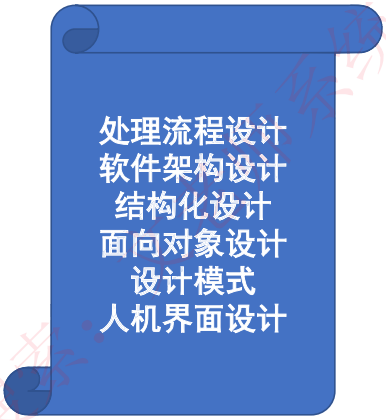
# 系统设计




文老师软考教育

## 1 处理流程设计

文老师软考教育

◆业务流程建模
标杆瞄准：以行业领先的标杆企业为目标，结合本企业情况分析建模。

IDEF（一系列建模、分析和仿真方法的统称）。

DEMO（组织动态本质建模法）

Petri网

业务流程建模语言：BPEL、BPML、BPMN、XPDL。

基于服务的BPM：基于Web服务的思想对业务流程进行建模。

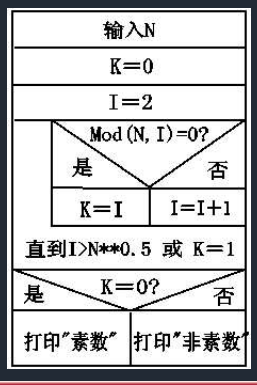
# 1 处理流程设计

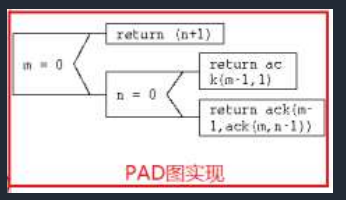


IDEF0：业务流程（功能）建模；
IDEF1：信息建模；
IDEF1X：数据建模（如ER模型）；
IDEF2：仿真建模设计；
IDEF3：过程描述获取；
IDEF4：面向对象设计；
IDEF5：本体论描述获取；
IDEF6：设计原理获取；
IDEF7：信息系统审计；
IDEF8：用户界面建模；
IDEF9：场景驱动信息系统设计；
IDEF10：实施架构建模；
IDEF11：信息制品建模；
IDEF12：组织结构建模；
IDEF13：三模式映射设计；
IDEF14：网络规划。

# 1 处理流程设计

文老师软考教育




◆流程表示工具
程序流程图（Program Flow Diagram，PFD）用一些图框表示各种操作，它独立于任何一种程序设计语言，比较直观、清晰，易于学习掌握。任何复杂的程序流程图都应该由顺序、选择和循环结构组合或嵌套而成。

IPO图也是流程描述工具，用来描述构成软件系统的每个模块的输入、输出和数据加工。

N-S图容易表示嵌套和层次关系，并具有强烈的结构化特征。但是当问题很复杂时，N-S图可能很大，因此不适合于复杂程序的设计。

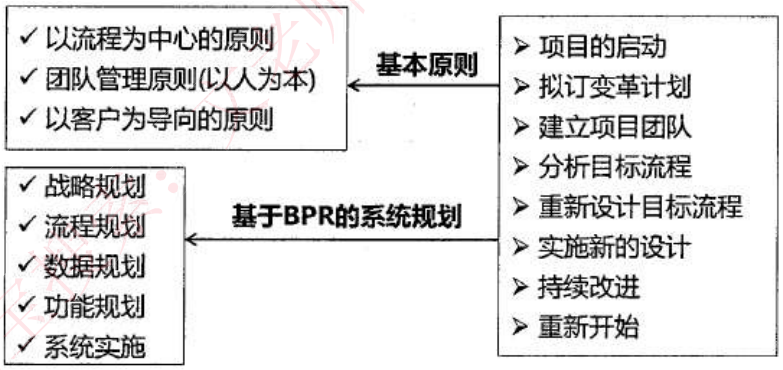
问题分析图（PAD）是一种支持结构化程序设计的图形工具。PAD具有清晰的逻辑结构、标准化的图形等优点，更重要的是，它引导设计人员使用结构化程序设计方法，从而提高程序的质量。

