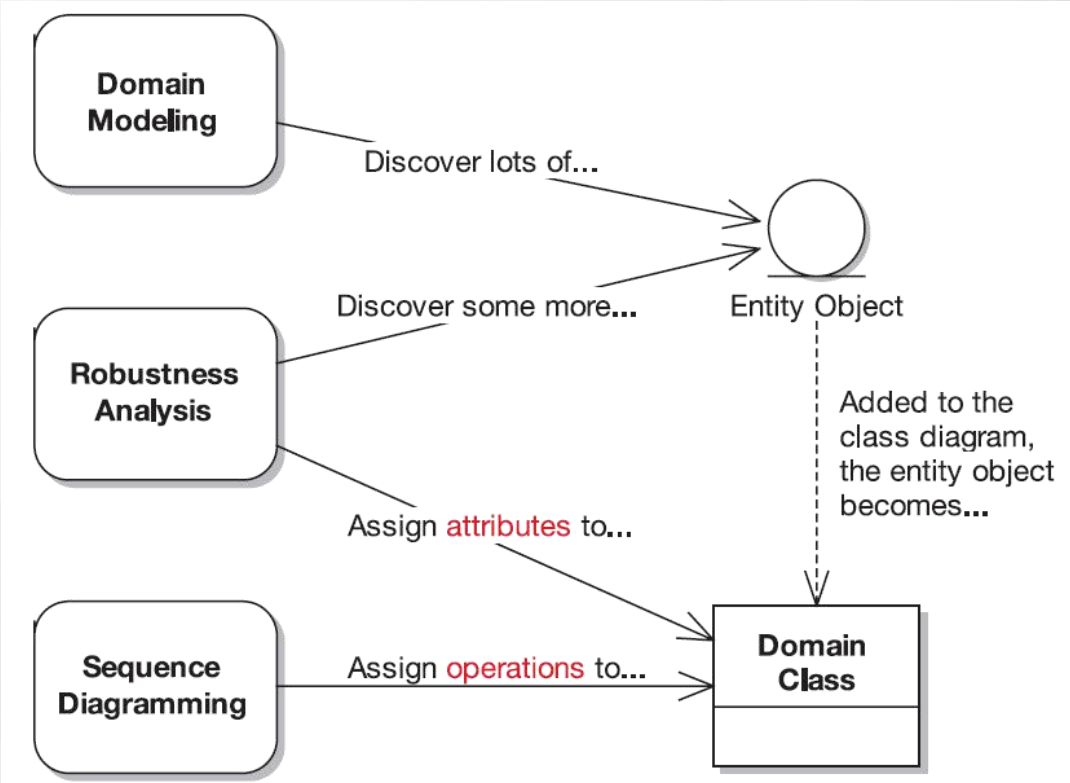
第七章　详细设计

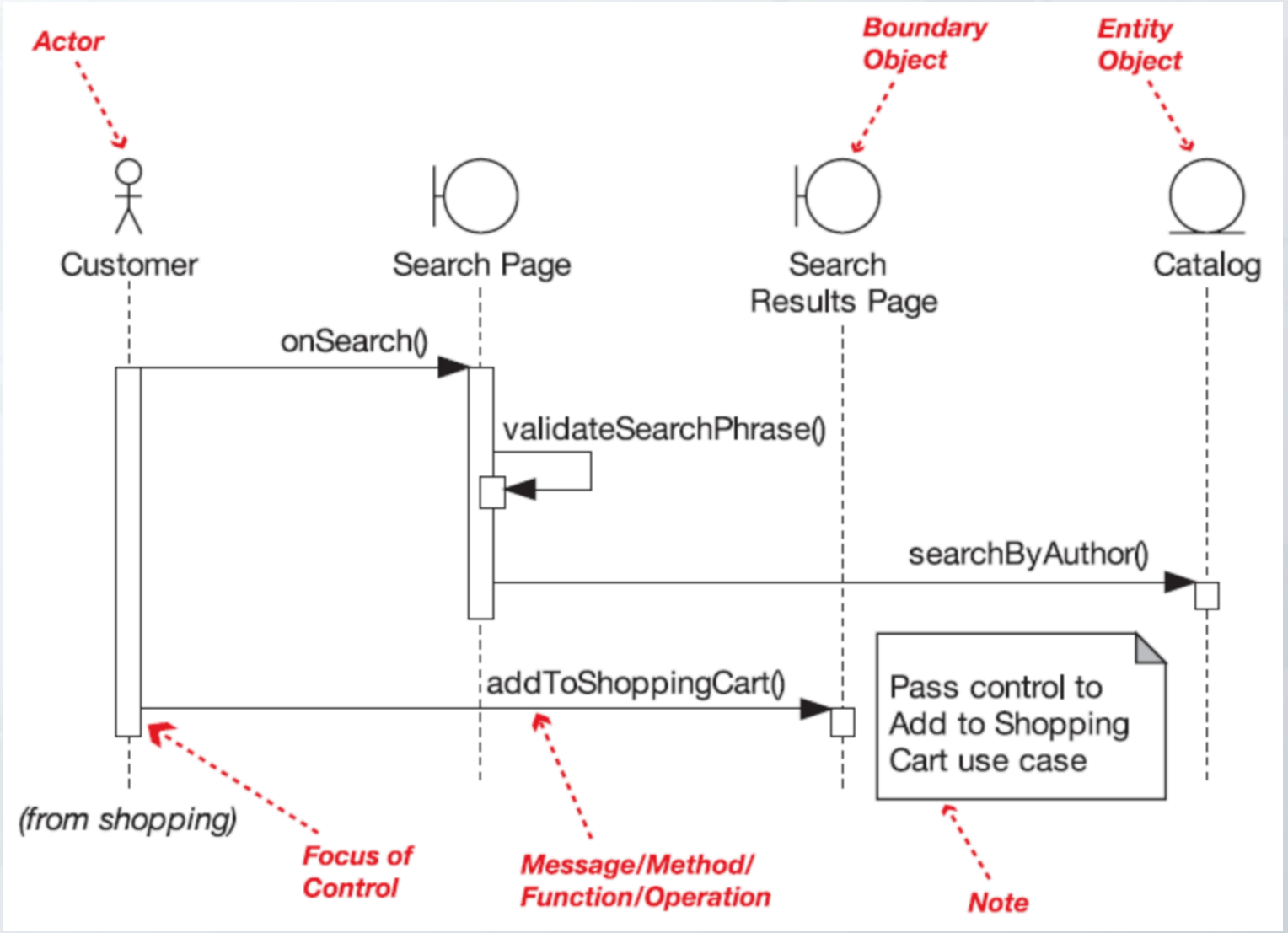
回顾：关键设计的方法和意义



主要意义：就是要通过寻找对象之间的交互关系，进而进行方法（操作或行为）分配

基于用例图、用例描述和健壮性图，采用序列图来描述参与者、边界、实体之间的交互.

