



# 系统架构设计师：《考前冲刺》

架构第5期：邹月平



# 考试时间安排

题型	数量	考试时间	做题时间	分数
单选题	75题	9：00~11：30	150分钟	75分
案例题	1+4选2	13：30~15：00	90分钟	75分
论文	4选1	15：20-17：20	120分钟	75分

# 计算机基础

1、

名称	指令要求	寻址方式	实现方式	编译
复杂指令系统 CISC	指令数量众多，使用频率相差悬殊。可变长指令格式。	多种寻址方式。	与主存直接交互。微程序控制。	编译复杂。
精简指令系统 RISC	指令数量少，长度固定。	寻址方式少。	硬布线逻辑控制。流水线技术。与寄存器交互。	优化的编译器。

2、流水线周期为执行时间最长的一段；流水线执行时间= (t1+t2+...+tk) + (n-1) \* Δt

流水线最大吞吐率 $=\frac{1}{\Delta t}$ ，流水线加速比 $S = \frac{\text{不使用流水线执行时间}}{\text{使用流水线执行时间}}$

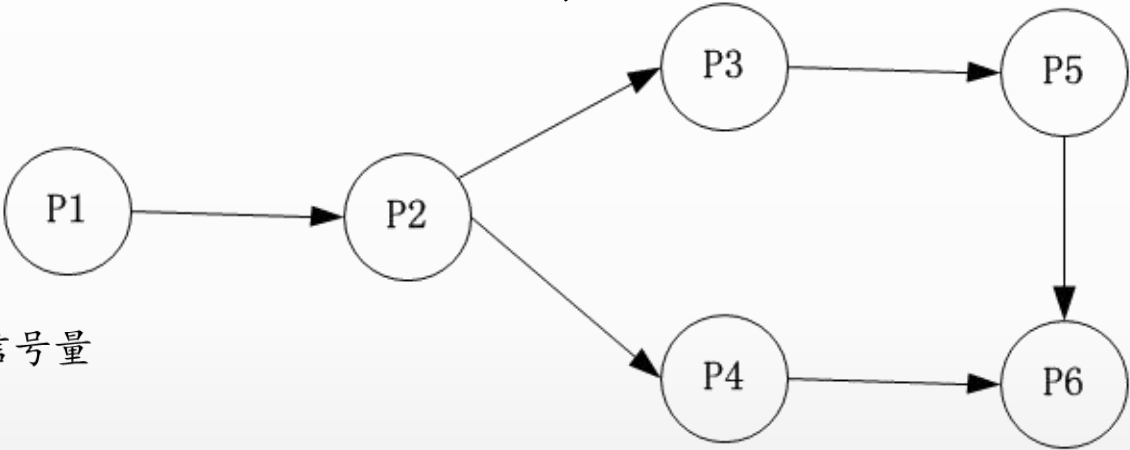
3、内存按字节编址，地址从A0000H到CFFFFH的内存，共有192K字节，若用存储容量为64k×8bit的存储器芯片构成该内存空间，至少需要3 片。若用存储容量为64k×4bit的存储器芯片构成该内存空间，至少需要6片。

# 操作系统

1、PV操作。例题1，假设系统中有 $n$ 个进程共享3台打印机，任一进程在任一时刻最多只能使用1台打印机。若用PV操作控制 $n$ 个进程使用打印机，则相应信号量 $s$ 的取值范围为 $3 \sim -(n-3)$ ；若信号量 $S$ 的值为 $-3$ ，则系统中有3个进程等待使用打印机。

# 操作系统

例题2 ， 进程P1、 P2、 P3、 P4、 P5和P6的前趋图如下所示。用PV操作控制这6个进程之间同步与互斥的程序如下， 程序中的空①和空②处应分别为 (D)， 空③和空④处应分别为 (B)， 空⑤和空⑥处应分别为 (A)。



```
begin
S1, S2, S3, S4, S5, S6: semaphore;    //定义信号量
S1:=0; S2:=0; S3:=0; S4:=0; S5:=0; S6:=0;
Cobegin
    process P1      process P2      process P3      process P4      process P5      process P6
    Begin           Begin           Begin           Begin           Begin           Begin
    P1 执行;        ①;              P (S2) ;        ④;              P (S4) ;        ⑥
    V(S1);          P2 执行;        P3 执行;        P4 执行;        P5 执行;        P (S6) ;
                   ②;              ③;              V (S5) ;        ⑤;              P6 执行;
    end;            end;            end;            end;            end;            end;
Coend;
end;
```



A.  $V(S1)$  和  $P(S2) \vee P(S3)$

B.  $V(S1)$  和  $V(S2) \vee V(S3)$

C.  $P(S1)$  和  $P(S2) \vee V(S3)$

D.  $P(S1)$  和  $V(S2) \vee V(S3)$

A.  $V(S3)$  和  $P(S3)$

B.  $V(S4)$  和  $P(S3)$

C.  $P(S3)$  和  $P(S4)$

D.  $V(S4)$  和  $P(S4)$

A.  $V(S6)$  和  $P(S5)$

B.  $V(S5)$  和  $P(S6)$

C.  $P(S5)$  和  $V(S6)$

D.  $P(S5)$  和  $V(S5)$

# 操作系统

3、存储管理。例题，进程P有8个页面，页号分别为0~7，页面大小为4K，假设系统给进程P分配了4个存储块，进程P的页面变换表如下所示。表中状态位等于1和0分别表示页面在内存和不在内存。若进程P要访问的逻辑地址为十六进制5148H，则该地址经过变换后，其物理地址应为十六进制（A） ；如果进程P要访问的页面6不在内存，那么应该淘汰页号为（B）的页面

页号	页帧号	状态位	访问位	修改位
0	—	0	0	0
1	7	1	1	0
2	5	1	0	1
3	—	0	0	0
4	—	0	0	0
5	3	1	1	1
6	—	0	0	0
7	9	1	1	0

- A. 3148H

B. 5148H
- C. 7148H

D. 9148H
- A. 1

B. 2
- C. 5

D. 9

# 操作系统

4、文件管理。例题，某文件系统文件存储采用文件索引节点法。假设文件索引节点中有8个地址项  $iaddr[0] \sim iaddr[7]$ ，每个地址项大小为4字节，其中地址项  $iaddr[0] \sim iaddr[5]$  为直接地址索引， $iaddr[6]$  是一级间接地址索引， $iaddr[7]$  是二级间接地址索引，磁盘索引块和磁盘数据块大小均为4KB。若要访问 `iclsClient.dll` 文件的逻辑块号分别为6、520和1030，则系统应分别采用一级间接地址索引、一级间接地址索引和二级间接地址索引。



# 数据库

- 1、概念模式是数据库中全体数据的逻辑结构和特征的描述，是所有用户的公共数据视图。一个数据库只有一个概念模式。外模式（子模式、用户模式）用以描述用户看到或使用的那部分数据的逻辑结构。外模式可以有多个。内模式定义的是存储记录的类型、存储域的表示以及存储记录的物理顺序，指引元、索引和存储路径等数据的存储组织。一个数据库只有一个内模式。
- 2、逻辑独立性是外模式和概念模式之间的映射。物理独立性是概念模式和内模式之间的映射。不管哪个独立性改变都不会影响应用程序。

3、

设计阶段	设计成果
需求分析	需求规格说明书、数据字典
概念设计	概念模型，E-R图。
逻辑设计	逻辑模型，关系模式。
物理设计	物理模型。

# 数据库

4、

实体完整性	实体的主属性不能取空值。
参照完整性	外键可以为空否则要有实际值。
用户定义完整性	数据必须满足一定的约束条件。如考试分数、人员工资等。

5、

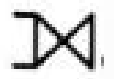
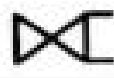

类型	特征
1NF	若关系模式R的每一个分量是不可再分的数据项，则关系模式R属于第一范式。
2NF	若关系模式R∈1NF，且每一个非主属性完全依赖主键时，则关系式R是第二范式。
3NF	即当2NF消除了非主属性对码（候选键）的传递函数依赖，则称为3NF。

不满足相应范式要求就有冗余、增、删、改四种异常问题。

6、关系代数运算

笛卡尔积	×	行：R×S。列：R+S
等值连接	⋈	关系R、S, 取两者笛卡尔积中属性值相等的元组。
自然连接		等值连接的结果去重。

# 数据库

运算符	含义	名词解释
	左外连接	取出左侧关系中所有与右侧关系中任一元组都不匹配的元组，用空值 null 填充所有来自右侧关系的属性，构成新的元组，将其加入自然连接的结果中。
	右外连接	取出右侧关系中所有与左侧关系中任一元组都不匹配的元组，用空值 null 填充所有来自左侧关系的属性，构成新的元组，将其加入自然连接的结果中。
	完全外连接	完成左外连接和右外连接的操作。即填充左侧关系中所有与右侧关系中任一元组都不匹配的元组，并填充右侧关系中所有与左侧关系中任一元组都不匹配的元组，将产生的新元组加入自然连接的结果中。

# ▶ 数据库

## 7、反规范化技术：

优点：降低表之间的连接操作、降低外码和索引的数目，还可能减少表的数目，能够提高查询效率。缺点：数据的重复存储，浪费了磁盘空间；可能出现数据的完整性问题，为了保障数据的一致性，增加了数据维护的复杂性，会降低修改速度。