# 1 处理流程设计



◆业务流程重组BPR

BPR是对企业的业务流程进行根本性的再思考和彻底性的再设计，从而获得可以用诸如成本、质量、服务和速度等方面的业绩来衡量的显著性的成就。BPR设计原则、系统规划和步骤如下图所示：




# 1 处理流程设计

文老师软考教育

◆业务流程管理BPM

BPM是一种以规范化的构造端到端的卓越业务流程为中心，以持续的提高组织业务绩效为目的的系统化方法。

BPM与BPR管理思想最根本的不同就在于流程管理并不要求对所有的流程进行再造。构造卓越的业务流程并不是流程再造，而是根据现有流程的具体情况，对流程进行规范化的设计

流程管理包含三个层面：规范流程、优化流程和再造流程

# 考试真题



流程设计的任务是设计出系统所有模块和它们之间的相互关系，并具体设计出每个模块内部的功能和处理过程。以下关于流程设计的叙述，正确的是（ ）。
A.任何复杂的程序流程图都应该由顺序、选择、循环结构构成
B.IPO图不适合用来进行流程设计
C.PAD图是一种支持原型化设计方法的图形工具
D.N-S图容易表示嵌套关系和层次关系，特别适合于设计非常复杂的流程
答案：A

业务流程重组(Business Process Reengineering，BPR)是针对企业业务流程的基本问题进行回顾，其核心思路是对业务流程的（ ）改造，BPR过程通常以（ ）为中心。
A、增量式　　B、根本性　　C、迭代式　　D、保守式
A、流程　　B、需求　　C、组织　　D、资源
答案：B A

# 2 软件架构设计



◆架构设计就是需求分配，即将满足需求的职责分配到组件上。


◆软件架构为软件系统提供了一个结构、行为和属性的高级抽象，由构成系统的元素的描述、这些元素的相互作用、指导元素集成的模式以及这些模式的约束组成。
◆软件架构是项目干系人进行交流的手段，明确了对系统实现的约束条件，决定了开发和维护组织的组织结构，制约着系统的质量属性。
◆软件架构使推理和控制的更改更加简单，有助于循序渐进的原型设计，可以作为培训的基础。
◆软件架构是可传递和可复用的模型，通过研究软件架构可能预测软件的质量。

# 2 软件架构设计

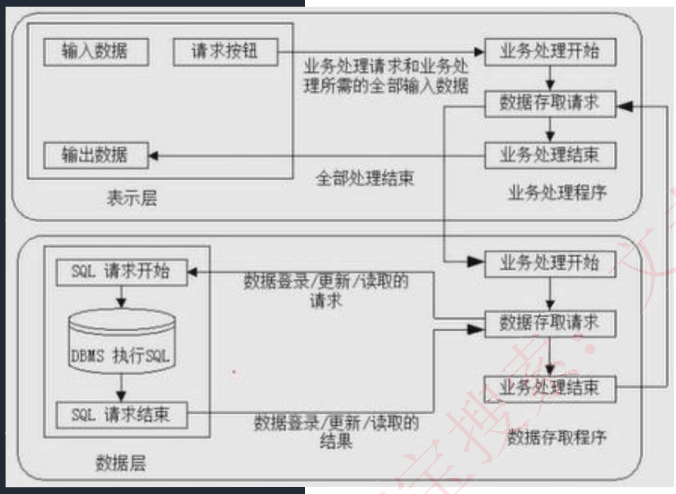


◆架构设计的一个核心问题是能否达到架构级的软件复用。
◆架构风格反映了领域中众多系统所共有的结构和语义特性，并指导如何将各个构件有效地组织成一个完整的系统。

有五种架构风格如下：
◆数据流风格：批处理序列（一个一个处理，顺序结构）、管道/过滤器（数据进出管道都要经过过滤器处理，分阶段的数据处理，常见于网络数据处理）。
◆调用/返回风格：主程序/子程序、面向对象风格、层次结构（与上下两层关系密切）。
◆独立构件风格：进程通信（构件之间是独立的，通过消息通信）、事件系统。
◆虚拟机风格：解释器（有虚拟机，可以仿真硬件执行过程，实现解释执行，效率较低，可跨平台）、基于规则的系统。
◆仓库风格：数据库系统、超文本系统、黑板系统（是一个解决方案的数据库，知识源与黑板进行通信，用于没有确定方案的系统）。