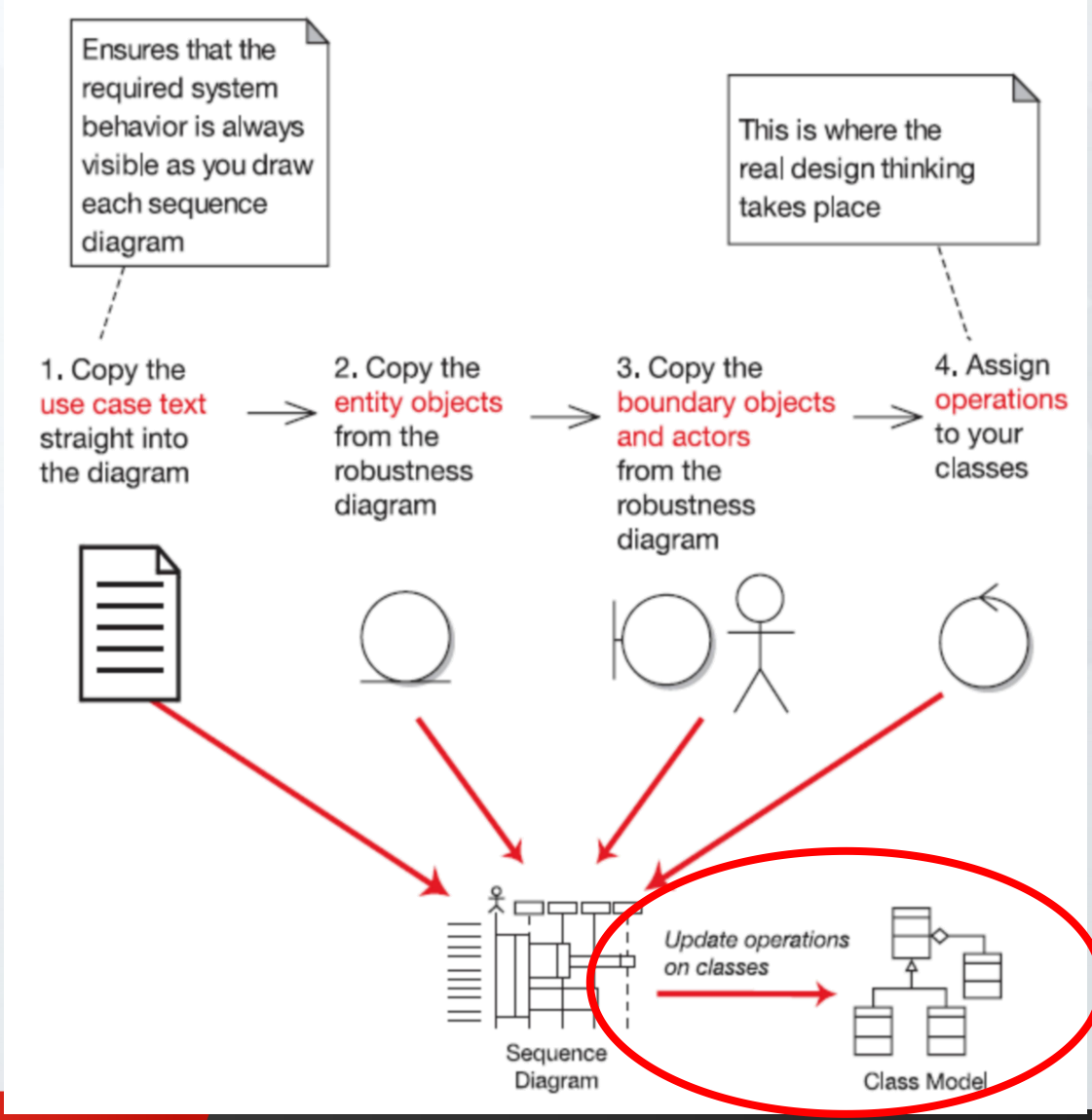
回顾：序列图[Sequence Diagram]的要素



注意：消息对应动词

当逻辑不清或业务逻辑逐渐复杂的情况下，绘制序列图是十分有意义的。

回顾：画出每个用例的序列图



1. 逐步贯穿健壮性图上的每一个控制器，每次一个，画出序列图上相应的方法，然后核对，移至下一个控制器。

2. 控制器和方法之间并不一定是完全一对一匹配的，也可能会转化为两个或多个方法。

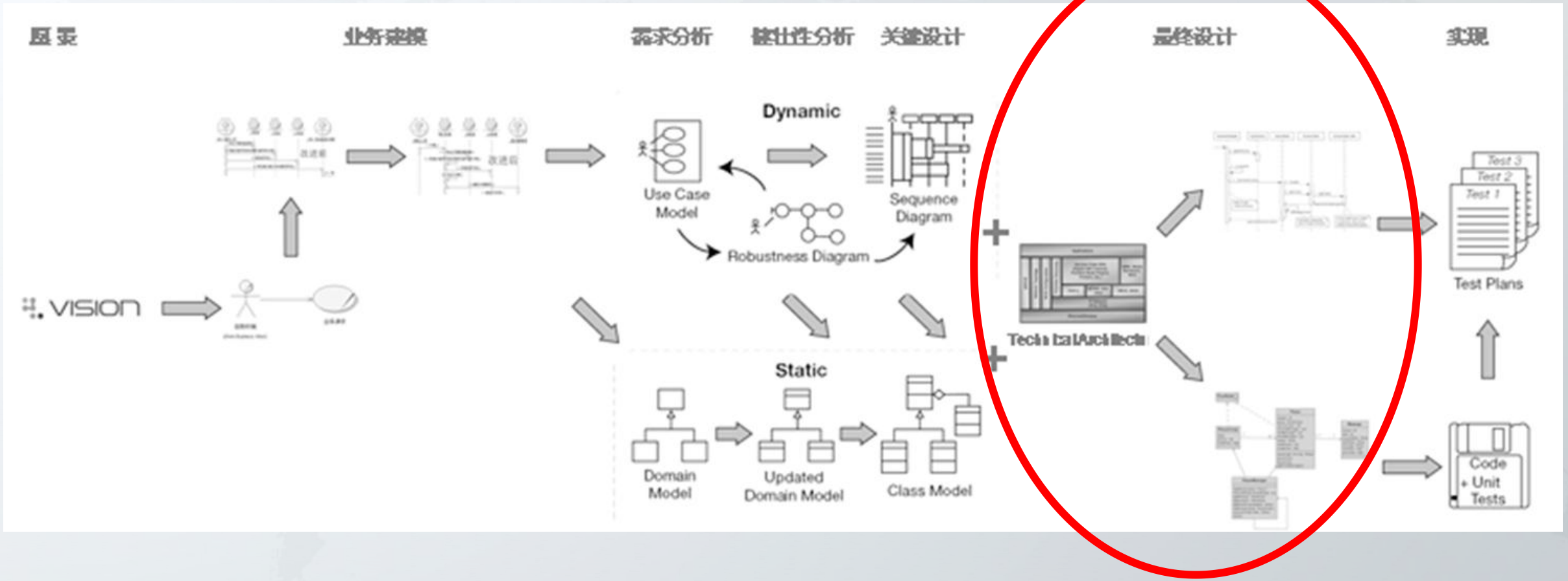
3. **有时，控制器也可能会转换为一个真正的控制类。**（例如：检查余额可以在账户类中，但跳转放在账户类中不合适，可以单独放到一个控制器类里）

4. 序列图会对类图做进一步的更新，完善其方法

目录

目录

详细设计


Dynamic
Use Case Model
Sequence Diagram
Robustness Diagram
VISION
Static
Domain Model
Updated Domain Model
Class Model
Test 3
Test 2
Test 1
Test Plans
Code + Unit Tests

自助银行系统的背景描述

- 环境现状
- 可靠性
- 可用性
- 性能
- 可支持性

环境现状

- 财神银行是一家城市性银行，业务范围仅覆盖天堂市市区。总行位于市区中心，同时在市区内分布有30家支行。支撑银行业务运转的中央银行系统位于总行，各支行的工作终端通过专线连接到中央系统上。
- 新的自助银行系统也要在每个分行安装1-3个终端，通过专线连接至总行的中央系统上。

