软件建模设计的原则

开闭原则

构件应该对外延具有开放性，对修改具有封闭性

设计者应该采用一种无需对构件自身内部做修改就可以进行扩展的方式来说明构件

动机：使得设计在发生变更时能够适应变更并且减少副作用的传播

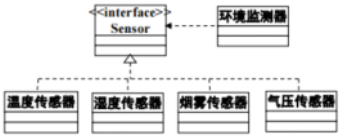
Liskov替换原则

子类可以替换它们的基类

源自基类的子类必须遵守基类与使用该基类的构件之间的隐含约定

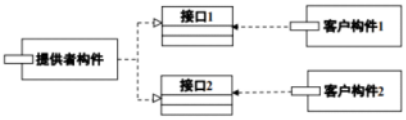
依赖倒置原则

依赖于抽象，而非具体实现



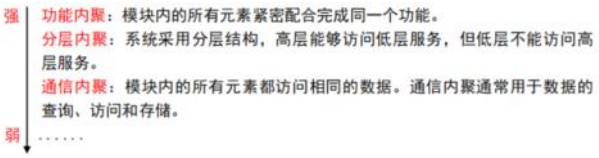
接口分离原则

多个客户专用接口比一个通用接口要好



内聚性

内聚性意味着构件或者类只封装那些相互联密切，以及与构件或类自身有密切关系的属性和操作。



内聚性越高，构件越易于实现、测试和维护

耦合性

构件级设计中要尽可能保持低耦合。

当两个对象必须相互联系时，应该通过类的公共接口实现耦合，不应该依类的具体实现细节

可重用性

尽量使用已有的类，包括开发环境提供的类库和已有的相似的类；
如果确实需要创建新类，则在设计这些新类时考虑将来的可重用性

软件静态结构建模类图

软件动态交互建模使用顺序图或通信图

软件状态机建模化状态机图

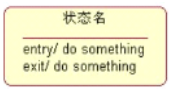
软件实现建模构件图部署图包图

UML软件静态结构视图建模

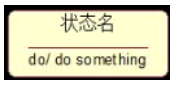
- 实体类：表示系统存储和管理的永久信息及其相关行为；
- 边界类：表示外部参与者与系统之间的交互，位于系统与外界的交界处，包括窗体、报表、系统硬件接口、与其他系统的接口等；
- 控制类：表示系统在运行过程中的业务控制逻辑，协调边界类和实体类。

状态机图

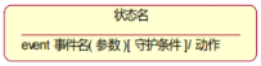
状态



入口动作指的是对象进入一个状态时所执行的动作，通常是内部初始化。
出口动作指的是对象退出一个状态时所执行的动作

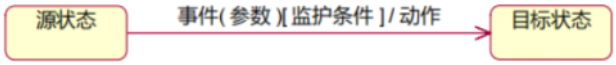


状态可以包含内部活动。当对象进入一个状态后，活动在入口动作完成后开始执行。如果活动结束，状态就完成，然后一个从这个状态出发的转换被触发

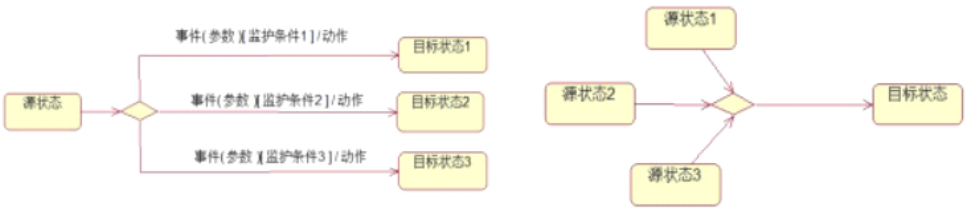


内部转换：状态可能包含一系列的内部转换，内部转换的结果并不改变状态本身。

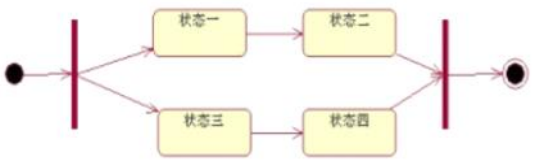
触发器事件



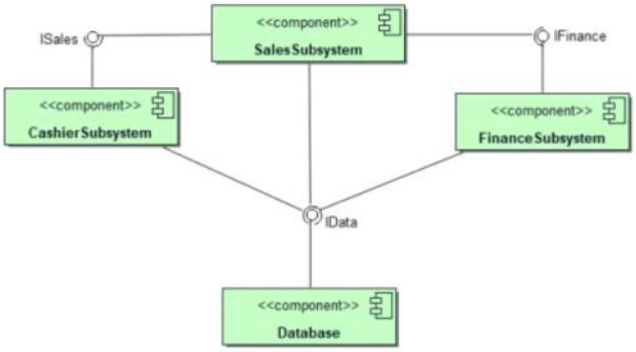
判定与合并



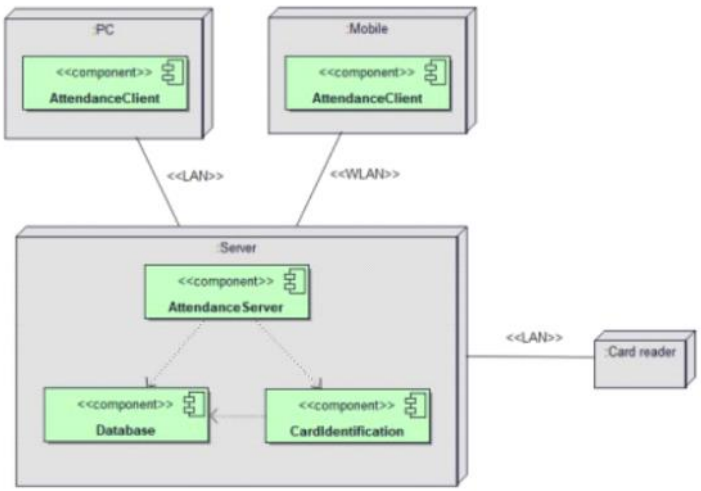
分叉与汇合



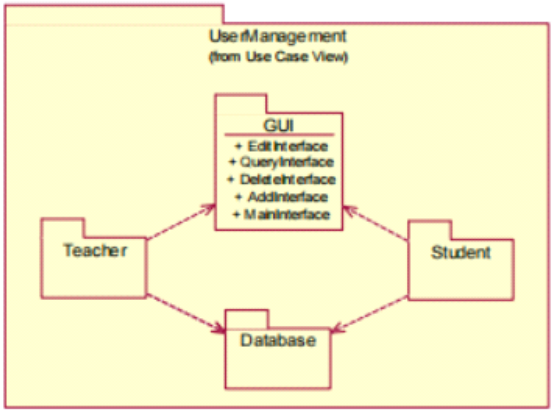
构件图


ISales
<<component>>
SalesSubsystem
IFinance
<<component>>
CashierSubsystem
<<component>>
FinanceSubsystem
IData
<<component>>
Database

部署图


PC
<<component>>
AttendanceClient
Mobile
<<component>>
AttendanceClient
<<LAN>>
<<WLAN>>
Server
<<component>>
AttendanceServer
<<component>>
Database
<<component>>
CardIdentification
<<LAN>>
Card reader

包图


UserManagement
(from Use Case View)
GUI
+ EditInterface
+ QueryInterface
+ DeleteInterface
+ AddInterface
+ MainInterface
Teacher
Student
Database