# 2 软件架构设计

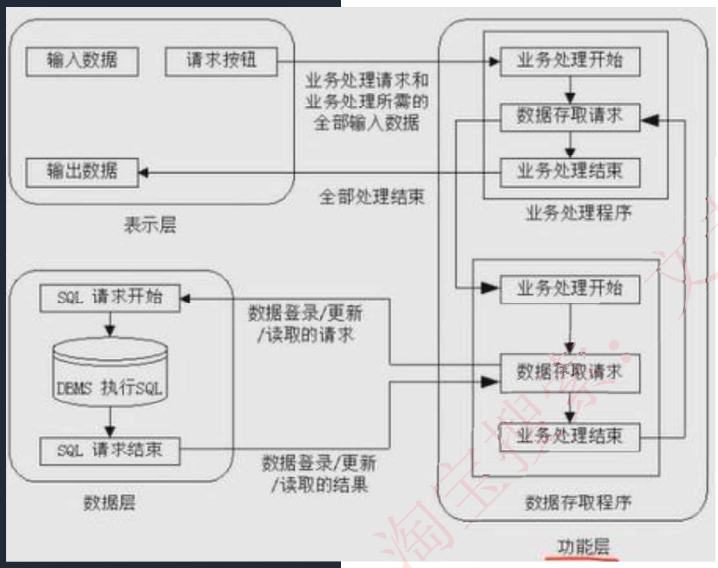
文老师软考教育




◆两层C/S架构
客户端和服务器都有处理功能，相比较于传统的集中式软件架构，还是有不少优点的，但是现在已经不常用，原因有：开发成本较高、客户端程序设计复杂、信息内容和形式单一、用户界面风格不一、软件移植困难、软件维护和升级困难、新技术不能轻易应用、安全性问题、服务器端压力大难以复用。

# 2 软件架构设计






◆三层C/S架构
◆将处理功能独立出来。表示层在客户机上，功能层在应用服务器上，数据层在数据库服务器上。
其优点下面四点：
◆各层在逻辑上保持相对独立，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；
◆允许灵活有效的选用相应的平台和硬件系统，具有良好的可升级性和开放性；
◆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；
◆功能层有效的隔离表示层与数据层，为严格的安全管理奠定了坚实的基础，整个系统的管理层次也更加合理和可控制。
◆三层C/S架构设计的关键在于各层之间的通信效率，要慎重考虑三层间的通信方法、通信频度和数据量，否则即使分配给各层的硬件能力很强，性能也不高。

# 2 软件架构设计

文老师软考教育

◆三层B/S架构
◆是三层C/S架构的变种，将客户端变为用户客户端上的浏览器，将应用服务器变为网络上的WEB服务器，又称为0客户端架构，虽然不用开发客户端，但有很多缺点，主要是数据处理能力差：

◆ B/S架构缺乏对动态页面的支持能力，没有集成有效的数据库处理功能；
◆安全性难以控制；
◆在数据查询等响应速度上，要远远低于C/S架构；
◆数据提交一般以页面为单位，数据的动态交互性不强，不利于OLTP应用。