可靠性

- 7*24小时不间断服务；
- 保障数据的完整安全；
- 保留半年以上交易记录，便于查询异常交易；
- 新系统不得对已有中央银行系统造成任何安全隐患；

可用性

- 用户完成任何一项业务，操作步骤不超过5步；
- 任何用户操作失误或系统错误都有友好的提示信息；

性能

- 系统支持的终端数为100个；
- 系统最大支持的业务量为1000笔/日；
- 每步响应时间<1秒；

可支持性

- 95%的系统故障可在两小时内解决；
- 系统的软件部分每年升级一次，并保持向下兼容；

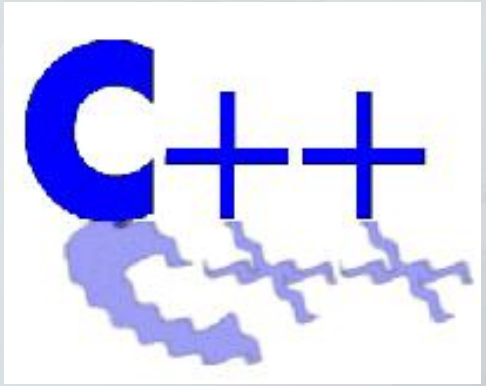
目录

技术架构及相关考虑

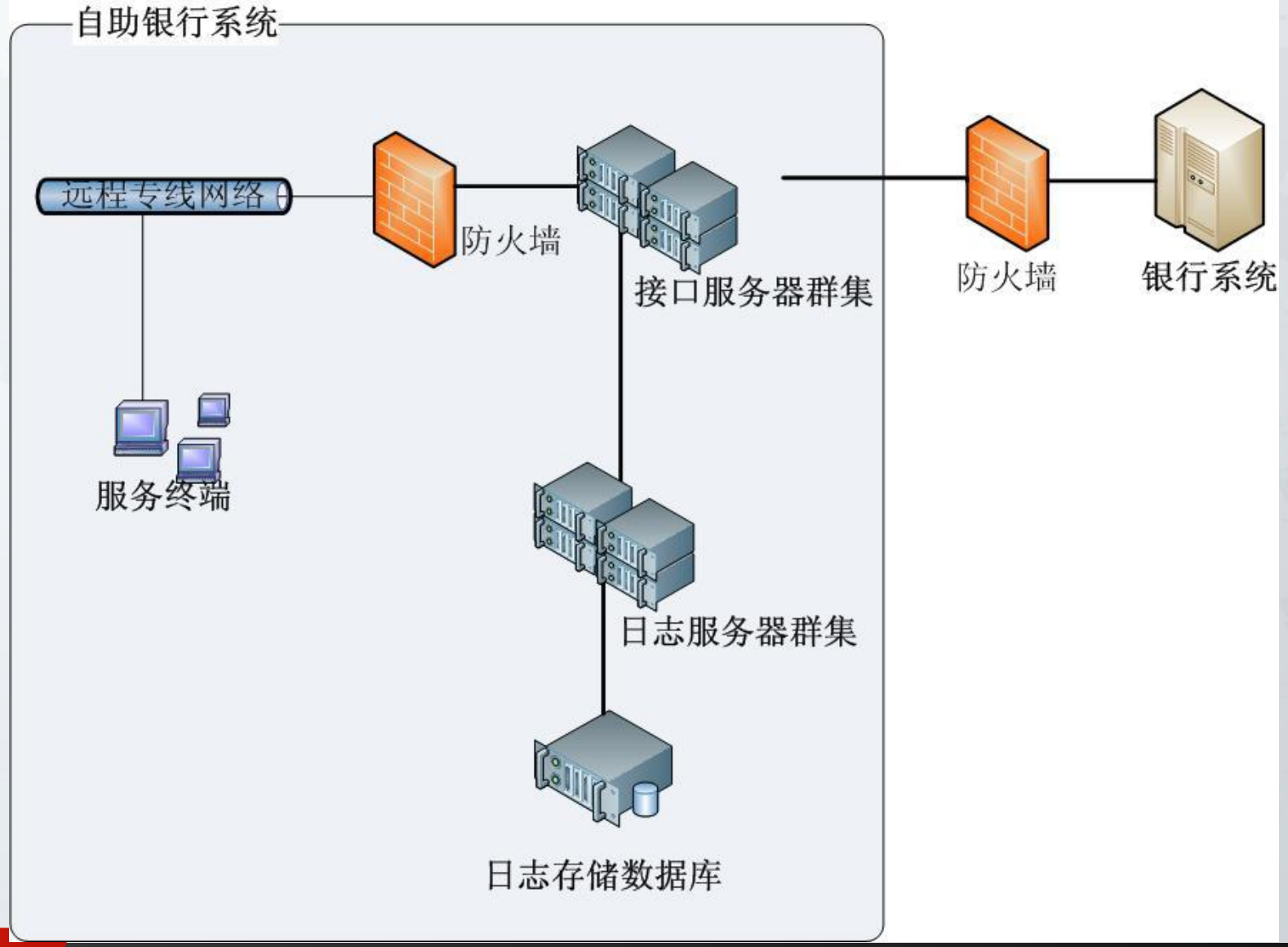
- 选择开发语言
- 网络拓扑及安全
- 体系结构
- 硬件支持环境
- 软件支持环境
- ...

