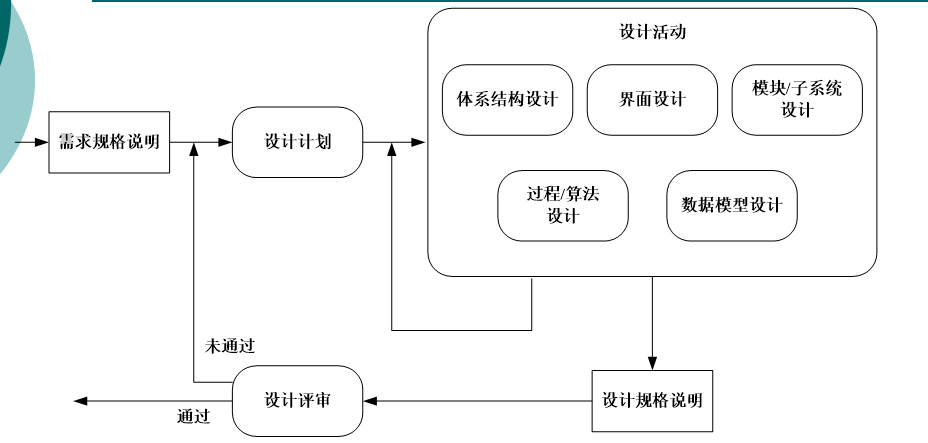
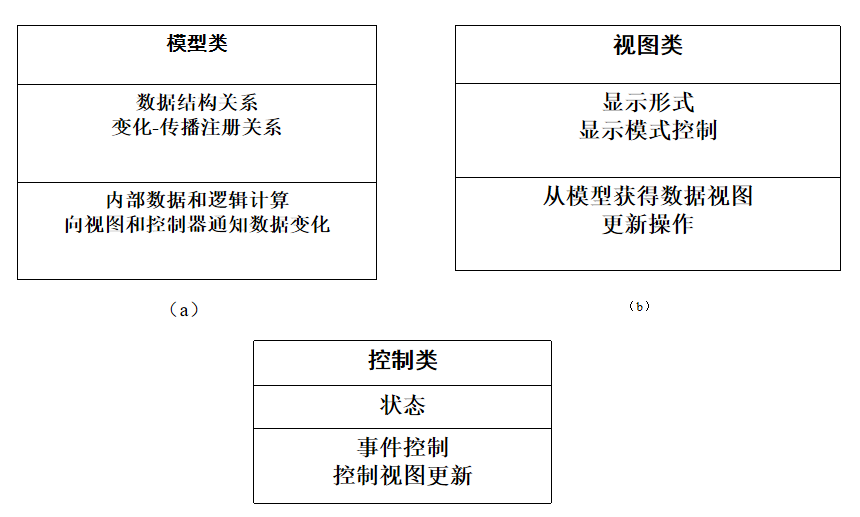
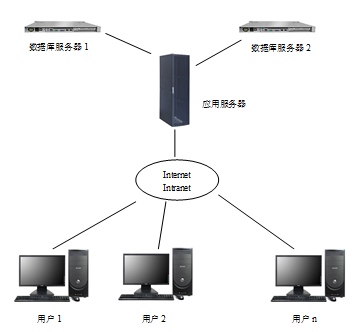
1. 构件：在计算机科学中指一个系统或软体中的基本组成部分。软件的构件应该被设计、实现成可在不同程序中复用的组件。优点：现代的可复用构件封装了数据和数据处理，使得软件工程师能够利用它来构造新的应用程序。
2. 软件过程的概念：工作产品构建时一系列活动、动作、任务的集合。
3. 一个通用的软件工程过程框架通常包含5个活动：沟通、策划、建模、构件、部署。
4. 软件设计原则：①抽象（是人类处理复杂问题的基本方法之一），包括过程抽象、数据抽象，②体系结构，③模式，④模块化，⑤信息隐蔽，表明：模块具备的特征是对其他所有模块都隐蔽自己的设计策略，⑥功能独立，⑦求精，⑧重构，在不改变软件现有功能的基础上，通过调整代码改善软件的质量、性能，使程序的设计模式和架构更趋于合理，从而提高软件的扩展性、维护性，⑨设计类.
5. 设计过程的目标：（1）必须实现所有分析模型中的明确需求，满足客户的所有隐性需求。（2）对于生成代码、进行测试、维护软件的人，设计必须是可读可理解的指南。（3）必须实现软件的全貌，并从实现角度说明数据域、功能域、行为域。
6. 软件系统的结构，包含软件元素、软件元素外部可见的属性以及这些软件元素之间的关系
7. 软件系统的基本组织，包含构件、构件之间、构件和环境之间的关系，以及相关的设计与演化原则等。
8. 软件体系结构作用：设计人员或者项目管理者能在一个比较高的层次上俯瞰软件的整体状况。它可以复用，减少开发成本，降低软件开发风险，例如在设计、实现、测试、评估、维护、升级中的潜在问题。
9. 功能模型：也称为过程模型或函数模型，基本原理是将一个系统分解为若干过基本功能模块，基本功能模块之间可以相互调用。核心1：基本功能模块的抽象及耦合，核心2：递归思想的具体实现
10. 对象模型：强调了对程序中数据组织的抽象，并将数据处理和数据组织统一进行考虑。以对象为核心，通过对象进行数据组织的抽象并实现数据组织和数据处理的统一，并在此基础上建立面向对象的软件构造方法。基本原理是将一个系统分解为若干个对象。核心1：对数据类型的抽象。核心2：同构（或同族）对象关系的定义
11. 