# 3 结构化设计



◆系统设计主要目的：为系统制定蓝图，在各种技术和实施方法中权衡利弊，精心设计，合理地使用各种资源，最终勾画出新系统的详细设计方法。

◆系统设计方法：结构化设计方法，面向对象设计方法。

◆系统设计的主要内容：概要设计、详细设计。

◆概要设计基本任务：又称为系统总体结构设计，是将系统的功能需求分配给软件模块，确定每个模块的功能和调用关系，形成软件的模块结构图，即系统结构图。

◆详细设计的基本任务：模块内详细算法设计、模块内数据结构设计、数据库的物理设计、其他设计（代码、输入/输出格式、用户界面）、编写详细设计说明书、评审。

# 3 结构化设计

文老师软考教育

◆系统设计基本原理
抽象化；
自顶而下，逐步求精；
信息隐蔽；
模块独立（高内聚，低耦合）。

◆系统设计原则
保持模块的大小适中；
尽可能减少调用的深度；
多扇入，少扇出；
单入口，单出口；
模块的作用域应该在模块之内；
功能应该是可预测的。

# 考试真题



系统设计是根据系统分析的结果，完成系统的构建过程。系统设计的主要内容包括（ ）;系统总体结构设计的主要任务是将系统的功能需求分配给软件模块，确定每个模块的功能和调用关系，形成软件的（ ）。

A.概要设计和详细设计　　B.架构设计和对象设计
C.部署设计和用例设计　　D.功能设计和模块设计

A.用例图　　B.模块结构图　　C.系统部署图　　D.类图

答案：A　　B

以下关于软件系统模块结构设计的叙述中，正确的是（ ）。

A.当模块扇出过大时，应把下级模块进一步分解为若干个子模块
B.当模块扇出过小时，应适当增加中间的控制模块
C.模块的扇入大，表示模块的复杂度较高
D.模块的扇入大，表示模块的复用程度高

答案：D

# 3 结构化设计



◆**系统设计基本原理**：抽象、模块化、信息隐蔽、模块独立。

衡量模块独立程度的标准有两个：耦合性和内聚性。内聚程度从低到高如下表所示：

| 内聚分类 | 定义 | 记忆关键字 |
|---|---|---|
| 偶然内聚 | 一个模块内的各处理元素之间没有任何联系 | 无直接关系 |
| 逻辑内聚 | 模块内执行若干个逻辑上相似的功能，通过参数确定该模块完成哪一个功能 | 逻辑相似、参数决定 |
| 时间内聚 | 把需要同时执行的动作组合在一起形成的模块。 | 同时执行 |
| 过程内聚 | 一个模块完成多个任务，这些任务必须按指定的过程执行 | 指定的过程顺序 |
| 通信内聚 | 模块内的所有处理元素都在同一个数据结构上操作，或者各处理使用相同的输入数据或者产生相同的输出数据 | 相同数据结构、相同输入输出 |
| 顺序内聚 | 一个模块中的各个处理元素都密切相关于同一功能且必须顺序执行，前一个功能元素的输出就是下一个功能元素的输入 | 顺序执行、输入为输出 |
| 功能内聚 | 最强的内聚，模块内的所有元素共同作用完成一个功能，缺一不可 | 共同作用、缺一不可 |

# 3 结构化设计



**◆耦合程度从低到高如下表所示：**

| 耦合分类 | 定义 | 记忆关键字 |
|---|---|---|
| 无直接耦合 | 两个模块之间没有直接的关系，它们分别从属于不同模块的控制与调用，不传递任何信息 | 无直接关系 |
| 数据耦合 | 两个模块之间有调用关系，传递的是简单的数据值，相当于高级语言中的值传递 | 传递数据值调用 |
| 标记耦合 | 两个模块之间传递的是数据结构 | 传递数据结构 |
| 控制耦合 | 一个模块调用另一个模块时，传递的是控制变量，被调用模块通过该控制变量的值有选择的执行模块内的某一功能 | 控制变量、选择执行某一功能 |
| 外部耦合 | 模块间通过软件之外的环境联合（如 I/O 将模块耦合到特定的设备、格式、通信协议上）时。 | 软件外部环境 |
| 公共耦合 | 通过一个公共数据环境相互作用的那些模块间的耦合 | 公共数据结构 |
| 内容耦合 | 当一个模块直接使用另一个模块的内部数据，或通过非正常入口转入另一个模块内部时。 | 模块内部关联 |

