前面1950-2000年软工的发展

9、10

1体系结构的概念

软件体系结构模型由部件、连接件和配置这样的高层结构组织组成

一个软件系统的体系结构规定了系统的计算部件和部件之间的交互（高层抽象）

2风格

①主程序/子程序风格

基于声明-使用（程序调用）关系建立连接件

上层可以调用下层，不允许逆向调用 控制权在调用中转移给下层

优点：

流程清晰，易于理解

强控制性

缺点：

强耦合，非常依赖接口，系统难以修改和复用

程序调用的连接方式限制了各部件之间的数据交互，可能会使得不同部件使用隐含的共享数据流，产生不必要的公共耦合

②面向对象式风格

基于信息内聚建立对象部件，对外提供接口

每个对象负责维护自身数据的一致性与完整性，不同对象之间是平级

基于方法调用建立连接件

优点：

内部实现的可修改性

易开发、易理解、易复用

缺点：

接口的耦合性

标识的耦合性

副作用（方法调用）

③分层风格

从底层到高层，部件的抽象层次升高

两个层次之间的连接遵循成熟稳定的交互协议

上层调用下层 不能跨层调用、不能逆向调用

优点：

清晰、易理解

支持并行开发

更好的可复用性和内部可修改性

缺点：

难以修改交互协议

性能损失（逐层调用，不允许跨层）

难以确定层次数量和粒度

④MVC模型-视图-控制风格

优点：

易开发性

视图和控制的可修改性

适宜于网络系统开发的特征

缺点：

复杂性

模型修改困难

3体系结构的设计过程

\*\*\*包图等

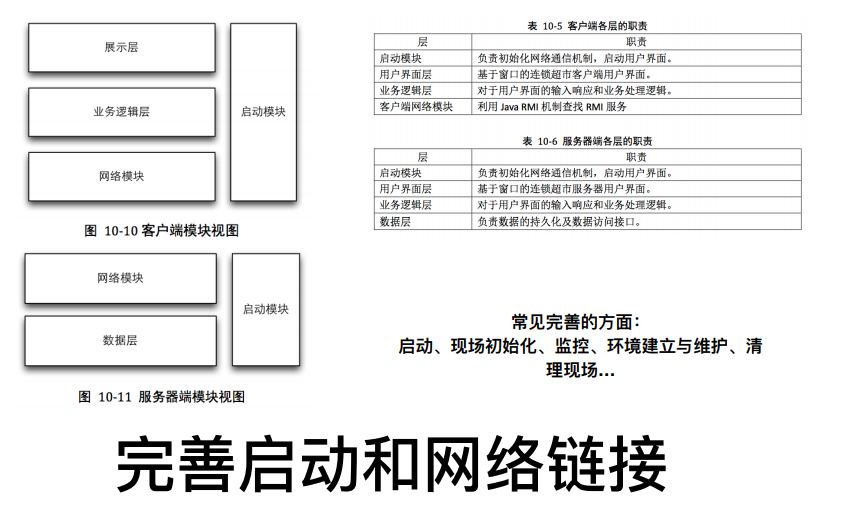
分析项目需求和约束