选择开发语言

- 客观条件需要
- 客户要求
- 开发团队习惯
- ……

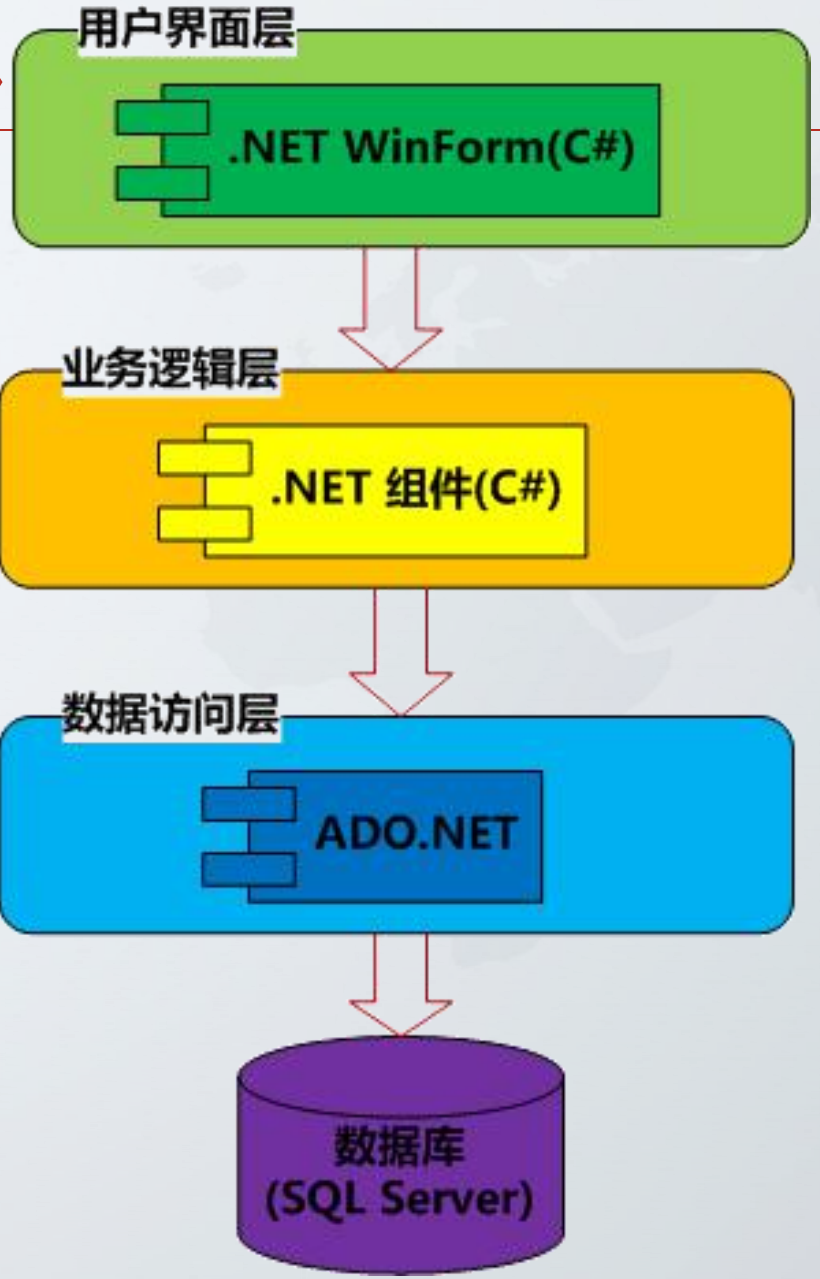


网络拓扑及安全



参见阿里云相关产品（部署图）

体系结构—三层



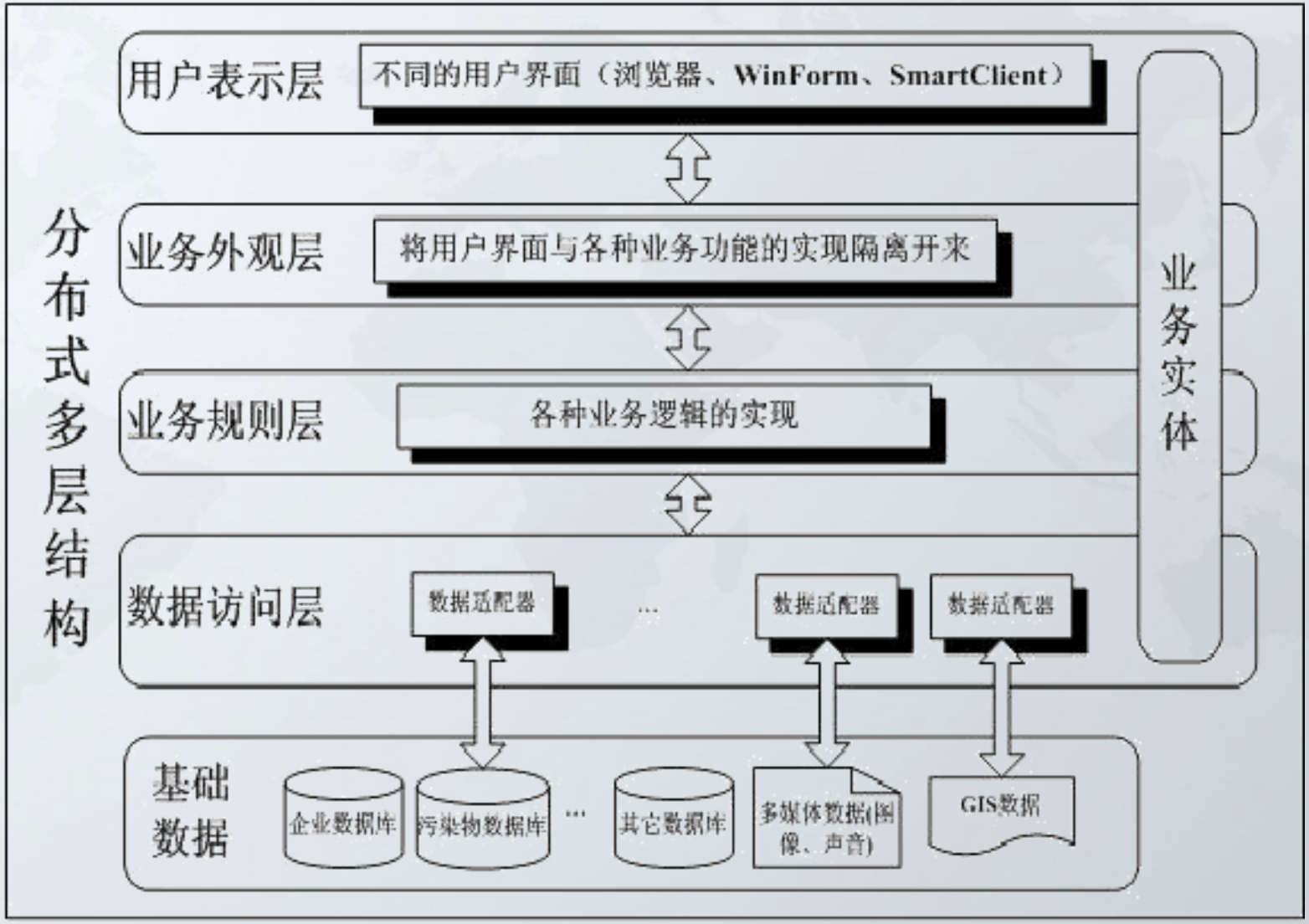
用户界面层（User Interface layer）
业务逻辑层（Business Logic Layer）、
数据访问层（Data access layer）

区分层次的目的即为了“高内聚低耦合”的思想。在软件体系架构设计中，分层式结构是常见和重要的一种结构。
优点：降低层与层之间的依赖 标准化

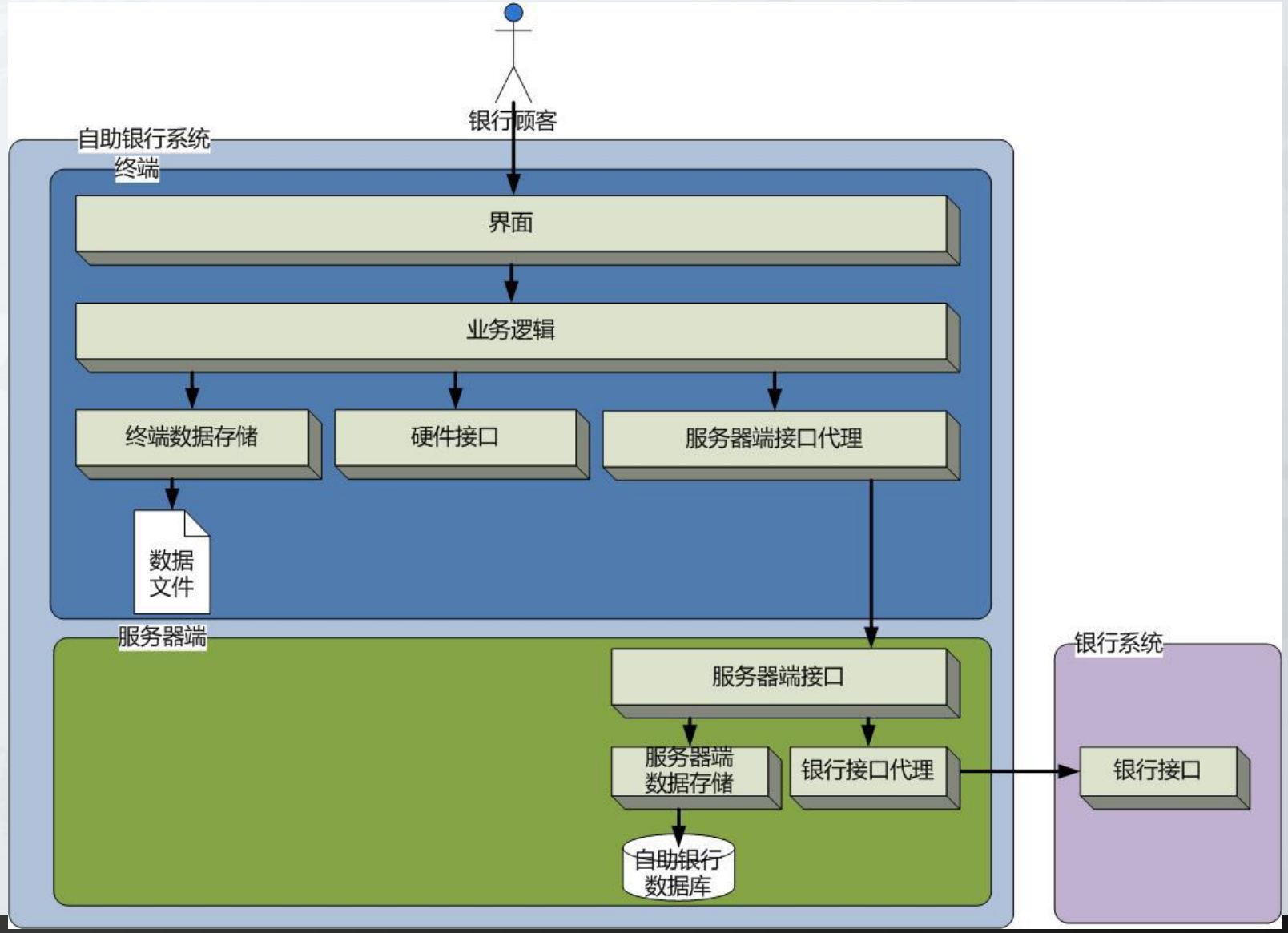
注意：数据访问层是对数据库的操作，而不是数据，具体为业务逻辑层或表示层提供数据服务。

MVC没有把业务的逻辑访问看成两个层，这是采用三层架构与MVC搭建程序最主要的区别。三层”中典型的Model层是由业务逻辑与访问数据组成的。而MVC里，则是以实体类构成的。

体系结构—多层


分布式多层结构
用户表示层
不同的用户界面（浏览器、WinForm、SmartClient）
业务外观层
将用户界面与各种业务功能的实现隔离开来
业务规则层
各种业务逻辑的实现
数据访问层
数据适配器
...
数据适配器
数据适配器
业务实体
基础数据
企业数据库
污染物数据库
...
其它数据库
多媒体数据(图像、声音)
GIS数据

ATM系统体系结构


银行顾客
自助银行系统
终端
界面
业务逻辑
终端数据存储
硬件接口
服务器端接口代理
数据
文件
服务器端
服务器端接口
服务器端
数据存储
银行接口代理
自助银行
数据库
银行系统
银行接口