增加冗余列	增加冗余列是指在多个表中具有相同的列，它常用来在查询时避免连接操作。
增加派生列	增加派生列指增加的列可以通过表中其他数据计算生成。它的作用是在查询时减少计算量，从而加快查询速度。
重新组表	重新组表指如果许多用户需要查看两个表连接出来的结果数据，则把这两个表重新组成一个表来减少连接而提高性能。
水平分割表	按记录进行分割，把数据放到多个独立的表中，主要用于表数据规模很大、表中数据相对独立或数据需要存放到多个介质上时使用。
垂直分割表	对表进行分割，将主键与部分列放到一个表中，主键与其它列放到另一个表中，在查询时减少 I/O 次数。

# 数据库

## 8、数据库安全性技术

措 施	说明
用户标识和鉴别	最外层的安全保护措施，可以使用用户账户、口令和随机数检验等方式
存取控制（数据授权）	对用户进行授权，包括操作类型（例如，查找、更新或删除等）和数据对象（主要是数据范围）的权限
密码存储和传输	对远程终端信息用密码传输
视图的保护	通过视图的方式进行授权
审计	用一个专用文件或数据库，自动将用户对数据库的所有操作记录下来

# ▶ 数据库

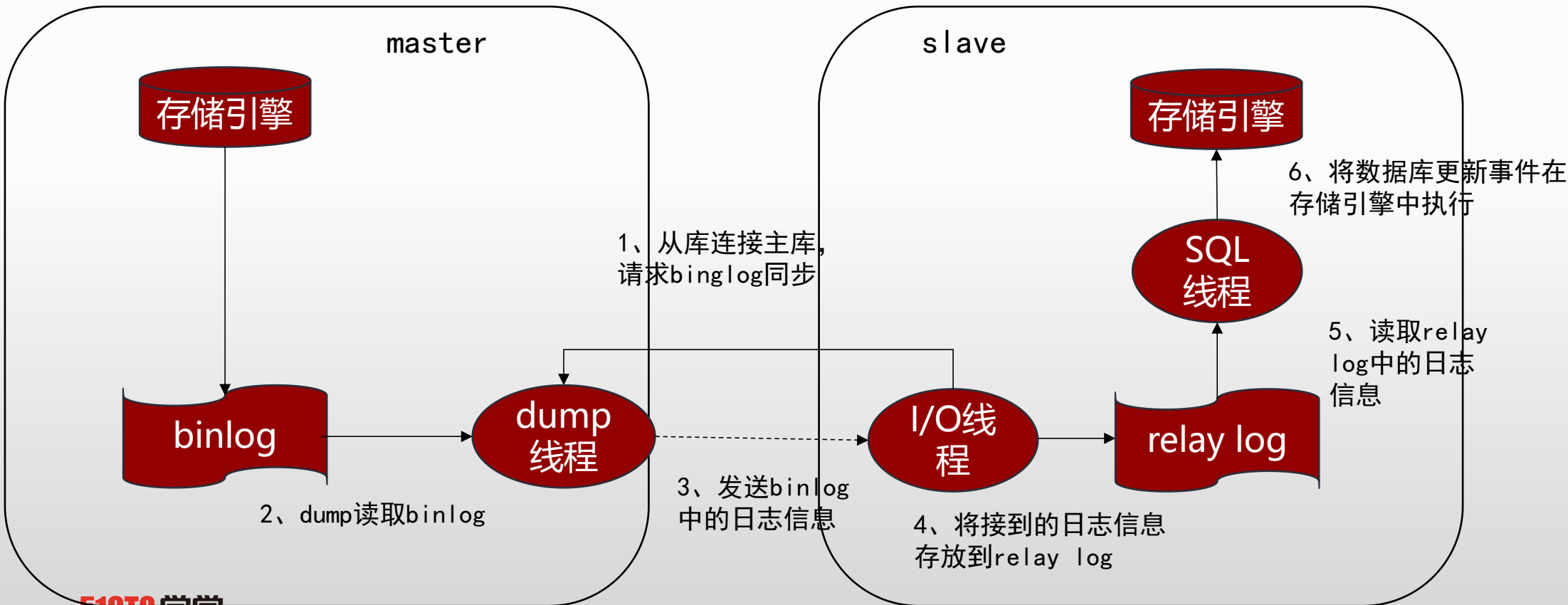
## 9、数据库分区技术

	范围分区	哈希分区	列表分区
数据值	<u>连续</u>	连续离散均可	<u>离散</u>
数据分布	<u>不均匀</u>	<u>均匀</u>	不均匀

- 10、主从复制，读写分离是设置物理上不同的主/从服务器，让主服务器负责数据的写操作，从服务器负责数据的读操作，从而有效减少数据并发操作的延迟。引入主从复制机制所带来的好处：
- ① 避免数据库单点故障、提高可用性：主服务器实时、异步复制数据到从服务器，当主数据库宕机时，可在从数据库中选择一个升级为主服务器，从而防止数据库单点故障。
  - ② 提高查询效率：根据系统数据库访问特点，可以使用主数据库进行数据的插入、删除及更新等写操作，而从数据库则专门用来进行数据查询操作，从而将查询操作分担到不同的从服务器以提高数据库访问效率。

# ▶ 数据库

11、主从数据库之间通过 binary log（二进制日志）进行数据的同步。具体过程如下：



# 数据库

12、binlog有三种模式：

①**基于SQL语句的复制**，每一条更新的语句（insert、update、delete）都会记录在binlog中，进而同步到从库的relaylog中，被从库的SQL线程取出来，回放执行。

该模式的优点是binlog的日志量可能会比较少，比如一个涉及行数为1000行的update语句：同步这一个语句，就同步了1000行的数据。缺点是：同步的SQL语句里如果含有绑定本地变量的函数、关键字时，可能造成主从不一致的情况。比如SQL语句中有time函数，如果主从数据库的服务器时间不是精确相等，就会造成结果不一致。

②**基于行的复制**，不记录SQL语句，只记录了哪个记录更新前和更新后的数据，可以保证主从之间数据绝对相同。缺点是：1条SQL更新1000行的数据无法再偷懒，必须原原本本同步1000行的数据量。

③**混合复制**：以上两种模式的混合，选取两者的优点。对于有绑定本地特性、评估可能造成主从不一致的SQL语句，则自动选用ROW，其他的选择STATEMENT。



# 数据库

## 13、mysql主从同步的同步模式：

	一致性	可用性	特征	典型框架/系统
同步复制技术	强	弱	主数据库需要等待所有备数据库均操作成功才可以响应用户，影响用户体验。	MySQL
异步复制技术	弱	强	主数据库处理完请求后可直接给用户响应，而不必等待备数据库完成同步，备数据库会异步进行数据的同步。	MySQL、Redis、Oracle
半同步复制技术	较强	较弱	用户发出写请求后，主数据库会执行写操作，并给备数据库发送同步请求，主数据库可以等待一部分备数据库同步完成后响应用户写操作执行成功。	MySQL、Zookeeper、CloudSQLServer、Redis、Oracle、Etcd等

# 数据库

14、redis用作缓存组件时，其基于内存的读写特性，比基于磁盘读写的数据库性能要高很多，适合缓存高频热点的数据，来提高读性能。这样可以降低对数据库服务器的查询请求，提高系统性能。

## (1) Redis支持的数据类型

string	基本类型。可用于缓存层或计数器，如视频播放量、文章浏览量等
hash	代替string类型，节省空间。描述用户信息较为方便
set	无序集合，每个值不能重复。可用于去重、抽奖、初始化用户池等
list	双向链表结构，可以模拟栈、队列等形式。可用于回复评论、点赞
zset	有序集合、每个元素有一个分数。如首页推荐10个最热门的帖子

# 数据库

(2) 引入Redis后，读写数据的基本步骤为：

## ➤ 读数据

- ① 根据key读缓存；
- ② 读取成功则直接返回；
- ③ 若key不在缓存中时，根据key读数据库 ；
- ④ 读取成功后，写缓存 ；
- ⑤ 成功返回。

## ➤ 写数据

- ① 根据key值写数据库；
- ② 成功后更新缓存key值 ；
- ③ 成功返回。

# 数据库

## (3) 过期策略

### ➤ 惰性删除

查询key的时候才对key进行检测，如果已经达到过期时间，则删除。缺点是如果这些过期的key没有被访问，那么他就一直无法被删除，而且一直占用内存。

### ➤ 定期删除

redis每隔一段时间对数据库做一次检查，删除里面的过期key。由于不可能对所有key去做轮询来删除，所以redis会每次随机取一些key去做检查和删除。

# 数据库

## ➤ 内存淘汰机制

volatile-lru 最近最少使用	从已设置过期时间的key中，移出最近最少使用的key进行淘汰
volatile-lfu 最不经常使用	从已设置过期时间的key中选择最不经常使用的进行淘汰。
volatile-random 随机淘汰算法	从已设置过期时间的key中随机选择key淘汰。
volatile-ttl 生存时间淘汰	从已设置过期时间的key中，移出将要过期的key。
allkeys-lru	从所有key中选择最近最少使用的进行淘汰
allkeys-lfu	从所有key中选择最不经常使用的进行淘汰。
allkeys-random	从所有key中随机选择key进行淘汰

# 数据库

(4) 一旦服务器宕机，内存中的数据将全部丢失。我们很容易想到的一个解决方案是，从后端数据库恢复这些数据，但这种方式存在两个问题：一是，需要频繁访问数据库，会给数据库带来巨大的压力；二是，这些数据是从慢速数据库中读取出来的，性能肯定比不上从 Redis 中读取，导致使用这些数据的应用程序响应变慢。所以，对 Redis 来说，实现数据的持久化，避免从后端数据库中进行恢复，是至关重要的。

