服务器之间维持长久的连接
  3. 更好的多媒体性能：网络带宽比较敏感的时候，胖客户端更擅长运行丰富的多媒体应用
  4. 更灵活
  5. 充分利用已有的设备资源
  6. 更高的服务器容量：客户端执行的工作越多，服务器所需要的工作就越少，这可以增加服务器支持的用户数量

缺点：

a) 部署安装不方便

b) 客户端执行的工作太多，会导致体积臃肿，对客户机性能要求较高

1. **本项目类似三层C/S结构，优缺点如下**

优点：

* 1. 在用户数目较多的情况下，三层C/S结构将极大改善性能与灵活性(通常可支持数百个甚至更多用户)；
  2. 允许合理地划分三层结构的功能，使之在逻辑上保持相对独立性，能提高系统和软件的可维护性和可扩展性；——UI、BL、DB可以分别加以复用
  3. 允许更灵活有效地选用相应的平台和硬件系统，并且这些平台和各个组成部分可以具有良好的可升级性和开放性。
  4. 并行开发，可以选择各自最适合的开发平台和开发语言
  5. 利用功能层有效地隔离开表示层与数据层，未授权的用户难以绕过功能层而非法的访问数据层，为严格的安全管理奠定了坚实的基础。
  6. 将遗留系统移植到三层C/S下将非常容易；

缺点：

1. 层C/S结构各层间的通信效率若不高，即使分配给各层的硬件能力很强，其作为整体来说也达不到所要求的性能。
2. 设计时必须慎重考虑三层间的通信方法、通信频度及数据量，这和提高各层的独立性一样是三层C/S结构的关键问题。——分层风格的固有缺点

#### 3. 基于事件的隐式调用风格

###### 组成

构件：对象、过程、事件

连接件：事件-过程绑定



事件的分发机制

###### 工作过程

过程（事件处理器，事件的接收或处理方）向特定的事件进行注册，构件（事件源）再合适的时间发布事件。当某些事件被发布，向其注册的过程被隐式调用，且调用的次序不确定。

图形用户界面, 文本

描述已自动生成

图形用户界面, 文本, 应用程序, 电子邮件

描述已自动生成

上图中，ToggleButton组件（事件源）发布Click事件后注册在其上的命令（事件处理方）隐式调用事件处理函数

###### 优缺点

优点:

1. 能够很好的支持交互式系统（如用户输入/网络通信等），系统中的操作可异步执行，不必同步等待执行结果。
2. 为软件复用提供了强大的支持。当需要将一个构件加入现存系统中时，只需将它注册到系统的事件中；
3. 为系统动态演化带来了方便。构件独立存在，当用一个构件代替另一个构件时，不会影响到其它构件的接口；
4. 对事件的并发处理将提高系统性能；
5. 健壮性：一个构件出错将不会影响其他构件。

缺点：

1. 分布式的控制方式使得系统的同步、验证和调试变得异常困难。
2. 隐式调用增加了中间层必要的消耗，会降低事件的响应速度
3. 构件放弃了对系统计算的控制。一个构件触发一个事件时，不能确定其他构件是否会影响它。而且即使它知道事件注册了哪些构件的构成，它也不能保证这些过程被调用的顺序。
4. 传统的基于先验和后验条件的验证变得不可能。
5. 既然过程的语义必须依赖于被触发事件的上下文约束，关于正确性的推理存在问题。
6. 数据交换的问题。有时数据可被一个事件传递，但另一些情况下，基于事件的系统必须依靠一个共享的仓库进行交互。在这种情况下，全局性能和资源管理便成了问题