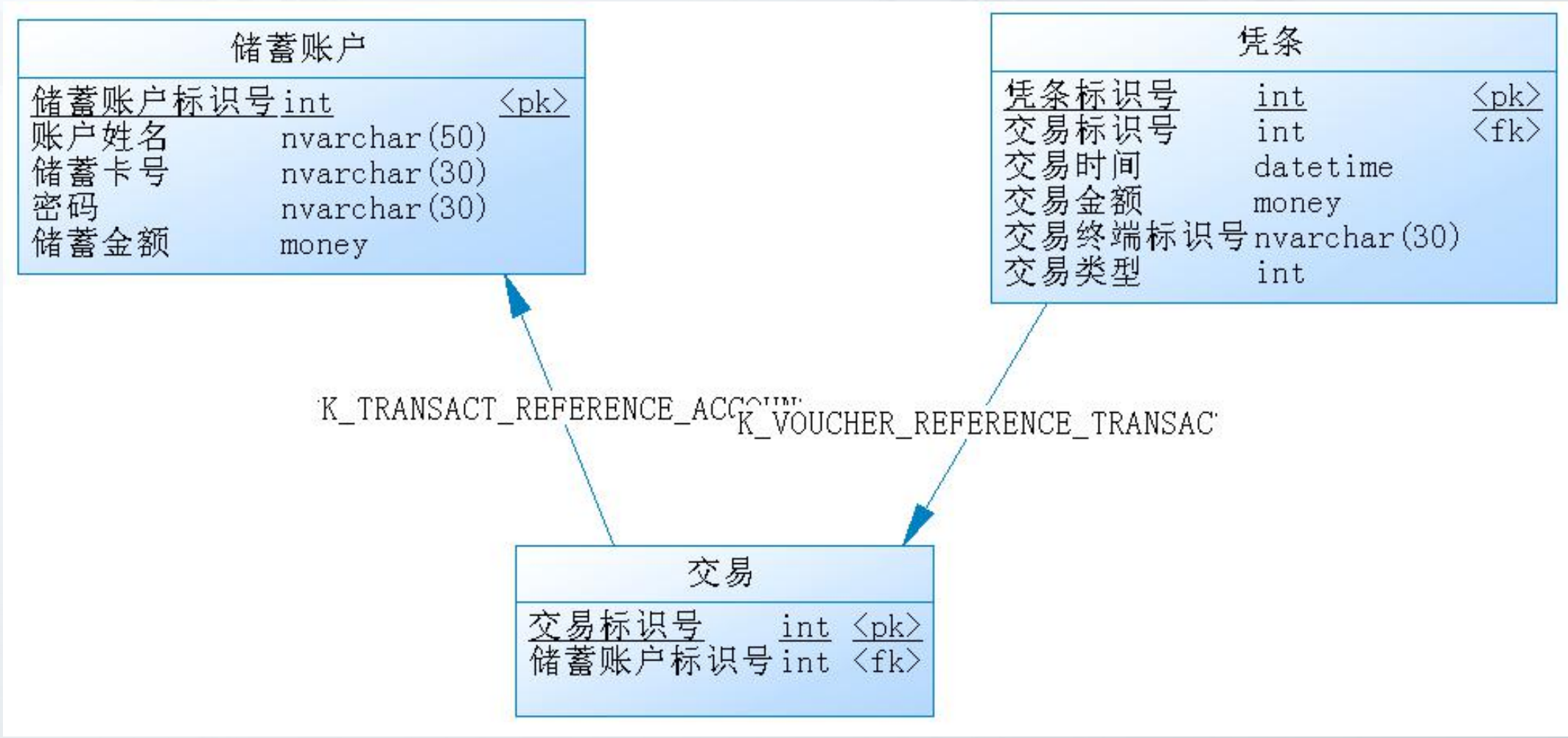
硬件支持环境

硬件名称	型号	数量
接口服务器	HP DL380	4
日志服务器	HP DL380	4
日志数据库	Drobo S 五盘位磁盘阵列	1
终端	普通PC Server	*
防火墙	网件(NETGEAR)ProSafe FVS318	1

硬件支持环境

操作系统	Windows Server 2008
数据库软件	Sql Sever 2005
支撑环境	.NET Framework 3.5
其它	

数据存储设计

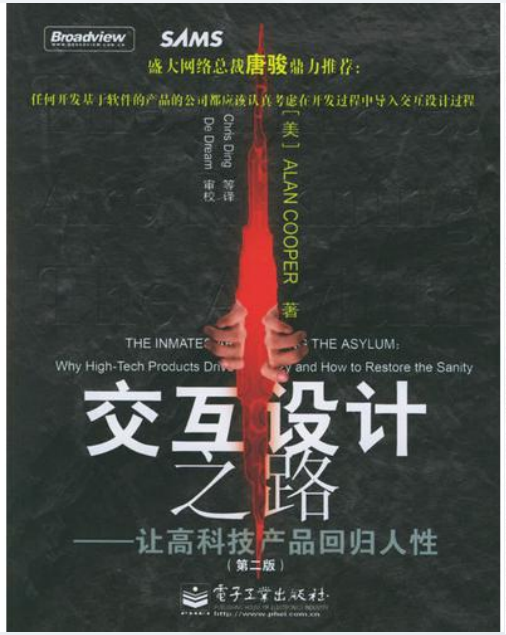

储蓄账户
储蓄账户标识号 int <pk>
账户姓名 nvarchar(50)
储蓄卡号 nvarchar(30)
密码 nvarchar(30)
储蓄金额 money
凭条
凭条标识号 int <pk>
交易标识号 int <fk>
交易时间 datetime
交易金额 money
交易终端标识号 nvarchar(30)
交易类型 int
K_TRANSACT_REFERENCE_ACC
K_VOUCHER_REFERENCE_TRANSAC
交易
交易标识号 int <pk>
储蓄账户标识号 int <fk>

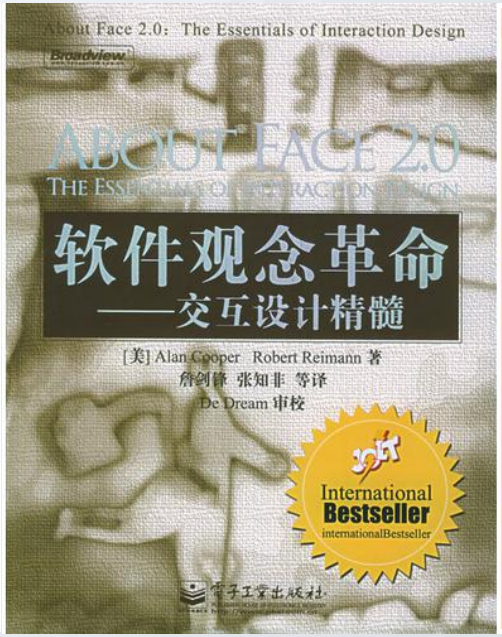
交互设计

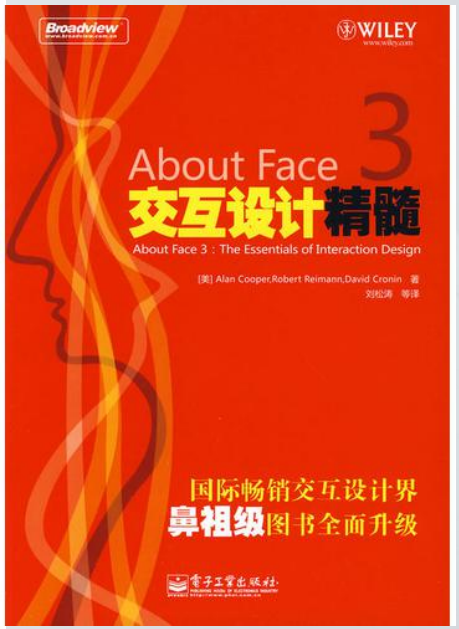
- 就是设计漂亮的界面吗？
- 对产品的界面和行为进行交互设计，让产品和它的使用者之间建立一种有机关系，从而可以有效达到使用者的目标，这就是交互设计的目的。
  http://baike.baidu.com/view/426920.htm
- 交互设计专家是非常稀缺和有前途的一个职业。
  - IBM：500人，25个可用性实验室
  - 微软：200人，30个可用性实验室
  - Philips：100多人
  - 西门子：100人
  - Oracle：约70人
  - 韩国三星：100多人
  - Adobe：21人
  - eBay：20人
  - Tencent：30

