**软件系统设计**是一个迭代的过程,最终结果是软件体系结构文档 (SAD)

**设计**是一种创造性过程 (continued)

多种利用现有解决方案的方法

**克隆:** 借鉴现有的整个设计，只做少许的调整

**参考模型:** 用于特定领域的分解系统的标准的、一般性的体系结构

软件体系结构也有一般性的解决方案, 称为**体系结构风格**

专注于一种体系结构风格会产生问题

好的体系结构设计应选择、改进以及集成多种体系结构风格，以产生符合期望的产品

体系结构风格反映了领域中众多系统所共有的结构和语义特性，并指导如何将各个模块和子系统有效地组织成一个完整的系统。

对体系结构风格的研究和实践为大粒度的软件复用提供了可能。

经典的体系结构风:

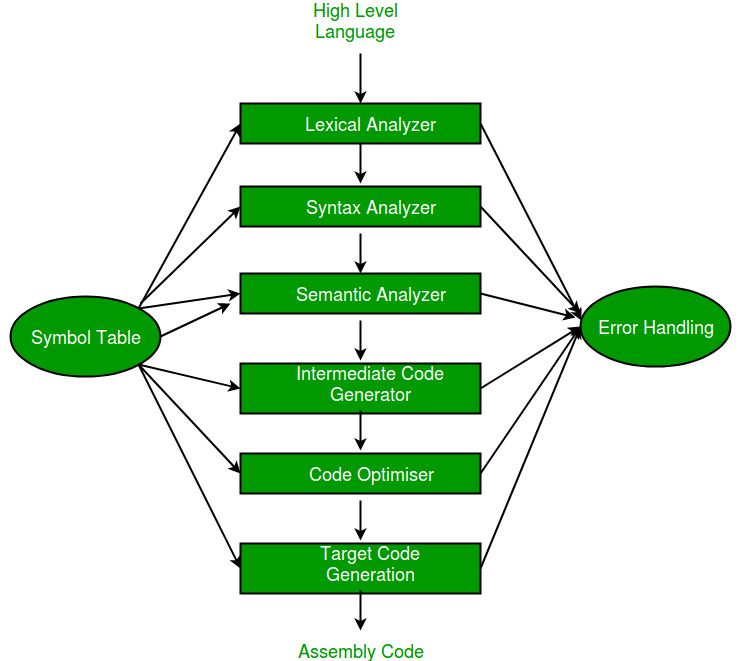
**1.数据流风格: 批处理序列; 管道/过滤器。**

把系统任务分成为几个独立的功能模块。这些功能模块采用通过系统的数据流连接。

在管道-过滤器风格下，每个功能模块都有一组输入和输出。功能模块称作过滤器（filters）；功能模块间的连接可以看作输入、输出数据流之间的通路，所以称作管道（pipes）。

管道-过滤器风格的特性之一在于过滤器的相对独立性，即过滤器独立完成自身功能，相互之间无需进行状态交互。管道-过滤器风格优点设计者可以将整个系统的输入、输出特性简单的理解为各个过滤器功能的合成。 基于管道-过滤器风格的系统具有较强的可维护性和可扩展性支持一些特定的分析，如吞吐量计算和死锁检测等,管道-过滤器风格具有并发性 管道-过滤器风格不足 交互式处理能力弱 具体实现比较复杂 数据流同步问题 数据加密与解析

编译器的管道-过滤器风格;



**2.调用/返回风格：主程序/子程序；面向对象风格；层次结构**。

层次结构风格

整个系统被组织成一个分层结构，每一层为上层提供服务，并作为下一层的客户。

层次结构的基本结构如下图所示

层次结构风格优点和缺点:

结构风格支持系统设计过程中的逐级抽象

基于层次结构风格的系统具有较好的可扩展性

层次结构风格支持软件复用

缺点：

并不是每个系统都可以很容易地划分为分层的模式

很难找到一个合适的、正确的层次抽象方法。

1. **独立构件风格：进程通讯；事件系统。**

独立构件风格的优势在于，由于构件之间的耦合度低，因此构件的可扩展性和可替换性要比调用返回风格高很多。但也是由于构件的独立特性，使得需要专门的IT系统来维护这些构件的运行。

虚拟机风格：解释器；基于规则的系统。解

1. **仓库风格：数据库系统；超文本系统；黑板系统。**

**虚拟机风格包含**

解释器

基于规则的系统

解释器

侧重于自定义，灵活的功能特性。这些功能特性通过解释器将其转换为目标平台上的可执行代码。

基于规则的系统

侧重于规则集，规则解释器，常见于人工智能领域和DSS（Decision Support Design）中。

相较于“数据流风格，调用返回风格，独立构件风格“，虚拟机风格搭建的是运行规则/语言/自定义描述的平台，而非实现具体的业务逻辑。

1. **过程控制环路**

是将过程输出的指定属性维护在一个特定的设定值，将事务处理看成输入、加工、输出、反馈、在输入的一个持续的过程模型。

**7.C/S风格**

客户机/服务器（C/S）体系结构是基于资源不对等，且为实现共享而提出来的。

C/S体系结构有三个主要组成部分：数据库服务器、客户应用程序和网络。

C/S风格—优点和缺点

优点：

界面和操作可以很丰富

安全性高

响应速度快

缺点：

适用面窄

用户群固定

维护成本高

**8.B/S风格**

B/S体系结构主要利用不断成熟的WWW浏览器技术，特别是浏览器嵌入的多种脚本语言

B/S体系结构有三个主要组成部分：浏览器、 Web服务器、数据库服务器。

B/S风格—优点和缺点

优点：

维护和升级方式简单

交互性较强

缺点：

在速度和安全性上需要花费巨大的设计成本

通常需要刷新页面（Ajax等可以一定程度缓解该问题）

通信开销大