# 考试真题



某模块中各个处理元素都密切相关于同一功能且必须顺序执行，前一处理元素的输出就是下一处理元素的输入，则该模块的内聚类型为（ ）内聚

A、过程　　B、时间　　C、顺序　　D、逻辑

答案：C

已知模块A给模块B传递数据结构X,则这两个模块的耦合类型为（ ）。

A.数据耦合　　B.公共耦合　　C.外部耦合　　D.标记耦合

答案：D

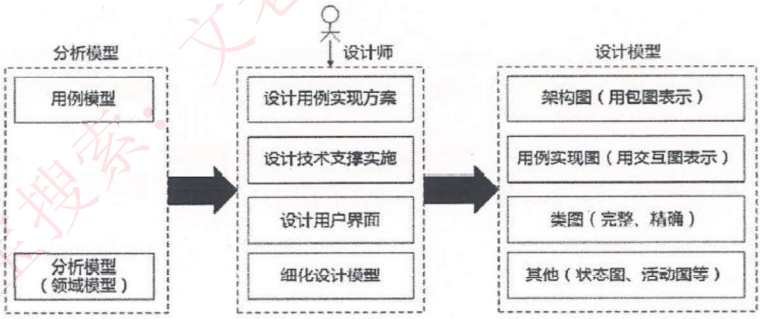
解析：特别说明是数据结构，不是数据，数据结构是标记耦合。

# 4 面向对象设计



◆**面向对象的设计**：是设计分析模型和实现相应源代码，设计问题域的解决方案，与技术相关。OOD同样应遵循抽象、信息隐蔽、功能独立、模块化等设计准则。

◆面向对象的分析模型主要由顶层架构图、用例与用例图、领域概念模型构成；设计模型则包含以包图表示的软件体系结构图、以交互图表示的用例实现图、完整精确的类图、针对复杂对象的状态图和用以描述流程化处理过程的活动图等。




# 4 面向对象设计

文老师软考教育

◆面向对象的设计原则：

（1）单一责任原则。就一个类而言，应该仅有一个引起它变化的原因。即，当需要修改某个类的时候原因有且只有一个，让一个类只做一种类型责任。

（2）开放 - 封闭原则。软件实体（类、模块、函数等）应该是可以扩展的，即开放的；但是不可修改的，即封闭的。

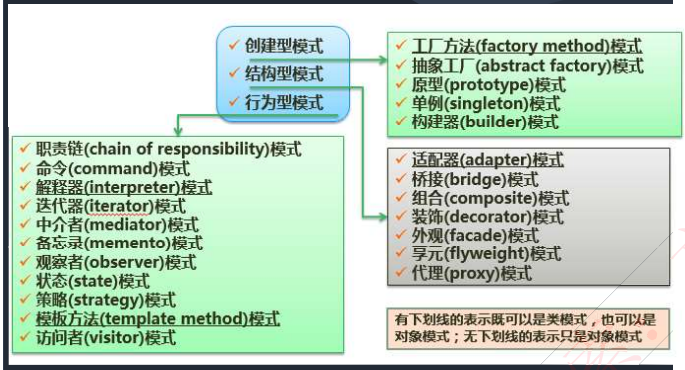
（3）里氏替换原则。子类型必须能够替换掉他们的基类型。即，在任何父类可以出现的地方，都可以用子类的实例来赋值给父类型的引用。

（4）依赖倒置原则。抽象不应该依赖于细节，细节应该依赖于抽象。即，高层模块不应该依赖于低层模块，二者都应该依赖于抽象。

（5）接口分离原则。不应该强迫客户依赖于它们不用的方法。接口属于客户，不属于它所在的类层次结构。即：依赖于抽象，不要依赖于具体，同时在抽象级别不应该有对于细节的依赖。这样做的好处就在于可以最大限度地应对可能的变化。