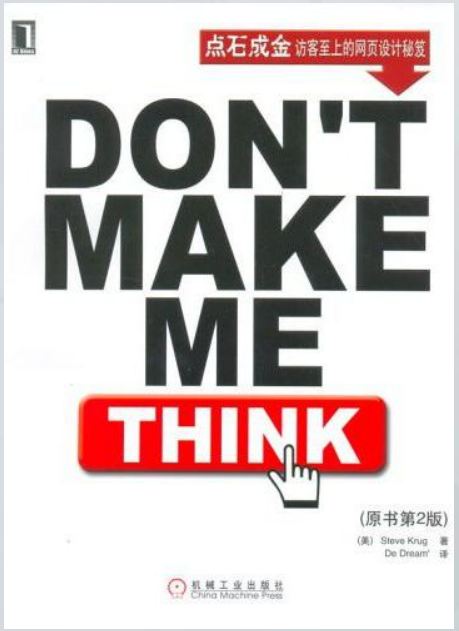
经典书籍

- http://wenku.baidu.com/view/ddd3a7254b35eefdc8d33307.html



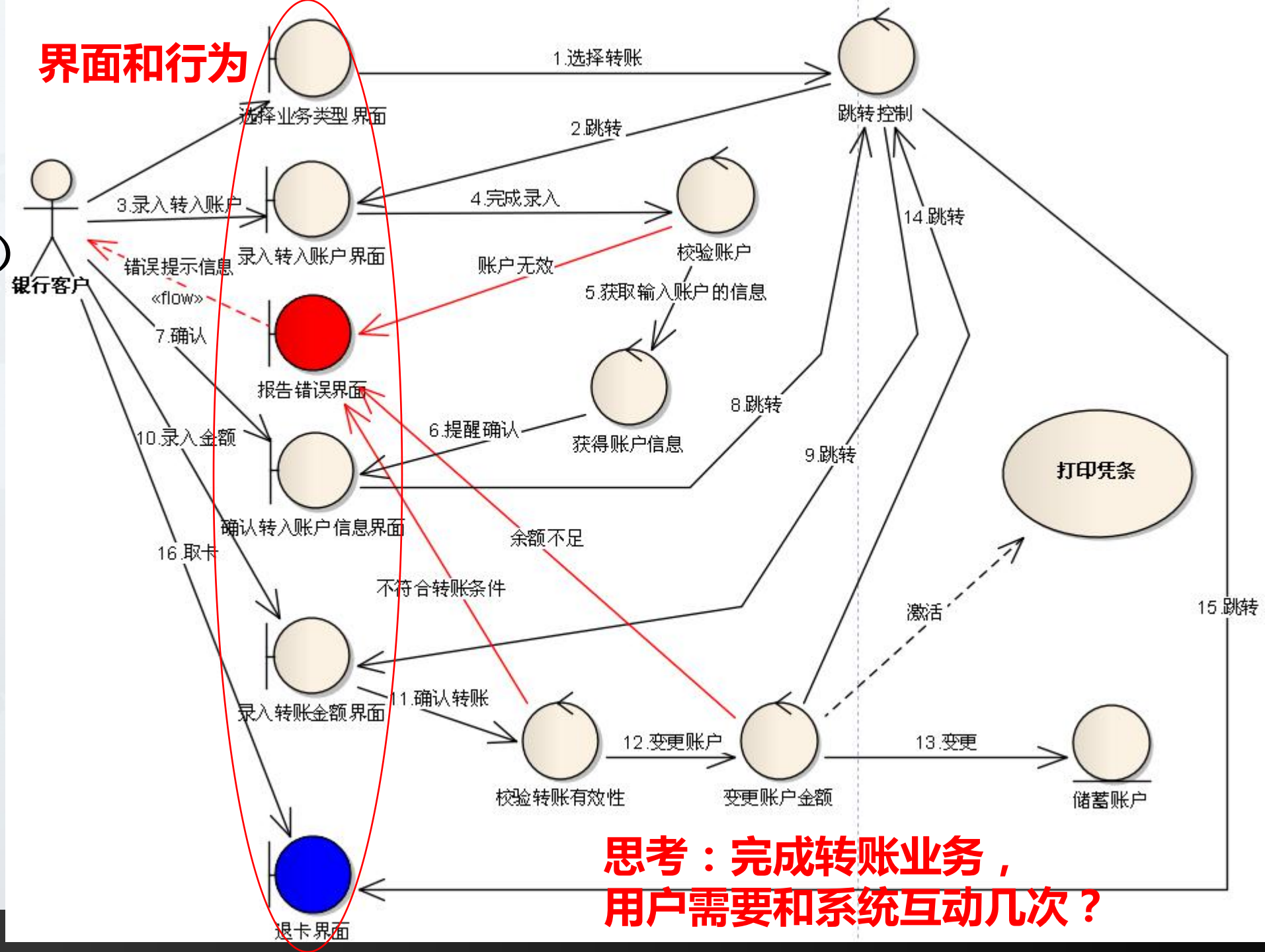






- 简约而不简单、Don't make me think原则

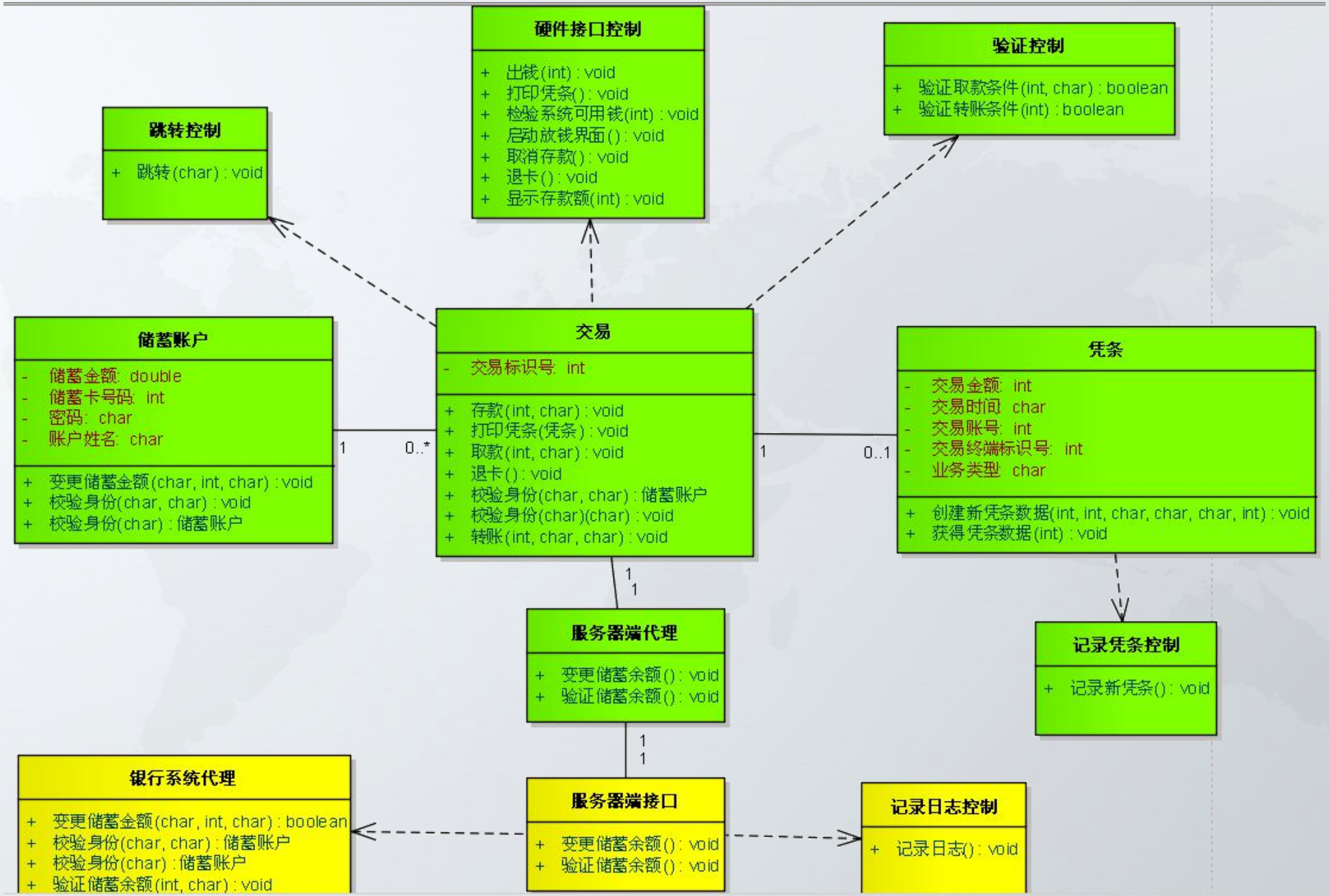
健壮性图（或用例）可以帮上忙



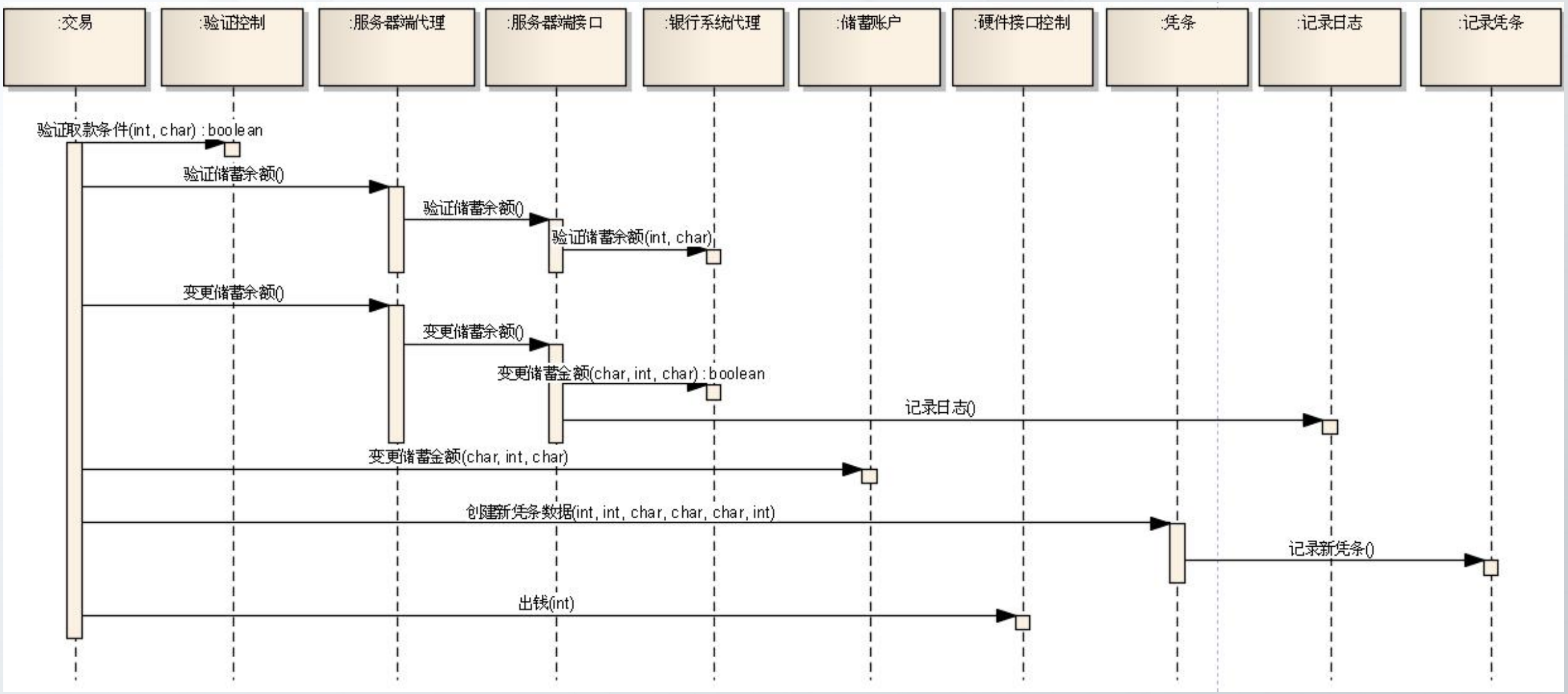
目录

详细设计的范例

- 结合体系结构、编程语言、数据模型和设计模式等来细化类图；
- 调整序列图（为了清晰易读，可以考虑去掉执行者和界面层的部分，因为这部分没有复杂的逻辑）

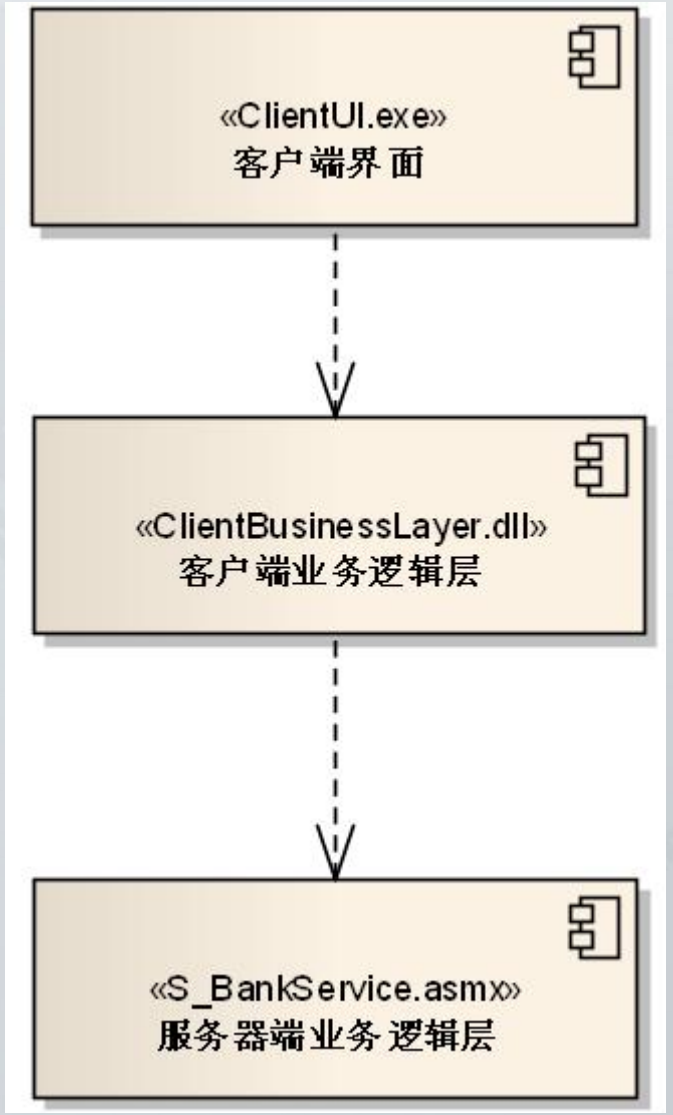

硬件接口控制
+ 出钱(int) : void
+ 打印凭条() : void
+ 检验系统可用钱(int) : void
+ 启动放钱界面() : void
+ 取消存款() : void
+ 退卡() : void
+ 显示存款额(int) : void
验证控制
+ 验证取款条件(int, char) : boolean
+ 验证转账条件(int) : boolean
跳转控制
+ 跳转(char) : void
储蓄账户
- 储蓄金额: double
- 储蓄卡号码: int
- 密码: char
- 账户姓名: char
+ 变更储蓄金额(char, int, char) : void
+ 校验身份(char, char) : void
+ 校验身份(char) : 储蓄账户
交易
- 交易标识号: int
+ 存款(int, char) : void