	RDB内存快照 (RedisDataBase)	AOF日志 (Append Only Flie)
说明	把当前内存中的数据快照写入磁盘（数据库中所有键值对数据）。恢复时是将快照文件直接读到内存里。	通过持续不断地保存Redis服务器所执行的更新命令来记录数据库状态，类似mysql的binlog。恢复数据时，需从头开始回放更新命令。
磁盘刷新频率	低	高
文件大小	小	大
数据恢复效率	高	低
数据安全	低	高

# 数据库

## (5) 缓存异常问题

问题	原因	应对方案
缓存雪崩	<ul style="list-style-type: none"><li>大量数据同时过期</li><li>缓存实例宕机</li></ul>	<ul style="list-style-type: none"><li>给缓存数据的过期时间上加上小的随机数，避免同时过期</li><li>服务降级</li><li>服务熔断</li><li>请求限流</li><li>Redis缓存主从集群</li></ul>
缓存击穿	访问非常频繁的热点数据过期	不给热点数据设置过期时间，一直保留
缓存穿透	缓存和数据库中都没有要访问的数据	<ul style="list-style-type: none"><li>缓存空值或缺省值</li><li>请使用布隆过滤器快速判断</li><li>请求入口前端对请求合法性进行检查</li></ul>

➤ 从考试的角度来讲，不管是缓存还是数据库，以下答案是万金油：

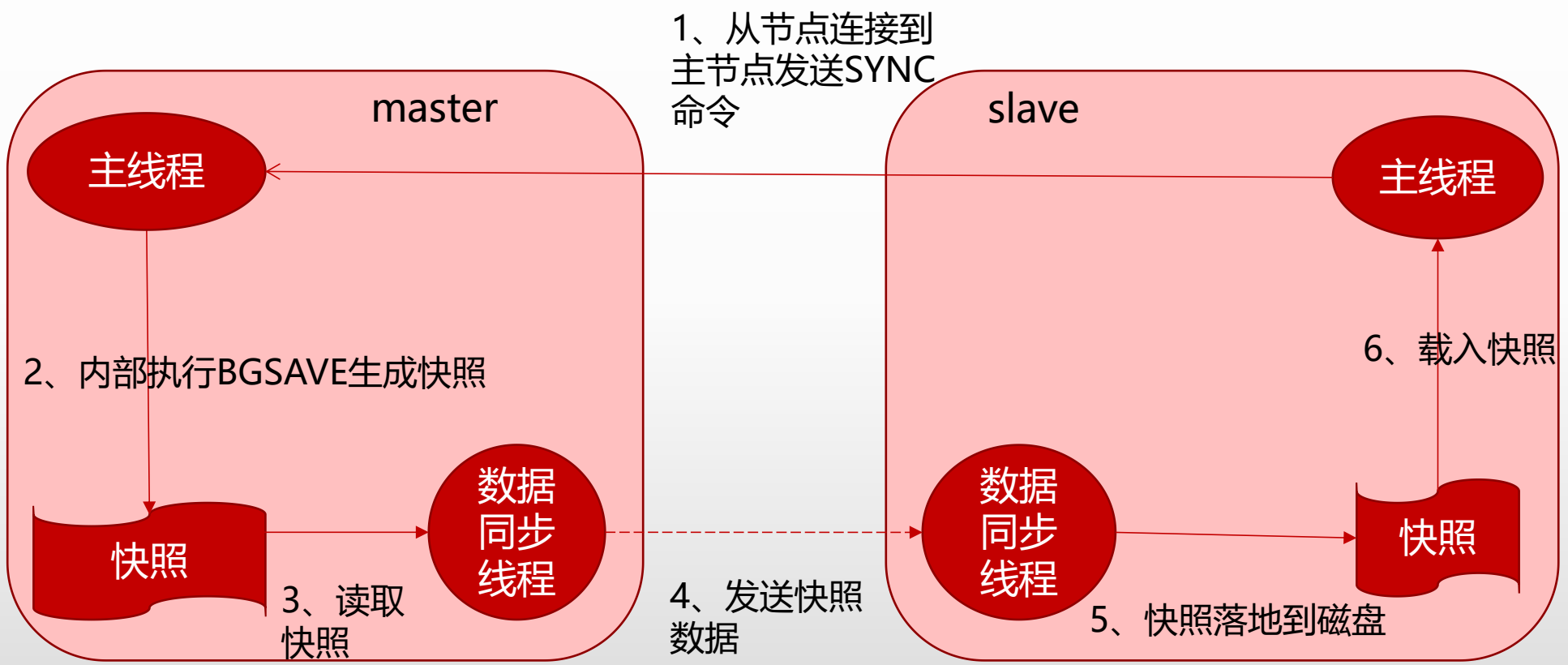
■提升性能（吞吐率、并发、延迟）、可用性、伸缩性（扩展性）的方法：采

51CTO学堂 用分布式集群（主从复制、读写分离、分库分表）

■防止异常的方法：服务降级、服务熔断、请求限流。

# 数据库

(6) Redis集群方式：主从复制、哨兵集群、cluster集群



Redis主从复制过程



# ▶ 数据库

## (7) Redis数据切片方式

客户端分片	将分片工作放在业务程序端。不依赖于第三方分布式中间件，实现方法和代码可掌控，对开发人员要求高。
代理分片	将分片工作交给专门的代理程序来做，运维方便。代表：Twemproxy，Codis
服务器端分片	Redis Cluster将所有Key映射到16384个Slot中，集群中每个Redis实例负责一部分，业务程序通过集成的Redis Cluster客户端进行操作。

# 数据库

15、分布式数据库的分布透明性包括：分片透明性、位置透明性和局部数据模型透明性。

(1) 分片透明性是分布透明性的最高层次。所谓分片透明性是指用户或应用程序只对全局关系进行操作而不必考虑数据的分片。

(2) 位置透明性是分布透明性的下一层次。所谓位置透明性是指，用户或应用程序应当了解分片情况，但不必了解片段的存储场地。

(3) 局部数据模型（逻辑透明）透明性是指用户或应用程序应当了解分片及各片断存储的场地，但不必了解局部场地上使用的是何种数据模型。

# ▶ 计算机网络

## 1、应用层协议

动态主机配置协议 DHCP	是集中的管理、分配IP地址，使网络环境中的主机动态的获得IP地址、网关地址、DNS服务器地址等信息。网络范围内可能存在多个DHCP服务器，也可能只有一个DHCP服务器。如果网络中有多个DHCP服务器发送OFFER报文，客户端只根据第一个收到的OFFER报文，返回REQUEST报文。
域名系统 DNS	把主机域名解析为IP地址的系统。而PTR（Pointer Recored）负责将IP地址映射到域名的解析。 DNS查询过程有两种方法：迭代查询、递归查询。

## 2、传输层协议

TCP	UDP
可靠的、面向连接的、字节流服务。首部开销20个字节。	不可靠的、无连接、面向报文。首部开销8个字节。
具有差错校验和重传、流量控制、拥塞控制等功能。 适用于数据量比较少，且对可靠性要求高的场合。	数据量大，对可靠性要求不是很高，但要求速度快的场合。

# ▶ 计算机网络

## 3、IPv6数据报的目的地址：

- (1) 单播（unicast）：点对点通信。
- (2) 多播/组播（multicast）：一点对多点的通信。
- (3) 任播（anycast）：新增类型。一对最近。

IPv4 to IPv6 过渡技术有：

双协议栈技术	两种技术共存。
隧道技术	在IPv4 网络中部署隧道。
NAT-PT 技术	NAT - PT 网关实现两种协议的转换翻译和地址的映射。

## 4、分层网络设计

接入层	用户管理功能（地址认证、用户认证、计费管理），用户信息收集（用户的IP地址、MAC地址、访问日志）
汇聚层	网络访问策略控制、数据包处理、过滤、寻址
核心层	高速转发通信、出口路由、常使用冗余机制。

# 信息安全

1、

类型	特征	算法
对称加密	又叫私钥加密算法，加密与解密密钥相同。适合大数据量（明文）加密。加密速度快。	DES、3DES、AES、IDEA、RC4
非对称加密	又叫公钥加密算法，加密与解密密钥不同。适合少量数据量加密。加密速度慢。身份认证抗抵赖性好。	RSA、ECC椭圆曲线

2、

类型	特征
消息摘要	使用单向哈希函数生成摘要发送给接收方。常用算法MD5（密文长度128位）、SHA-256 安全散列算法。
数字签名	利用公钥加密对消息摘要签名。其中发送方私钥用于签名而发送方公钥用于验证。
数字证书	用电子手段来证实一个用户的身份和对网络资源的访问权限。由数字证书认证机构CA和签发和管理。CA可以证明数字证书的持有者合法拥有证书中列出的公开密钥。

# 信息系统基础

1、信息化需求包括三个层次：

01

## 战略需求

组织信息化的目标是提升组织的竞争能力、为组织的可持续发展提供一个支持环境。  
信息化战略是企业竞争的基础。

02

## 运作需求

组织信息化的运作需求是组织信息化需求非常重要且关键的一环，它包含三方面的内容：一是实现信息化战略目标的需要；二是运作策略的需要。三是人才培养的需要。

03

## 技术需求

由于系统开发时间过长等问题在信息技术层面对系统的完善、升级、集成和整合提出了需求。

# ► 信息系统基础

2、企业信息化涉及对企业管理理念的创新，管理流程的优化，管理团队的重组和管理手段的革新。管理创新是按照市场发展的要求，对企业现有的管理流程重新整合，从作为管理核心的**财务、物料管理**，转向**技术、物资、人力资源**的管理，并延伸到企业技术创新、工艺设计、产品设计、生产制造过程的管理，进而还要扩展到**客户关系管理、供应链管理**乃至发展到电子商务。

