定义

软件体系结构是具有一定形式的结构化元素的集合（构件的集合）。构件分为三类包括处理构件（对数据进行加工）、数据构件（被加工的信息）和连接构件（连接体系结构的不同部分）。当然也可以说是由构件、连接件和约束条件组成的。 **体系结构=部件 + 连接件 + 约束**

构件

构件是具有某种功能的可复用的软件结构单元，表示系统中主要的**计算元素** 和 **数据存储**。
构件是一个抽象的概念，在程序中可以指程序函数、模块、对象、类等。、
构件是可预制和可重用的软件部件，是组成体系结构的基本计算单元或数据存储单元.它可大可小，比如共享变量的访问中共享变量就是一个连接件。而在现实世界的网络中，通信协议就充当了连接件

连接件

连接件是构件间建立和维护行为关联与信息传递的途径。连接包含下面两种要素：其中，机制指的实际中的**消息传递方式**。而协议则决定了 **消息的语义理解**。

连接件是可预制和可重用的软件部件，是组成体系结构的基本计算单元或数据存储单元.它可大可小，比如共享变量的访问中共享变量就是一个连接件。而在现实世界的网络中，通信协议就充当了连接件。

常用的软件体系结构风格

1、管道/过滤器 体系结构风格

主要包括过滤器和管道两种元素。在这种结构中，构件被称为过滤器，负责对数据进行加工处理。每个过滤器都有一组输入端口和输出端口，从输入端口接收数据，经过内部加工处理之后，传送到输出端口上。数据通过相邻过滤器之间的连接件进行传输，连接件可以看作输入数据流和输出数据流之间的通路，这就是管道。

优点

1. 简单性。
2. 支持复用。
3. 系统具有可扩展性和可进化型。
4. 系统并发性（每个过滤器可以独立运行，不同子任务可以并行执行，提高效率）。

5. 便于系统分析。

缺点

1. 系统处理工程是批处理方式。
2. 不适合用来设计交互式应用系统。
3. 由于没有通用的数据传输标准，因此每个过滤器都需要解析输入数据和合成数据。
4. 难以进行错误处理。

2、面向对象 体系结构风格

在面向对象体系结构中，软件工程的模块化、信息隐藏、抽象和重用原则得到了充分的体现。在这种体系结构中，数据表示和相关原语操作都被封装在抽象数据类型中。在这种风格中，对象是构件，也成为抽象数据类型的实例。对象与对象之间，通过函数调用和过程调用来进行交互。

优点：

1. 一个对象对外隐藏了自己的详细信息
2. 对象将数据和操作封装在一起
3. 继承和封装方法为对象服用提供了技术支持

缺点：

1. 如果一个对象要调用另一个对象，则必须知道它的标识和名称
2. 会产生连锁反应

3、事件驱动体系结构风格

事件驱动就是在当前系统的基础之上，根据事件声明和发展状况来驱动整个应用程序运行。
事件驱动体系结构的基本思想是：系统对外部的行为表现可以通过它对事件的处理来实现。在这种体系结构中，构件不再直接调用过程，而是声明事件。系统其他构件的过程可以在这些事件中进行注册。当触发一个事件的时候，系统会自动调用这个事件中注册的所有过程。因此，触发一个事件会引起其他构件的过程调用。

优点：

1. 事件声明者不需要知道哪些构件会响应事
2. 提高了软件复用能力
3. 便于系统升级

缺点：

1. 构件放弃了对计算的控制权，完全由系统来决定

2. 存在数据·传输问题

4、分层体系结构风格

惯用模式：

在分层风格中，系统将划分为一个层次结构。
每一层都具有高度的内聚性，包含抽象程度一致的各种构件，支持信息隐藏。
分层有助于将复杂系统划分为独立的模块，从而简化程序的设计和实现。
通过分解，可以将系统功能划分为一些具有明确定义的层，较高层是面对特定问题，较低层具有一般性。
每层都为上层提供服务，同时又利用了下层的逻辑功能。在分层体系结构中，每一层只对相邻层可见。层次之间的连接件是协议和过程调用。用以实现各层之间的交互。

优点：

1. 设计者可以将系统分解为一个增量的步骤序列从而完成复杂的业务逻辑。
2. 每一层之多和相邻的上下两层进行交互。
3. 只要给相邻层提供相同的接口。

缺点：

1. 并非所有系统都能够按照层次来进行划分。
2. 很难找到一种合适和正确的层次划分方法。
3. 在传输数据是，需要经过多个层次。
4. 多层结构难以调试。