+ 打印凭条(凭条) : void
+ 取款(int, char) : void
+ 退卡() : void
+ 校验身份(char, char) : 储蓄账户
+ 校验身份(char)(char) : void
+ 转账(int, char, char) : void
凭条
- 交易金额: int
- 交易时间: char
- 交易账号: int
- 交易终端标识号: int
- 业务类型: char
+ 创建新凭条数据(int, int, char, char, char, int) : void
+ 获得凭条数据(int) : void
1
0..*
1
0..1
1
1
服务器端代理
+ 变更储蓄余额() : void
+ 验证储蓄余额() : void
记录凭条控制
+ 记录新凭条() : void
1
1
银行系统代理
+ 变更储蓄金额(char, int, char) : boolean
+ 校验身份(char, char) : 储蓄账户
+ 校验身份(char) : 储蓄账户
+ 验证储蓄余额(int, char) : void
服务器端接口
+ 变更储蓄余额() : void
+ 验证储蓄余额() : void
记录日志控制
+ 记录日志() : void

取款序列图


:交易
:验证控制
:服务器端代理
:服务器端接口
:银行系统代理
:储蓄账户
:硬件接口控制
:凭条
:记录日志
:记录凭条
验证取款条件(int, char) : boolean
验证储蓄余额()
验证储蓄余额()
验证储蓄余额(int, char)
变更储蓄余额()
变更储蓄余额()
变更储蓄金额(char, int, char) : boolean
记录日志()
变更储蓄金额(char, int, char)
创建新凭条数据(int, int, char, char, char, int)
记录新凭条()
出钱(int)