# 信息系统基础

## 3、企业信息化规划

### 企业战略规划

是用机会和威胁评价现在和未来的环境，  
用优势和劣势评价企业现状，  
进而选择和确定企业的总体和长远目标，  
制定和抉择实现目标的行动方案。

### 信息系统战略规划

关注的是如何通过信息系统来支撑业务流程的运作，  
进而实现企业的关键业务目标，  
其重点在于对信息系统远景、  
组成架构、各部分逻辑关系进行规划。

### 信息技术战略规划

通常简称为IT战略规划，  
是在信息系统规划的基础上，  
对支撑信息系统运行的  
硬件、软件、支撑环境  
等进行具体的规划，它更关心技术层面的问题。



# 信息系统基础

## 4、信息系统战略规划

01

第一个阶段主要以数据处理为核心，围绕职能部门需求的信息系统规划，主要的方法包括企业系统规划法、关键成功因素法和战略集合转化法；

02

第二个阶段主要以企业内部管理信息系统为核心，围绕企业整体需求进行的信息系统规划，主要的方法包括战略数据规划法、信息工程法和战略栅格法；

03

第三个阶段的方法在综合考虑企业内外环境的情况下，以集成为核心，围绕企业战略需求进行的信息系统规划，主要的方法包括价值链分析法和战略一致性模型。

低



高

# 信息系统基础

名称	特点
企业系统规划法	BSP方法主要用于大型信息系统的开发。BSP方法的目标是提供一个信息系统规划，用以支持企业短期的和长期的信息需求。
关键成功因素法	通过对CSF的识别，找出实现目标所需要的关键信息集合，从而确定系统开发的优先次序。
战略集合转化法	SST将企业战略看成是一个“信息集合”，包括使命、目标、战略和其他企业属性，例如，管理水平、发展趋势以及重要的环境约束等。
战略数据规划法	SDP是挖掘数据、信息的规律。
信息工程方法	以企业内部管理信息系统为核心，围绕企业整体需求进行信息系统规划。
战略栅格法	该方法创建一个2×2的矩阵（栅格表），从战略影响方面标出企业现有的和将来的信息系统组合的特征，也就是它们对企业生存前景的影响。
价值链分析法	企业为一系列的输入、转换与输出的活动序列集合，每个活动都有可能相对于最终产品产生增值行为，从而增强企业的竞争地位。
战略一致性模型	SAM把企业战略规划和信息化战略规划的关系划分为内、外两大部分。外部是指企业所面临的外部竞争环境，例如，产品或IT市场等；内部区域包括企业组织结构、整体信息架构和业务流程等。

# 信息系统基础

## 5、电子政务：

在社会中，与电子政务相关的主体主要有三个，即政府、企（事）业及居民

名称	解释
政府对政府 G2G	政府内部的政务活动，包括国家和地方基础信息的采集、处理和利用，如人口信息；政府之间各种业务流所需要采集和处理的信息，如计划管理；政府之间的通信系统，如网络系统；政府内部的各种管理信息系统，如财务管理；以及各级政府的决策支持系统和执行信息系统，等等。
政府对企业 G2B	政府面向企业的活动主要包括政府向企（事）业单位发布的各种方针、政策、法规、行政规定，即企（事）业单位从事合法业务活动的环境,政府向企（事）业单位颁发的各种营业执照、许可证、合格证和质量认证等。

# 信息系统基础

名称	解释
政府对居民 G2C	政府面向居民所提供的服务，以及各种关于社区公安和水、火、天灾等与公共安全有关的信息。户口、各种证件和牌照的管理等，还包括各公共部门，如学校、医院、图书馆和公园等。
企业对政府 B2G	企业面向政府的活动包括企业应向政府缴纳的各种税款，按政府要求应该填报的各种统计信息和报表，参加政府各项工程的竞、投标，向政府供应各种商品和服务，以及申请的援助。
居民对政府 C2G	包括个人应向政府缴纳的各种税款和费用，按政府要求应该填报的各种信息和表格，以及缴纳各种罚款等。此外，报警服务（盗贼、医疗、急救、火警等）即在紧急情况下居民需要向政府报告并要求政府提供的服务，也属于这个范围。

# 信息系統基礎

7、BI系統主要包括數據預處理、建立數據倉庫、數據分析和數據展現4個主要階段。

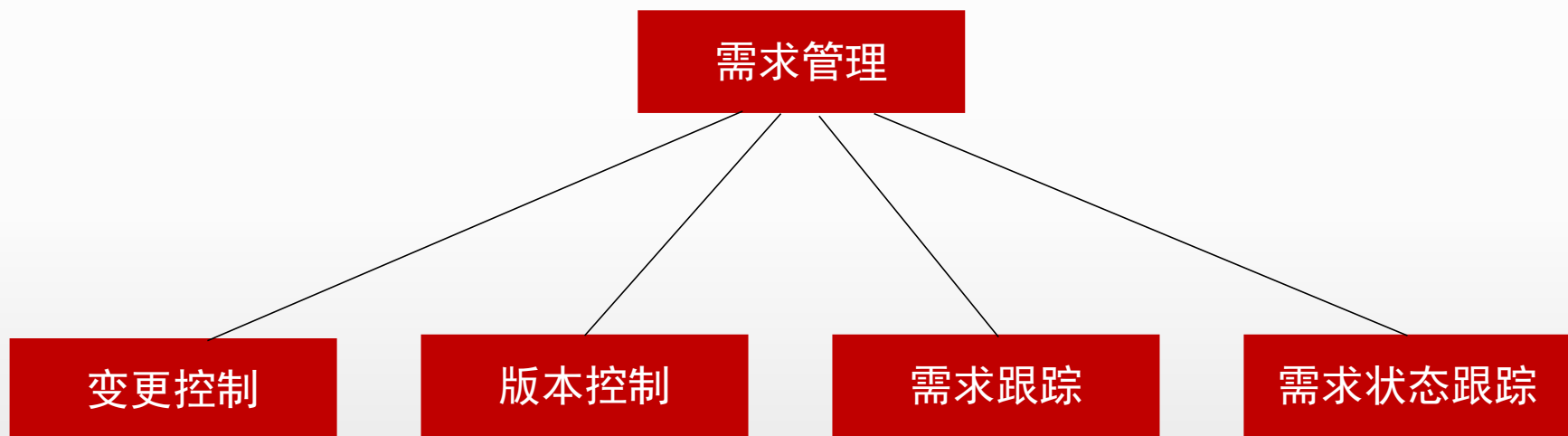
數據預處理	第一步，它包括數據的抽取（extraction）、轉換（transformation）和加載（load）三個過程（ETL過程）
建立數據倉庫	處理海量數據的基礎
數據分析	體現系統智能的關鍵，一般採用OLAP和數據挖掘兩大技術。OLAP不僅進行數據汇总/聚集，同時還提供切片、切塊、下鑽、上卷和旋轉等數據分析功能。數據挖掘的目標則是挖掘數據背後隱藏的知識。
數據展現	保障系統分析結果的可視化。

