（1）管道过滤器风格：功能模块称为过滤器；连接部分称为管道；过滤器的相对独立性；优点（易于理解；支持功能模块复用；较强可维护和可拓展性；支持特定分析；并发性）；不足（处理过程成批操作；输入输出特殊处理导致复杂性；交互能力弱）
（2）主程序/子程序风格：组件为主程序和子程序；连接件为调用返回机制；层次化结构；优点（数据访问效率高；计算功能分开）；缺点（缺少应变能力；难以复用）
（3）面向对象风格：对象负责维护其表示完整性；对象之间是隐蔽的；优点（隐藏实现细节具有安全可靠性；数据存取操作分解）；缺点（对象之间交互时必须知道其他对象的标识）
（4）层次化风格：分层系统中，每一层次由一系列组件组成；层次之间存在接口；优点（支持复杂问题分解；支持扩展和重用）；缺点（使用困难；难以定义合适抽象层次）
（5）事件驱动风格：隐式调用；优点（组件之间关联弱；提高复用能力；便于系统升级）；缺点（组件放弃计算控制权完全由系统决定；数据交换问题；正确性验证问题）
（6）解释器风格：虚拟机；优点（可移植性强；对未来硬件进行模拟仿真降低成本）；缺点（性能下降）
（7）基于规则的系统风格：业务逻辑=固定业务逻辑+可变业务逻辑（规则）+规则引擎；优缺点和（6）类似
（8）仓库风格：仓库存储和维护数据；优点（便于数据共享；方便模块添加、更新和删除；避免重复储存）；缺点（增加同步机制保证完整性和一致性）
（9）黑板系统风格：由知识源、黑板数据结构和控制器组成；对黑板数据结构可进行协同操作；优点（便于大量共享数据；便于添加和拓展；知识源重用；支持容错性和健壮性）；缺点（黑板数据结构修改困难；同步机制）
（10）C2 风格：并行组件网络；交互通过异步消息机制；优点（组件重用和替换容易实现；一定拓展能力；组件不需要共享地址空间；实现多用户多系统交互；动态更新系统框架结构）；缺点（不适合大规模流式风格系统以及对数据库的频繁使用）
（20）模型-视图-控制器风格：广泛用于用户交互式程序设计；包括模型（核心数据和处理问题）、视图（交互界面）和控制器（处理交互）三部分；优点（便

（11）客户机/服务器风格：两层 C/S 架构；优点（利于分布式数据组织和处理；组件之间位置相互透明；便于实现异构环境和多种开发技术融合；灵活性和可拓展性；降低整体成本）；缺点（开发成本高；客户机设计复杂负荷中；信息内容形式单一；维护成本增加；难以扩展到因特网；数据安全性不高）；三层 C/S 架构；相比于两层的优点（逻辑清晰提高可维护性和可扩充性；良好的负荷处理能力和较好的开放性;服务器和客户机可分别选择合适语言并发开发;较高安全性）
（12）浏览器/服务器风格：三层 C/S 风格的一种实现方式；包括浏览器、Web 服务器和数据库服务器；优点（操作简单；跨平台通信；开发成本低）；缺点（个性化程度低;客户端数据处理能力差;动态交互性不强;可拓展性差;安全性差;访问数据库速度慢）
（13）平台/插件风格：程序主体为平台；功能扩充为插件；优点（降低模块之间依赖性；系统模块独立；动态需求安装插件更灵活）；缺点（可重用性差）
（14）面向 Agent 风格：Agent 具有自主性、智能性和交互性；优点（便于解决复杂问题尤其是分布开放异构环境）；缺点（Agent 自身缺乏社会性结构描述和与环境的交互）
（15）面向方面架构风格：对横切关注点进行模块化设计；优点（可定义跨模块跨对象的交叉关系；可读性和易于维护；与面向对象编程互补）
（16）面向服务架构风格：服务之间定义良好接口和契约联系在一起；优点（灵活；支持复用；以业务为核心）；缺点（服务划分困难；接口不标准带来额外开销和不稳定性；接口难以统一；只限于不带界面的服务共享）
（17）正交架构风格：按照功能正交相关性垂直分割成子系统；有一个公共驱动层（最高层）和公共数据层（最低层）；优点（结构清晰易于理解；可维护性强；可移植性强）；缺点（并非所有软件系统都能正交化或正交成本太高）
（18）异构风格：多种风格组合；优点（实现遗留代码重用；解决标准不同问题）；缺点（兼容问题）
（19）基于层次消息总线的架构风格：层次消息总线、支持组件的分布和并发；组件之间通过消息总线通讯；优点（降低构建耦合性增强重用性；支持运行时系统演化，体现在动态增删构建、动态改变响应消息和消息过滤）；缺点（重用性差）
于维护；可移植性；系统三部分独立）；缺点（增加了设计和运行的复杂性；视图和控制器过于紧密妨碍重用；视图访问效率低；频繁访问未变化数据降低系统性能）