图 1 软件设计的一般过程框架

1. 管道-过滤器风格：适用于对有序数据进行一系列已经定义的相互计算的应用程序。管道过滤器模式下，每个功能模块都有一组输入和输出。功能模块从输入集合读入数据流，并在输出集合产生输出数据流，即功能模块对输入数据流进行增量计算得到输出数据流。管道过滤器模式下，功能模块称作过滤器（filter）；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（pipe）。
2. 基于事件的隐式调用风格：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。系统中的其他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。
3. 分层系统风格：系统按照层次结构组织，每一层向它的上层提供服务，同时又是它的下层客户；在某些系统中，除了邻接的层，一个内部层次对于其他外部层次是隐藏的，对体系结构的约束包括把系统内的交互限制在邻接层次之间。交互只在相邻的层间发生；同时，这些交互按照一定协议进行。相邻层的部件可直接从上向下调用，还可以设计统一的层调用接口对层进行保护。
4. 模型-视图-控制器风格：



1. 三层C/S结构风格：



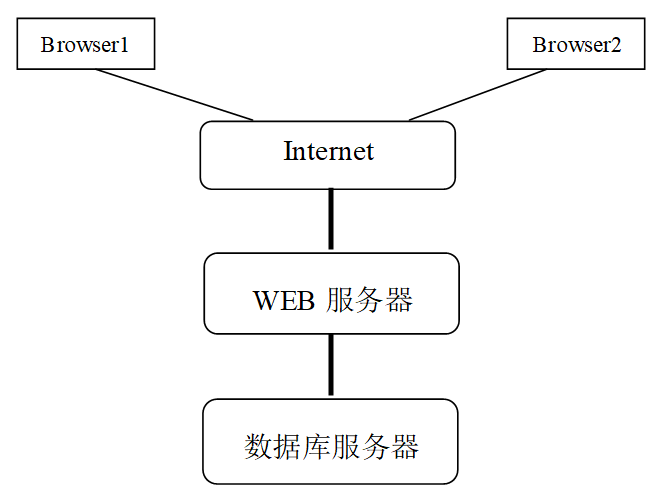
其将应用功能分成表示层、功能层、数据层3部分

表示层：是应用的用户接口。负责用户与应用间的对话功能，检查应用从键盘输入的数据，显示应用输出的数据。

功能层：相当于应用的本体，用于将具体的业务处理逻辑编入程序

数据层；数据库管理系统，负责管理对数据的读写

1. B/S风格：第一层客户机的主体是浏览器，第二层web服务器实现B/S结构的关键，第三层数据库机服务器的任务类似于C/S模式，负责协调不同web服务器发出的请求来管理数据库。



1. 设计模式实现原则：（1）.单一原则：一个对象只负责完成一个职责；高内聚，低耦合；（2）.开闭原则：对扩展开放，对修改关闭；对类的改动通过增加代码实现，而不是通过修改代码。（3）.里氏替换原则：任何父类对象都可使用子类进行替换；（4）.依赖注入原则：依赖于抽象，不依赖于具体实现（面向接口编程）（5）.接口分离原则：一个接口不要提供过多的行为（6）.迪米特原则：一个对象对其他对象尽可能少的理解（降低耦合）

19、云计算服务模型：基础设施层：IaaS（基础），平台层：PaaS，应用程序层：SaaS