➤ 一般認為，數據倉庫、OLAP和數據挖掘技術是BI的三大組成部分。

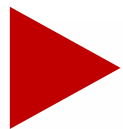


# 信息系统基础

8、需求管理是一个对系统需求变更、了解和控制的过程。需求管理的主要活动如下：

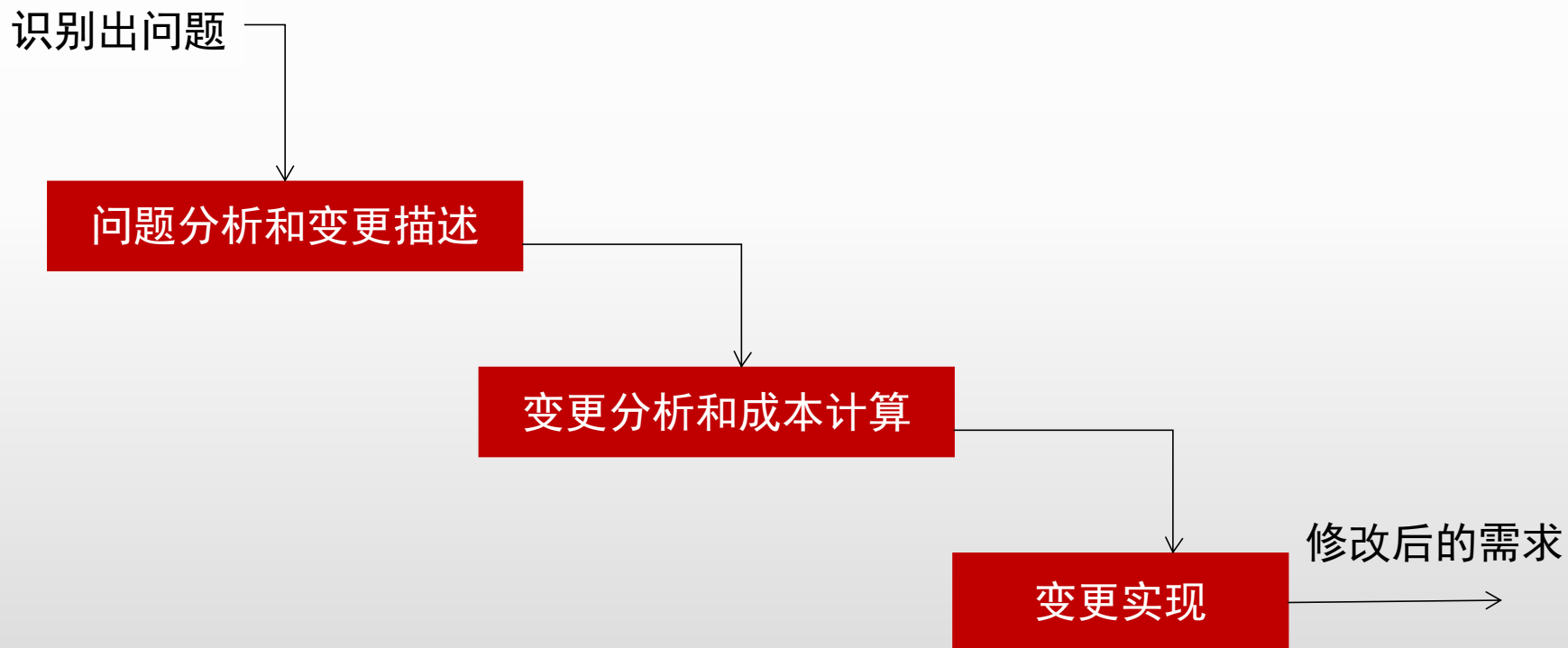


需求管理的主要活动



# 信息系统基础

## 9、需求变更管理过程：



# ▶ 知识产权

知识产权客体	知识产权类型	保护期限
公民作品	署名权、修改权、保护作品完整权	没有限制
	发表权、使用权、获得报酬权	作者终生及其死亡后的第50年的12月31日
单位作品	发表权、使用权、获得报酬权	首次发表后的第50年的12月31日，若未发表则不受保护。



# 知识产权

客体类型		说明	归属
作品	职务作品	利用单位的物质技术条件创作，并由单位承担责任的。	除署名权外其他著作权归单位
		合同约定	除署名权外其他著作权归单位
		其他	作者拥有著作权，单位有权在业务范围内优先使用
软件	职务作品	属于本职工作中明确规定的开发目标	单位享有著作权
		属于从事本职工作活动的结果	单位享有著作权
		使用了单位资金、专用设备、未公开的信息等物质、技术条件，并由单位或组织承担责任的软件	单位享有著作权
专利权	职务作品	本职工作中做出的发明创造	单位享有专利
		履行本单位交付的本职工作之外的任务所作出的发明创造	单位享有专利
		离职、退休或调动工作后1年内，与原单位工作相关	单位享有专利

# 知识产权

客体类型		说明	归属
软件作品	委托创作	合同约定，著作权归委托方	委托方
		合同中未约定著作权归属	创作方
	合作开发	只进行组织、提供咨询意见、物质条件或者进行其他辅助工作	不享有著作权
		共同创作的	共同享有著作权、按人头比例成果可分割的，可分开申请
商标		谁先申请谁拥有（除知名商标的非法抢注） 同时申请，谁先使用谁拥有（需提供证据） 无法提供证据，可协商，协商不成的抽签决定。	
专利		谁先申请谁拥有 同时申请，协商归属	

# 软件工程师

1、按照传统的软件生命周期方法学，可以把软件生命周期划分为**软件定义**、**软件开发**、**软件运行与维护**三个阶段。软件定义包括可行性研究和详细需求分析过程，任务是确定软件开发工程必须完成的目标。具体可分成**问题定义**、**可行性研究**、**需求分析**等。软件开发时期就是软件的设计与实现，可分成**概要（总体）设计**、**详细设计**、**编码**、**测试**等。

2、软件开发方法分类：

按照开发风范	自顶向下	大问题分化成小问题，逐一解决。
	自底向上	从具体的部件开始，凭借设计者的技巧进行相互连接、修改和扩大。
按照性质	形式化开发	数学推理方法。代表有净室工程。
	非形式化开发	各种开发模型。
适用范围	整体性	软件开发全过程的方法。
	局部性	某个具体阶段的软件方法。

# 软件工**程**

结构化开发	优点：开发目标清晰化、开发工作阶段化、开发文档规范化、设计方法结构化	
	缺点：开发周期长、难以适应需求变化、很少考虑数据结构。	
面向对象	Coad/Yourdon方法	OOA和OOD采用完全一致的概念和表示法，使分析和设计之间不需要表示法的转换。
	Booch方法	静态模型描述系统的构成和结构。分为逻辑模型（类图、对象图）和物理模型（模块图、进程图），动态模型描述对象的状态变化和交互过程。包括状态图和顺序图。
	OMT方法	使用了建模的思想，使用对象模型（对象图）、动态模型（状态图）和功能模型（DFD）。
	OOSE	用例取代DFD进行需求分析和功能建模。

# 软件工程

基于构件的软件开发	构件的获取	1、使用现有构件。2、提取遗留工程构件。3、购买。4、开发。	
	构件复用方法	检索与提取构件、理解与评价构件、修改构件、构件组装。	
	构件分类	关键字分类法：将应用概念分为 <b>树形或有向无回路图结构</b> ，用关键字表示。	
		刻面分类法：用“ <b>刻面</b> ”描述构件执行的功能、被操作的数据、构件应用的语境或其他特征。	
面向服务的开发	抽象级别	超文本方法：可按照 <b>人类的联想思维</b> 方式 <b>任意跳转</b> 到包含相关概念或构件的文档。	
		操作。 <b>最低层</b> ，代表单个逻辑单元的事物，包含特定的结构化接口，并且返回结构化的响应。	
		服务。代表操作的逻辑分组。	
敏捷开发		业务流程。 <b>最高层</b> ，为了实现特定业务目标而执行的一组长期运行的动作或者活动。	
		以人为本，与用户紧密协作， <b>面对面沟通</b> ， <b>尽早发布增量</b> ，小而自主的开发团队。适用于规模小的项目。	
原型方法	按照实现功能划分	水平原型	行为原型，用于 <b>界面</b> 。细化需求但并未 <b>实现功能</b> 。
		垂直原型	结构化原型，用于复杂 <b>算法的实现</b> ，实现了部分功能。
	按照最终结果划分	抛弃式	探索式原型，解决需求不确定性、 <b>二义性</b> 、不完整性、含糊性等。
		演化式	逐步演化为最终系统，用于易于 <b>升级和优化</b> 的场合，适用于Web项目。

# 软件工程

## 3、敏捷方法

01

xp极限编程

高效、低风险、**测试先行**（先写测试代码，再编写程序）

02

Cockburn 水晶方法

**不同**项目，不同策略。

03

SCRUM并列争求法

**迭代**。30天为一个迭代周期，按照需求优先级实现。

04

FDD功用驱动方法

开发人员**分类**。分为指挥者（首席程序员）、类程序员。

05

开放式源码

**虚拟团队**，开发成员分布各地。

06

ASD自适应方法

**预测—协作—学习**

# 软件工程

- 敏捷方法是一种以人为核心、迭代、循序渐进的开发方法。敏捷型方法主要有两个特点，这也是其区别于其他方法，尤其是重型方法的最主要的特征。
  - ① 敏捷型方法是“适应性”而非“预设性”。重型方法试图对一个软件开发项目在很长的时间跨度内做出详细的计划，然后依计划进行开发。这类方法在计划制定完成后拒绝变化。而敏捷型方法则欢迎变化。
  - ② 敏捷型方法是，“面向人的”而非“面向过程的”。它们试图使软件开发工作能够利用人的特点，充分发挥人的创造能力，它们强调软件开发应当是一项愉快的活动。
- 敏捷方法的核心思想主要有下面三点：
  - (1) 敏捷方法是适应型，而非可预测型。
  - (2) 敏捷方法是以人为本，而非以过程为本。
  - (3) 迭代增量式的开发过程。

# ► 软件工程

➤ 敏捷方法的主要内容

敏捷方法的主要内容包括4个核心价值观和12条过程实践规则。

沟通	简单	反馈	勇气
设计者、开发者和客户三者之间的有效交流是开发成功的关键。	是设计和编码的指导原则，它强调只满足当前功能需求，尽量使代码简单化。	强调设计者、开发者和客户之间及时和详尽的意见反馈是开发成功的保证。	要求设计者和开发者在必需做出取舍或重构时，勇于抉择，勇于实践。



# 软件工程

实践类型	内容
计划游戏	快速制定计划、随着细节的不断变化而完善。
小型发布	系统的设计要能够尽可能早地交付。
隐喻	找到合适的比喻传达信息。
简单设计	只处理当前的需求，使设计保持简单。
测试先行	先写测试代码，然后再编写程序。
重构	重新审视需求和设计，重新明确地描述它们以符合新的和现有的需求。
结对编程	两个程序员在一个计算机上共同工作。一个人输入代码，而另一个人审查。
集体代码所有制	开发人员轮换完成系统不同领域中不同模块的不同任务。每个人都对程序负责。
持续集成	可以按日甚至按小时为客户提供可运行的版本。
每周工作40个小时	保证工作质量。
现场客户	系统用户代表全程配合XP团队。
编码标准	规范代码的编写。

# ► 软件工程

4、软件重用是指在两次或多次不同的软件开发过程中**重复使用相同或相似软件元素的过程**。软件元素包括需求分析文档、设计过程、设计文档、程序代码、测试用例和领域知识等。软件重用可区别为横向重用和纵向重用。