# 5.设计模式






◆架构模式：软件设计中的高层决策，例如C/S结构就属于架构模式，架构模式反映了开发软件系统过程中所作的基本设计决策。

◆设计模式：每一个设计模式描述了一个在我们周围不断重复发生的问题，以及该问题的解决方案的核心。这样，你就能一次又一次地使用该方案而不必做重复劳动。设计模式的核心在于提供了相关问题的解决方案，使得人们可以更加简单方便的复用成功的的设计和体系结构。四个基本要素：模式名称、问题（应该在何时使用模式）、解决方案（设计的内容）、效果（模式应用的效果）。

◆惯用法：是最低层的模式，关注软件系统的设计与实现，实现时通过某种特定的程序设计语言来描述构件与构件之间的关系。每种编程语言都有它自己特定的模式，即语言的惯用法。例如引用 - 计数就是C++语言中的一种惯用法。

# 5 设计模式

文老师软考教育

| 创建型设计模式 | 定义 | 记忆关键字 |
|---|---|---|
| Abstract Factory 抽象工厂模式 | 提供一个接口，可以创建一系列相关或相互依赖的对象，而无需指定它们具体的类 | 抽象接口 |
| Builder 构建器模式 | 将一个复杂类的表示与其构造相分离，使得相同的构建过程能够得出不同的表示 | 类和构造分离 |
| Factory Method 工厂方法模式 | 定义一个创建对象的接口，但由子类决定需要实例化哪一个类。使得子类实例化过程推迟 | 子类决定实例化 |
| Prototype 原型模式 | 用原型实例指定创建对象的类型，并且通过拷贝这个原型来创建新的对象 | 原型实例，拷贝 |
| Singleton 单例模式 | 保证一个类只有一个实例，并提供一个访问它的全局访问点 | 唯一实例 |

# 5 设计模式



| 结构型设计模式 | 定义 | 记忆关键字 |
| --- | --- | --- |
| Adapter<br>适配器模式 | 将一个类的接口转换成用户希望得到的另一种接口。它使原本不相容的接口得以协同工作 | 转换，兼容接口 |
| Bridge<br>桥接模式 | 将类的抽象部分和它的实现部分分离开来，使它们可以独立的变化 | 抽象和实现分离 |
| Composite<br>组合模式 | 将对象组合成树型结构以表示“整体-部分”的层次结构，使得用户对单个对象和组合对象的使用具有一致性 | 整体-部分，树形结构 |
| Decorator<br>装饰模式 | 动态的给一个对象添加一些额外的职责。它提供了用子类扩展功能的一个灵活的替代，比派生一个子类更加灵活 | 附加职责 |
| Façade<br>外观模式 | 定义一个高层接口，为子系统中的一组接口提供一个一致的外观，从而简化了该子系统的使用 | 对外统一接口 |
| Flyweight<br>享元模式 | 提供支持大量细粒度对象共享的有效方法 | 细粒度，共享 |
| Proxy<br>代理模式 | 为其他对象提供一种代理以控制这个对象的访问 | 代理控制 |

# 5 设计模式

文老师软考教育

| 行为型设计模式 | 定义 | 记忆关键字 |
| --- | --- | --- |
| Chain of Responsibility<br>职责链模式 | 通过给多个对象处理请求的机会，减少请求的发送者与接收者之间的耦合。将接收对象链接起来，在链中传递请求，直到有一个对象处理这个请求 | 传递请求、职责、链接 |
| Command<br>命令模式 | 将一个请求封装为一个对象，从而可用不同的请求对客户进行参数化，将请求排队或记录请求日志，支持可撤销的操作 | 日志记录、可撤销 |
| Interpreter<br>解释器模式 | 给定一种语言，定义它的文法表示，并定义一个解释器，该解释器用来根据文法表示来解释语言中的句子 | 解释器，虚拟机 |
| Iterator<br>迭代器模式 | 提供一种方法来顺序访问一个聚合对象中的各个元素而不需要暴露该对象的内部表示 | 顺序访问，不暴露内部 |
| Mediator<br>中介者模式 | 用一个中介对象来封装一系列的对象交互。它使各对象不需要显式地相互调用，从而达到低耦合，还可以独立的改变对象间的交互 | 不直接引用 |