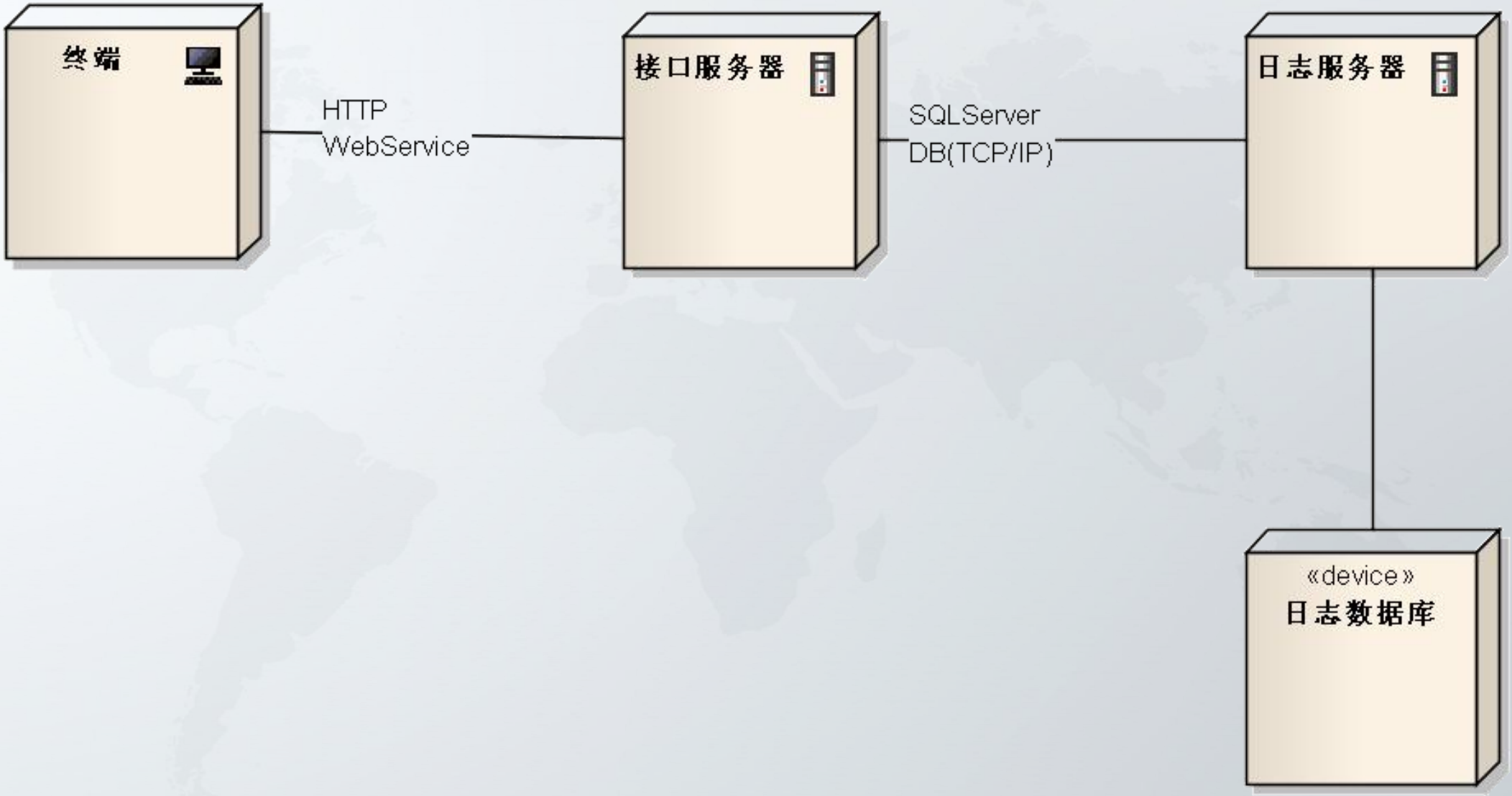
组件图

- 组件图(component diagram)是用来反映代码的物理结构。从组件图中，可以了解各软件组件（如源代码文件或动态链接库）之间的编译器和运行时依赖关系。使用组件图可以将系统划分为内聚组件并显示代码自身的结构。
- 描述如何把设计的类分配给不同实体组件。例如在.NET中，实体组件可能是库文件(DLL)，执行文件(EXE)，也可能是COM+；



部署图

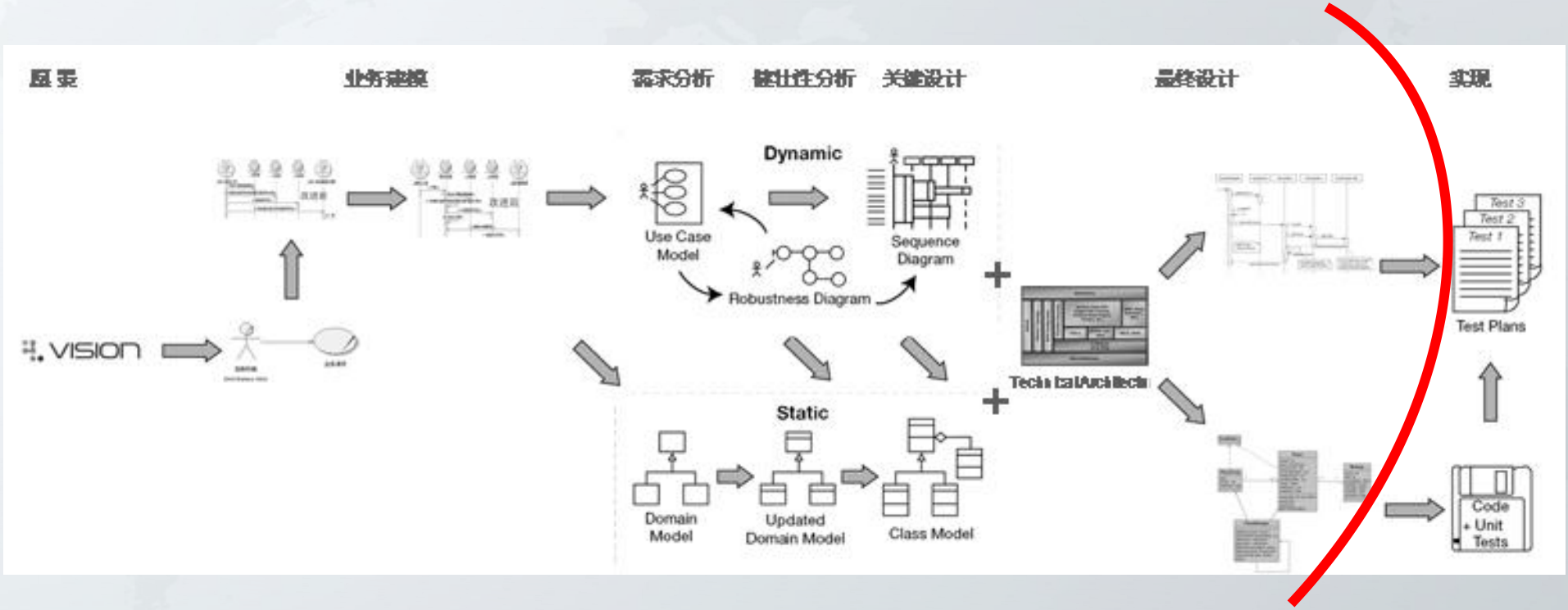
- 部署图(deployment diagram，配置图)是用来显示系统中软件和硬件的物理架构。从部署图中，您可以了解到软件和硬件组件之间的物理关系以及处理节点的组件分布情况。使用部署图可以显示运行时系统的结构，同时还传达构成应用程序的硬件和软件元素的配置和部署方式。
- 描述如何把实体组件部署在不同的机器上。
  - 节点代表某个保存设备、电脑或其他的实体资源；
  - 部署图上每个节点都对应于一到多个组件；


终端
HTTP
WebService
接口服务器
SQLServer
DB(TCP/IP)
日志服务器
«device»
日志数据库

目录

详细设计复核

- 可参见关键设计复核



详细设计之后……

- 编码
- 测试
- 部署
- 维护
- 升级

THANKS