横向重用	指重用不同应用领域中的软件元素，例如数据结构、分类算法和人机界面构件等。
纵向重用	指在一类具有较多公共性的应用领域之间进行软件重用。

# ► 软件工程

5、逆向工程（Reverse Engineering）与逆向工程相关的概念有重构、设计恢复、再工程和正向工程。

（1）**重构（restructuring）**。重构是指在同一抽象级别上转换系统描述形式。

（2）**设计恢复（design recovery）**。设计恢复是指借助工具从已有程序中抽象出有关数据设计、总体结构设计和过程设计等方面的信息。

（3）**再工程（re-engineering）**。再工程是指在逆向工程所获得信息的基础上，**修改或重构已有的系统**，产生系统的一个**新版本**。再工程是对现有系统的重新开发过程，包括逆向工程、新需求的考虑过程和正向工程三个步骤。它不仅能从已存在的程序中重新获得设计信息，而且还能使用这些信息来重构现有系统，以改进它的综合质量。在利用再工程重构现有系统的同时，一般会增加新的需求，包括增加新的功能和改善系统的性能。

（4）**正向工程（Forward Engineering）**。正向工程是指不仅从**现有系统中恢复设计信息**，而且使用**51CTO 学堂**该信息去改变或重构现有系统，以改善其整体质量。

# ► 软件工程

- 逆向工程导出的信息可以分为如下4个抽象层次：

实现级别	抽象的语法树、符号表、过程的设计表示。
结构级别	反映程序分量之间相互依赖关系的信息，如调用图、结构图、程序和数据结构。
功能级别	反映程序段功能及程序段之间关系的信息，如数据和控制流模型。
领域级别	反映程序分量或程序实体与应用领域概念之间对应关系的信息，如E-R模型。

- 逆向工程的完备性可以用在某一个抽象层次上提供信息的详细程度来描述。抽象层次越高完备性越低，通过逆向工程恢复的难度越大。

# 软件工程

6、软件过程模型的基本概念：软件过程是制作软件产品的一组活动以及结果，这些活动主要由软件人员来完成，软件活动主要有：

- ① **软件描述**。必须定义软件功能以及使用的限制。
- ② **软件开发**。也就是软件的设计和实现，软件工程师制作出能满足描述的软件。
- ③ **软件有效性验证**。软件必须经过严格的验证，以保证能够满足客户的需求。
- ④ **软件进化**。软件随着客户需求的变化不断改进。

# ► 软件工程

## 7、开发模型

类型		特征
瀑布模型		结构化方法。开发阶段性、 <b>需求明确</b> 、文档齐全、风险控制弱。
演化模型	原型模型	原型方法。分为原先开发与目标软件开发。 <b>需求不明确</b> 、需开发环境和工具的支持。
	螺旋模型	瀑布与原型模型结合体。适用于 <b>大型、复杂、风险</b> 项目。
喷泉模型		面向对象方法。 <b>复用好</b> 、开发过程 <b>无间隙</b> 、节省时间。
快速应用开发 RAD		基于构件的开发方法。 <b>用户参与</b> 、开发或复用构件、 <b>模块化要求高</b> ，不适用新技术。
RUP/UP		基于构件的开发方法。 <b>用例驱动、架构为中心、迭代、增量</b> 。

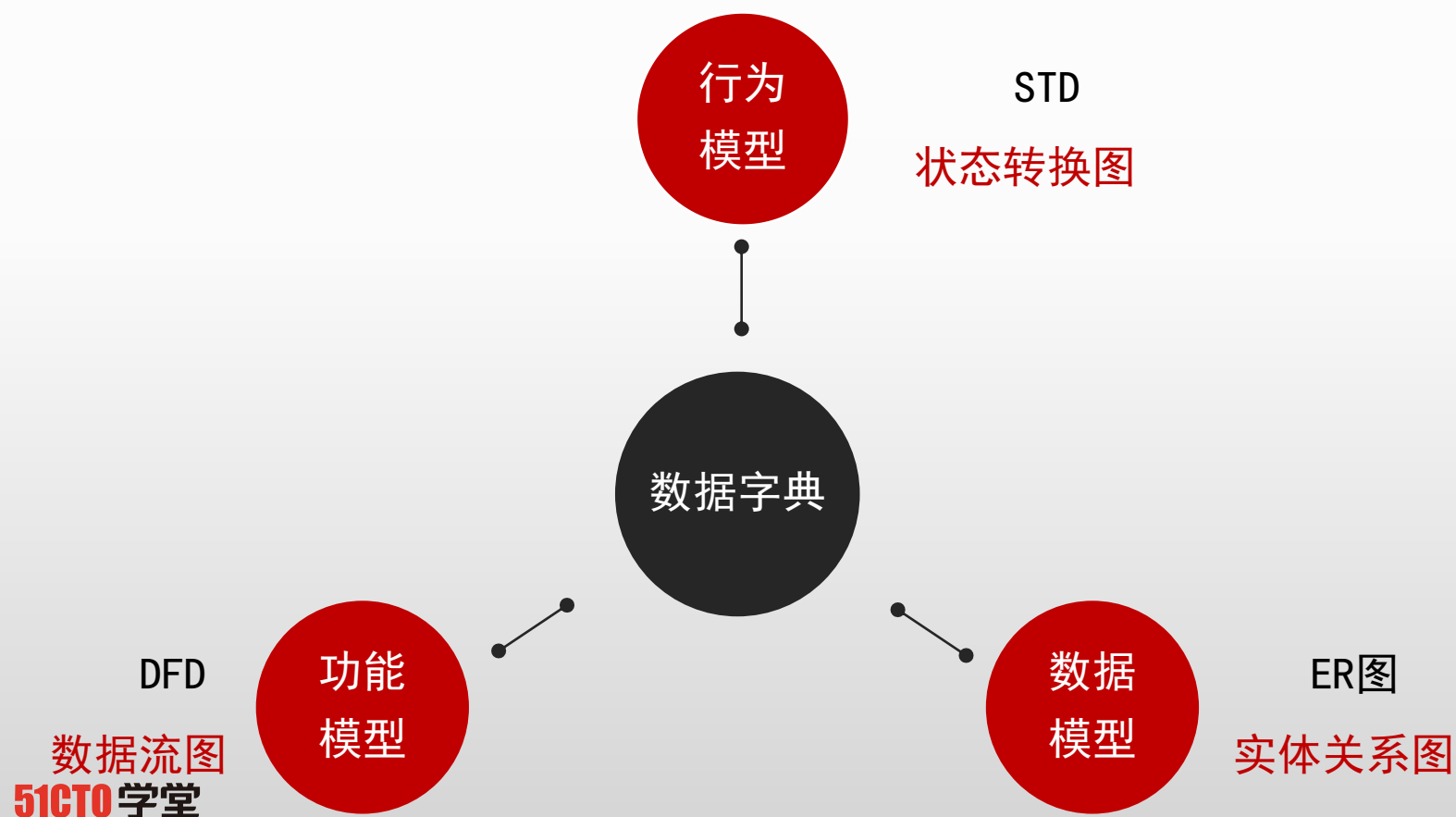
# ► 软件工程

8、RUP强调要采用迭代和增量的方式来开发软件，把整个项目开发分为多个迭代过程。在每次迭代中，只考虑系统的一部分需求，进行分析、设计、实现、测试和部署等过程，每次迭代是在已完成部分的基础上进行的，每次增加一些新的功能实现，以此进行下去，直至最后项目的完成。软件开发采用迭代和增量的方式有以下好处：

- ① 在软件开发的早期就可以对关键的、影响大的风险进行处理。
- ② 可以提出一个软件体系结构来指导开发。
- ③ 可以更好地处理不可避免的需求变更。
- ④ 可以较早地得到一个可运行的系统，鼓舞开发团队的士气，增强项目成功的信心。
- ⑤ 为开发人员提供一个能更有效工作的开发过程。

# ▶ 系统分析

1、结构化分析（Structured Developing）是面向数据流的需求分析方法，它的思想是通过功能分解方式把系统功能分解到各个模块中。





# ▶ 系统分析

2、面向对象分析的**关键**是面向对象的建模，面向对象建模主要使用uml图。建模可以分为**分析模型**与**设计模型**。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。

- 基于uml的面向对象需求分析过程大致分为以下几个步骤：
- ① **利用用例及用例图表示需求**。从业务需求描述出发获取执行者和场景；对场景进行汇总、分类、抽象，形成用例；确定执行者与用例、用例与用例图之间的关系，生成用例图。其中用例之间的关系主要有**包含**、**扩展**和**泛化**。

类型	含义
包含	两个或两个以上的用例的公共行为可以表示为用例之间的包含关系。
扩展	一个用例混合了两种或两种以上不同的场景（分支）可以表示为用例之间的扩展关系。
泛化	多个用例拥有一种共性，可以将共性抽象为父用例，其他用例作为泛化关系的子用例。

# ▶ 系统分析

② 利用包图及类图表示目标软件系统的总体框架结构。根据领域知识、业务需求描述和既往经验设计目标软件系统的顶层架构；从业务需求描述中提取“关键概念”，形成领域概念模型；从概念模型和用例出发，研究系统中主要的类之间的关系，生成类图。其中类之间的关系有：

关联	是拥有的关系。提供了不同类的对象之间的结构关系，它在一段时间内将多个类的实例连接在一起。
聚合	是整体与部分的关系，各自具有不同的生命周期。
组合	是整体与部分的关系，具有相同的生命周期。
依赖	是使用的关系。两个类A和B，如果B的变化可能会引起A的变化。
泛化	是父类与子类之间的关系。是继承的反关系。
实现	一个或多个类可以实现一个接口，每个类分别实现接口中的操作。