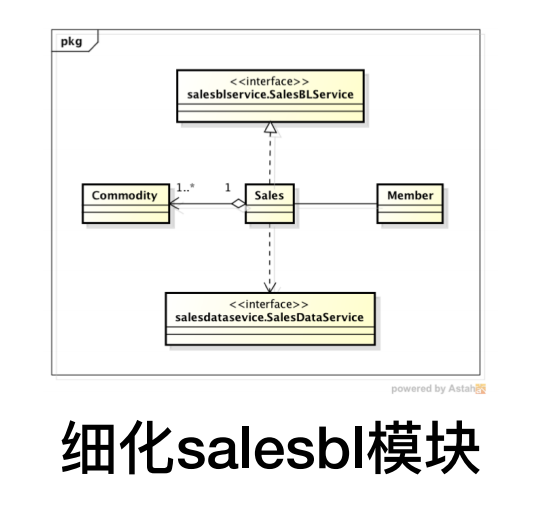
选择体系结构风格

逻辑设计 包图初步——>改进

软件体系结构实现（开发包，即物理包设计）！！！

完善体系结构





定义构件接口

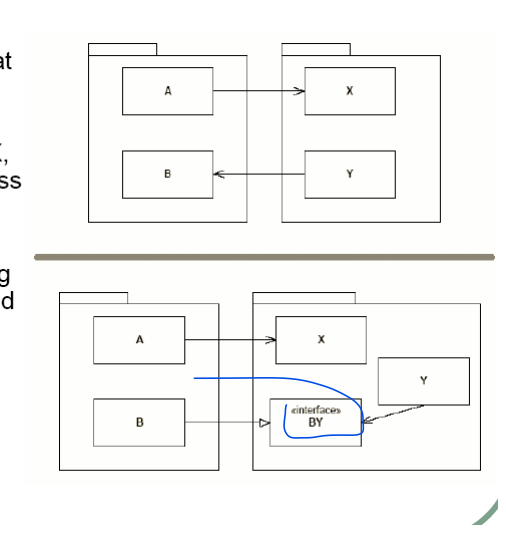
4包的设计原则：

内聚：CCP共同封闭原则（让包尽可能大）

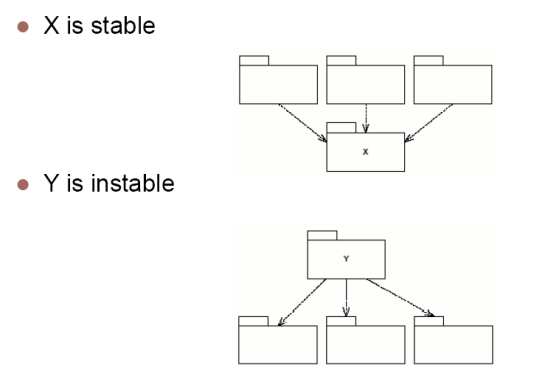
CRP 共同重用原则（让包尽可能小）

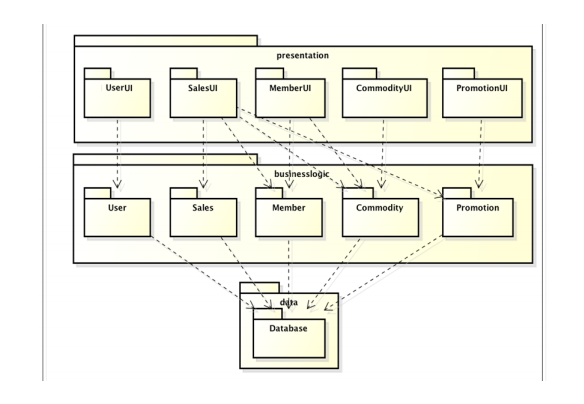
REP 重用发布等价原则

ADP**无环依赖原则**



SDP**稳定依赖原则**（包之间的依赖）





这个图第一层一个包依赖的包太多了，不符合SDP

SAP稳定抽象原则

稳定的包是抽象包

让包符合OCP（扩展开放，修改封闭）

用DIP依赖倒置原则，接口实现（67的黑板图）

5接口的定义？。。。

6Stub和Driver

11、

1可用性：

是多维度质量属性，包括

易学性、易记性、效率、出错率（用户使用会犯更少的错误）、主观满意度

2界面设计的原则

（可视化设计不要简单地把每一个对象方法设置为一个键、不应该表现内部细节，

还应该基于界面隐喻，尽可能把功能和任务细节表现出来）

①简洁设计（图片更好，太大的菜单和过多的信息类别都是不好的，不要用太多字体和颜色）

②一致性设计

③低出错率设计

④易记性设计

3导航、反馈

协作式设计：

调整计算机因素以更好地适应并帮助⽤户的设计⽅式被称为协作式设计

