软件体系结构风格的定义：

* 软件体系结构是描述某一特定应用领域中系统组织方式的惯用模式
* 体系结构定义了一个系统家族，即一个体系结构定义一个词汇表（包含一些构建和连接件类型）和一组约束（系统中构建和连接件的组合方式）
* 体系结构反应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
* 定义了描述系统的术语表和一组指导构建系统的规则

在管道/过滤器风格中，每个构件都有一组

输人和输出，构件读输人的数据流，经过内部处理，然后产生输出数据

流。这里的构件被称为过滤器。连接件被称为管道。此风格特别重要

的过滤器必须是独立的实体，它不能与其它的过滤器共享数据，而且

一个过滤器不知道它上游和下游的标识。

这种 风 格 的优点:(1)使得软构件具有良好的隐蔽性和高内聚、低

藕合的特点;(2)有助于设计者将整个系统的输人/输出行为看成是多

个过滤器的行为的简单合成;(3)支持软件重用;(4)易于系统维护和

增强系统性能;(5)允许对一些如吞吐量、死锁等属性的分析;(6)支持

并行执行。

这种 风 格 的缺点:(1)通常导致进程成为批处理的结构;(2)不适

合处理交互的应用;(3)数据传输上没有通用的标准，每个过滤器都增

加了解析和合成数据的工作，导致了系统性能下降，增加了编写过滤

器的复杂性，使数据传输量增加。

主程序/子过程风格：

该风格是结构化程序设计的一种典型风格，从功能的观点设计系统，通过逐步分解和逐步细化，得到系统体系结构。

构件：主程序、子程序

连接器：调用-返回机制

拓扑结构：层次化结构

本质：将大系统分解成若干模块，也就是我们常说的“模块化思想”，主程序调用了这些模块实现完整的系统功能。

仓库风格：

在仓库风格中，有两种不同的构件：中央数据结构（说明当前状态），独立构件（在中央数据存储上执行），仓库与独立构件之间的相互作用在系统中会有很大的变化。

按照控制策略的选取分类，可以产生两个主要的子类。如果输入流中某类时间触发进程执行的选择，则仓库是传统型数据库；另一方面，若中央数据结构的当前状态触发进程执行的选择，则仓库是黑板系统。

优点：（1）善于管理数据信息，适合大量数据的应用场；（2）适用于复杂的逻辑系统；（3）黑板系统模型能够更容易地处理任务之间的协作，系统更加灵活。数据库系统一直得到广泛的应用，比如在企业中使用的管理信息系统，ERP软件等等；黑板系统主要应用在需要复杂翻译解释的系统中，比如信号处理领域中的语音和模式识别。

层次系统风格：

层次系统组织成一个层次结构，每一层为上层服务，并作为下层客户。在一些层次系统中，内部的层只对相邻的层可见。这种风格支持基于可增加抽象层的设计。允许将一个复杂问题分解成一个增量步骤序列的实现。由于每一层最多只影响两层，同时只要给相邻层提供相同的接口，允许每层用不同的方法实现，同样为软件重用提供了强大的支持。

这种风格的优点:(1)支持基于抽象程度递增的系统设计，使设计者可以把一个复杂系统按递增的步骤进行分解;(2)支持功能增强，因为每一层至多和相邻的上下层交互，因此功能的改变最多影响相邻的上下层;(3)支持重用。可以定义一组标准的接口，允许各种不同的实现方法。

这种风格的缺点:(1)并不是每个系统都可以很容易地划分为分层的模式;(2)很难找到一个合适的、正确的层次抽象方法。

这种风格的典型应用有分层的通讯协议，如TCP/IP协议等

客户/服务器风格：是基于资源不对等，且为实现共享而提出来的，是20世纪90年代成熟起来的技术，客户机1服务器结构将应用一分为二，服务器(后台)负责数据管理，客户机(前台)完成与用户的交互任务。C/S体系结构具有强大的数据操作和事务处理能力模型思想简单，易于人们理解和接受。   现在使用较广泛的是三层C/S结构。

在客户端与数据库服务器之间增加了一个中间层 – 第一层：用户界面—表示层 – 第二层：业务逻辑—功能层 – 第三层：数据库—数据层

三层C/S结构具有以下优点:(1)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，从而使整个系统的逻辑结构更为清晰，能提高系统和软件的可维护性和可扩展性。(2)允许更灵活有效地选用相应的平台和硬件系统。(3)三层C/S结构中，应用的各层可以并行开发，各层也可以选择各自最适合的开发语言。(4)为严格的安全管理奠定了坚实的基础。这种风格可以应用在远程文件系统中。

面向对象风格：

系统被看作对象的集合，每个对象都有一个它自己的功能集合；数据及作用在数据上的操作被封装成抽象数据类型(ADT)；只通过接口与外界交互，内部的设计决策则被封装起来。

– 构件：类和对象

– 连接件：对象之间通过函数调用、消息传递实现交互

  这种风格的特点:(1)数据抽象是指对每一类对象进行概括，抽出这类对象的公共性质并用计算机语言加以描述的过程，把具有相同属性和相同操作的一些对象抽象为 一个类，这些对象都是这个类的实例。(2)封装是面向对象风格的又一个特点，它是一种信息隐减技术，通过封装，可以将一部分属性和操作隐藏起来，不让使用者访问，另一部分作为类的外部接口，使用者可以访问。(3)继承是新建的类从已有的类那里得到已有的特征，继承有效地实现了软件的重用，增强了系统的可扩充性。

  这种风格的缺点::(1)为了使一个对象和另一个对象通过过程调用等进行交互，必须知道对象的标识。这样就增强了对象之间的依赖关系，降低了独立性。(2)必须修改所有显式调用它的其它对象，并消除由此带来的一些副作用。

这种风格由于其具有的优势，近年来得到了广泛的应用，如飞行模拟系统的设计、CORBA等。

基于事件的隐式调用：

基于事件的隐式调用风格的思想是构件不直接调用一个过程，而是触发或广播一个或多个事件。系统中的其它构件中的过程在一个或多个事件中注册，当一个事件被触发，系统自动调用在这个事件中注册的所有过程，这样，一个事件的触发就导致了另一模块中的过程的调用。

– 调用过程与次序不是固定的、预先未知；

– 各构件之间通过事件的方式进行交互；

   隐式调用系统的优点有:(1)为软件重用提供了强大的支持。任何构件只需将它注册到系统的事件中，就可以加人到现存的系统中。(2)为改进系统带来了方便。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。

  隐式调用系统的缺点有:(1)构件无法控制系统所执行的计算。(2)数据交换的间题。有时数据可被一个事件传递，在这些情况下，整体性能和资源管理便成了问题。(3)因为过程的语义必须依赖于被触发事件的上下文约束，所以正确性推理业可能出问题。   支持基于事件的隐式调用的应用系统很多。例如，在编程环境中用于集成各种工具;在数据库管理系统中确保数据的一致性约束等等。

总结：

* 一个系统不止具有一种体系结构风格，在系统的不同层次不同抽象级别上，可以具有多种风格。虽然系统组织方式是无穷的，但是如果多人共同合作开发，应该尽量吧组织方式限制在少量的风格种类中，这样可以增强重用性、有利于系统分析，缩短选择时间、方便合作。
* 将软件体系结构分成不同风格指导软件开发活动，有利于提高软件的质量和开发效率。