# 系统设计

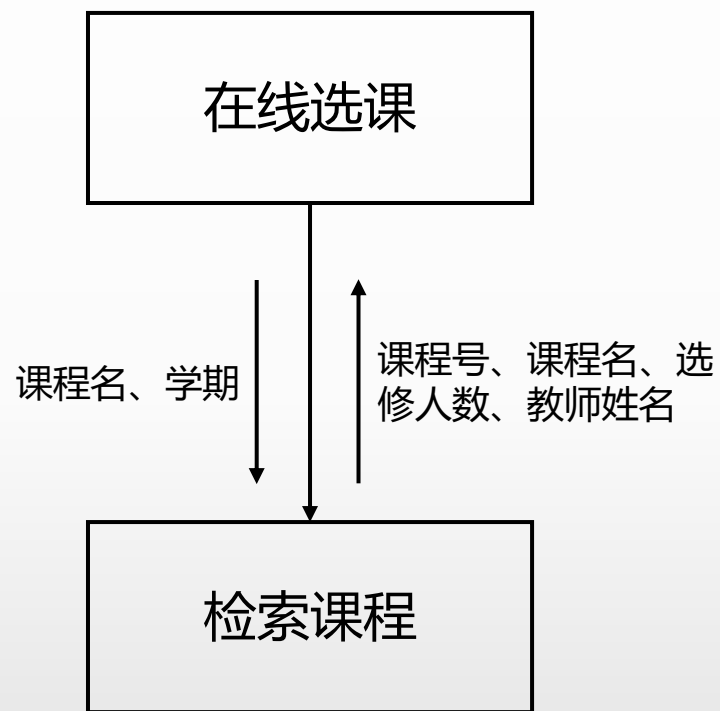
1、系统设计阶段又称为物理设计阶段，其任务是根据系统规格说明书中规定的功能要求，设计新系统的物理模型，为下一阶段的系统实施工作奠定基础。

概要设计	详细设计
又称为系统 <b>总体结构设计</b> ，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 <b>模块结构图</b> ，即 <b>系统结构图</b> 。软件概要设计将软件需求转化为 <b>数据结构</b> 和 <b>软件的系统结构</b> 。	详细设计也称为低层设计,即对结构图进行细化,得到 <b>详细的数据结构与算法</b> 。详细设计主要包括：网络设计、代码设计、输入输出设计、处理流程设计、数据存储设计、用户界面设计、安全性和可靠性设计等。

# 系统设计

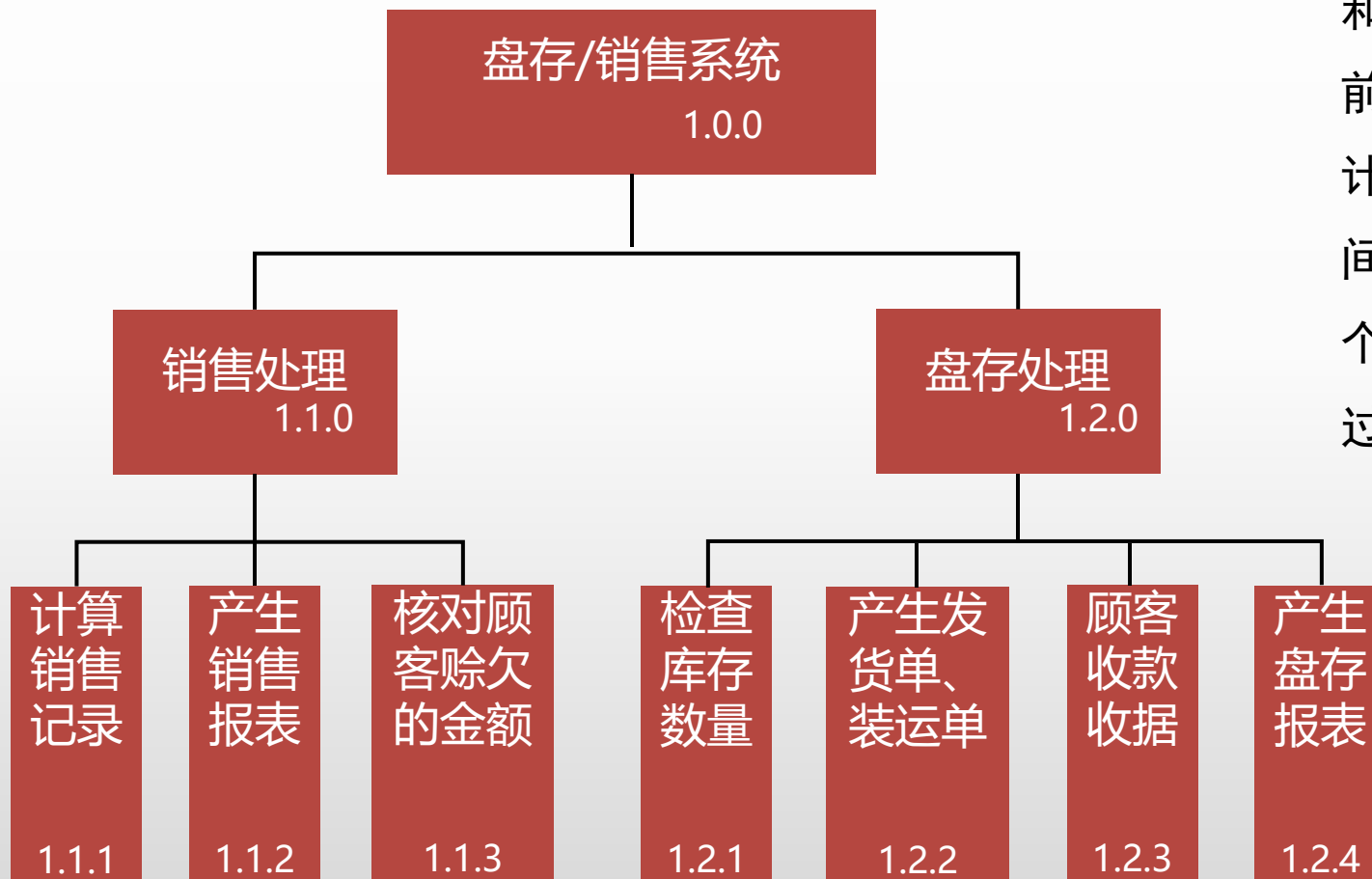
- 2、概要设计过程中主要使用**模块结构图**、**层次图**和**HIP0图**。
- 3、详细设计过程中主要使用：**程序流程图**、**伪代码**、**盒图**。
- 4、结构化设计是一种面向数据流的方法，它以数据流图和数据字典等文档为基础，是一个自顶向下、逐步求精和模块化的过程。结构化设计具有如下特征：
  - **信息隐蔽与隐藏**。信息隐蔽是开发整体程序结构时使用的法则，即将每个程序的成分隐蔽或封装在一个单一的设计模块中，并且尽可能少地暴露其内部的处理过程。通过信息隐蔽可以提高软件的可修改性、可测试性和可移植性。
  - **模块化**
  - **高内聚与低耦合**

◆ **系统结构图** (Structure Chart, SC) 又称为**模块结构图**，它是软件概要设计阶段的工具，反映系统的功能实现和模块之间的联系与通信，包括各模块之间的层次结构，即反映了系统的总体结构。