# 5 设计模式



| 行为型设计模式 | 定义 | 记忆关键字 |
| --- | --- | --- |
| Memento<br>备忘录模式 | 在不破坏封装性的前提下，捕获一个对象的内部状态，并在该对象之外保存这个状态，从而可以在以后将该对象恢复到原先保存的状态 | 保存，恢复 |
| Observer<br>观察者模式 | 定义对象间的一种一对多的依赖关系，当一个对象的状态发生改变时，所有依赖于它的对象都得到通知并自动更新 | 通知、自动更新 |
| State<br>状态模式 | 允许一个对象在其内部状态改变时改变它的行为 | 状态变成类 |
| Strategy<br>策略模式 | 定义一系列算法，把它们一个个封装起来，并且使它们之间可互相替换，从而让算法可以独立于使用它的用户而变化 | 算法替换 |
| Template Method<br>模板方法模式 | 定义一个操作中的算法骨架，而将一些步骤延迟到子类中，使得子类可以不改变一个算法的结构即可重新定义算法的某些特定步骤 | |
| Visitor<br>访问者模式 | 表示一个作用于某对象结构中的各元素的操作，使得在不改变各元素的类的前提下定义作用于这些元素的新操作。 | 数据和操作分离 |

# 考试真题



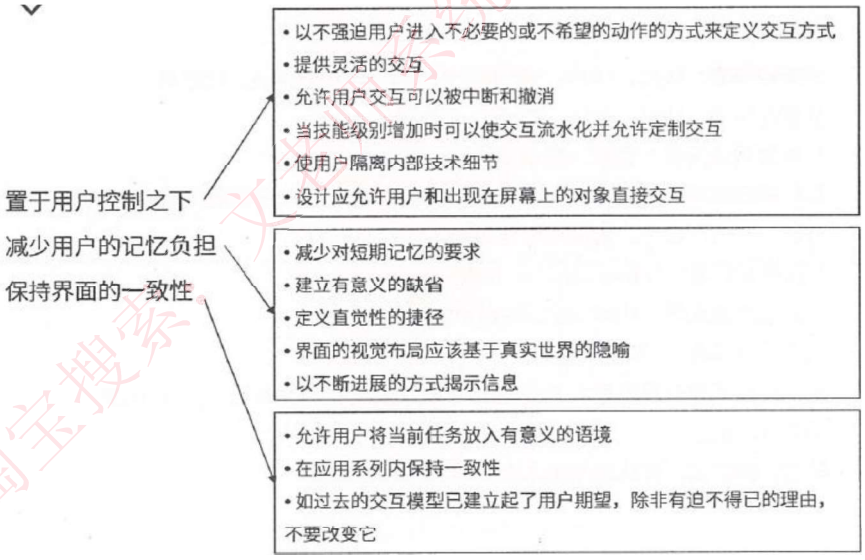
设计模式描述了一个出现在特定设计语境中的设计再现问题，并为它的解决方案提供了一个经过充分验证的 通用方案，不同的设计模式关注解决不同的问题。例如，抽象工厂模式提供一个接口，可以创建一系列相关或相互依赖的对象， 而无需指定它们具体的类，它是一种（54）模式；（55）模式将类的抽象部分和它的实现部分分离出来，使它们可以独立变化，它属于（56）模式；（57）模式将一个请求封装为一个对象，从而可用不同的请求对客户进行参数化，将请求排队或记录请求日志，支持可撤销的操作。

（54）A.组合型　B.结构型 C.行为型 D.创建型
（55）A.Bridge　B.Proxy　C.Prototype　D.Adapter
（56）A.组合型　B.结构型 C.行为型 D.创建型
（57）A.Command　B.Facade C.Memento　D.Visitor
答案：（54）D，（55）A，（56）B，（57）A

# 6 人机界面设计



◆人机界面设计三大黄金原则：




# 谢谢！