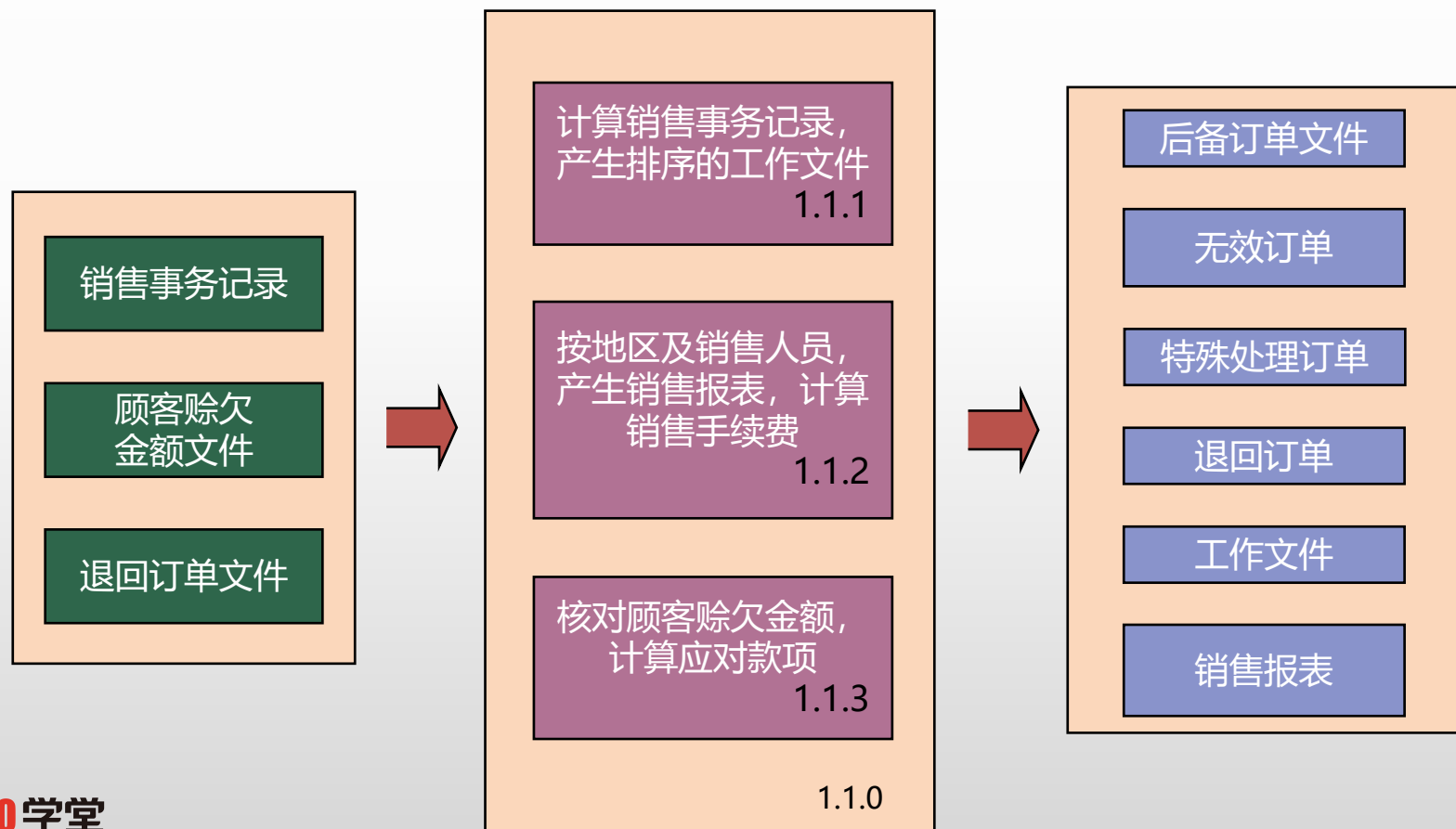
模块结构图

## ◆ HIPO图

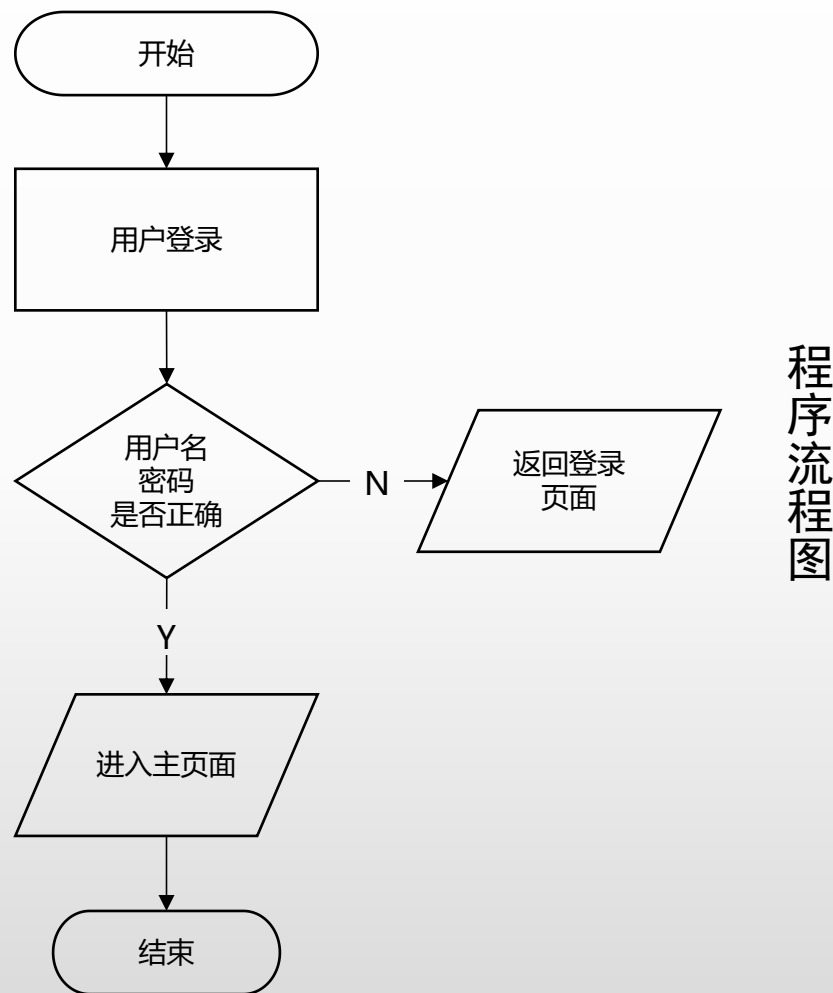


HIPO 图由层次结构图和IPO 图两部分构成,前者描述整个系统的设计结构以及各类模块之间的关系,后者描述某个特定模块内部的处理过程和输入/输出关系。

- ◆ **IPO图**用来描述每个模块的输入、输出和数据加工。IPO图的主体是处理过程说明，可以采用流程图、判定树/表等来进行描述。

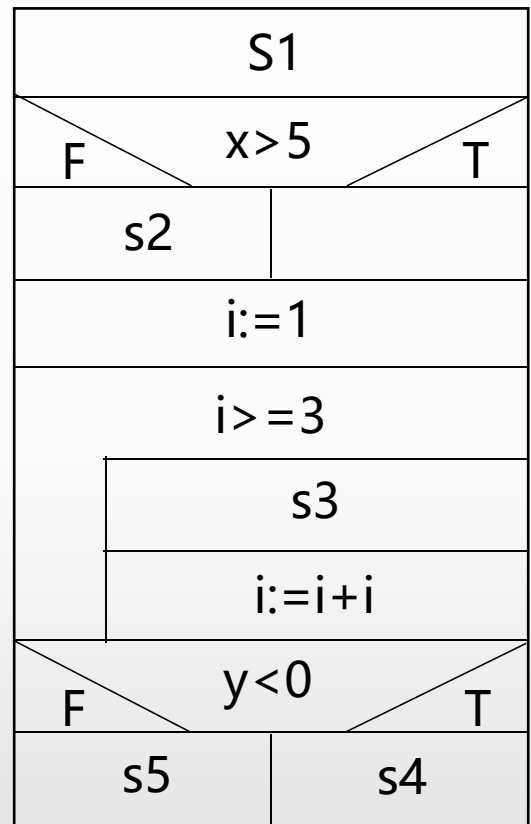


◆ **程序流程图** (Program Flow Diagram, PFD) 用一些图框表示各种操作，它独立于任何一种程序设计语言，比较直观、清晰，易于学习掌握。流程图中只能包括5种基本控制结构：顺序型、选择型、WHILE循环型(当型循环)、UNTIL循环型(直到型循环)和多分支选择型。





◆盒图（N-S图）在N-S图中也包括五种控制结构，分别是顺序型、选择型、WHILE循环型（当型循环）、UNTIL循环型（直到型循环）和多分支选择型，任何一个N-S图都是这五种基本控制结构相互组合与嵌套的结果，很容易表示层次关系，有结构化的特点。N-S图避免了流程图在描述程序逻辑时的随意性与灵活性。



盒图

◆ **过程设计语言** (Process Design Language, PDL) 也称为结构化语言或 **伪代码** (pseudo code), 它是一种混合语言, 采用自然语言的词汇和结构化程序设计语言的语法, 用于描述处理过程怎么做, 类似于编程语言。

```
i ← 1  
S ← 0  
While i ≤ 10  
    S ← S + i  
    i ← i + 1  
End While  
Print S
```

伪代码  
(过程设计语言)

# 系统设计

5、面向对象的设计过程可以分为：

① 设计用例实现方案

UML的交互图（顺序图、协作图）适于用例实现方案的表示. 该设计方法包含如下三个步骤：

（1）提取边界类、实体类和控制类。

实体类	边界类	控制类
<ul style="list-style-type: none"><li>● 领域模型中的类</li><li>● 需要进行持久化存储</li><li>● 名词</li></ul>	<ul style="list-style-type: none"><li>● 用于系统接口与系统外部进行交互，如窗口、菜单栏、二维码等。</li></ul>	<ul style="list-style-type: none"><li>● 控制其他类工作的类</li><li>● 体现应用程序的执行逻辑</li><li>● 动词</li></ul>

# 系统设计

(2) 构造交互图。UML交互图，以交互图作为用例的精确实现方案。

(3) 根据交互图精化类图。

在UML交互图中，对每个类的对象都规定了它必须响应的消息以及类的对象之间的消息传递通道。

## ② 设计技术支撑方案

在许多软件项目中，应用功能往往都需要一组技术支撑机制为其提供服务。例如，数据持久存储服务、安全控制服务、分布式事务管理服务、并发与同步控制服务和可靠消息服务等。

## ③ 设计用户界面

## ④ 精化设计模型

之前的设计模型可能是粗粒度的，这一步对设计模型进行更加详细的设计。

# 系统设计

## 6、工具的区别

序列图	协作图
序列图主要用来更直观的表现各个对象交互的时间顺序，将体现的重点放在以时间为参照，各个对象发送、接收消息，处理消息，返回消息的时间流程顺序，也称为时序图。	协作图是一种类图,强调参与交互的各个对象的结构信息和组织。
强调时间顺序	强调空间结构

# 系统设计

状态图	活动图
用来描述一个特定的对象所有可能的状态,以及由于各种事件的发生而引起的状态之间的转移和变化。	将进程或其他计算的结构展示为计算内部一步步的控制流和数据流, 主要用来描述系统的动态视图。
状态图主要描述 <u>行为的结果</u> 。	活动图主要描述 <u>行为的动作</u> 。
用于对系统的动态方面建模。	用于对系统的动态方面建模。

# 系统设计

流程图	活动图
描述处理过程。主要控制结构是顺序、分支和循环，各个处理过程之间有严格的顺序和时间关系。	描述的是对象活动的顺序关系所遵循的规则，它着重表现的是系统的行为，而非系统的处理过程。
不能表示并发活动的情形	能够表示并发活动的情形
面向过程	面向对象

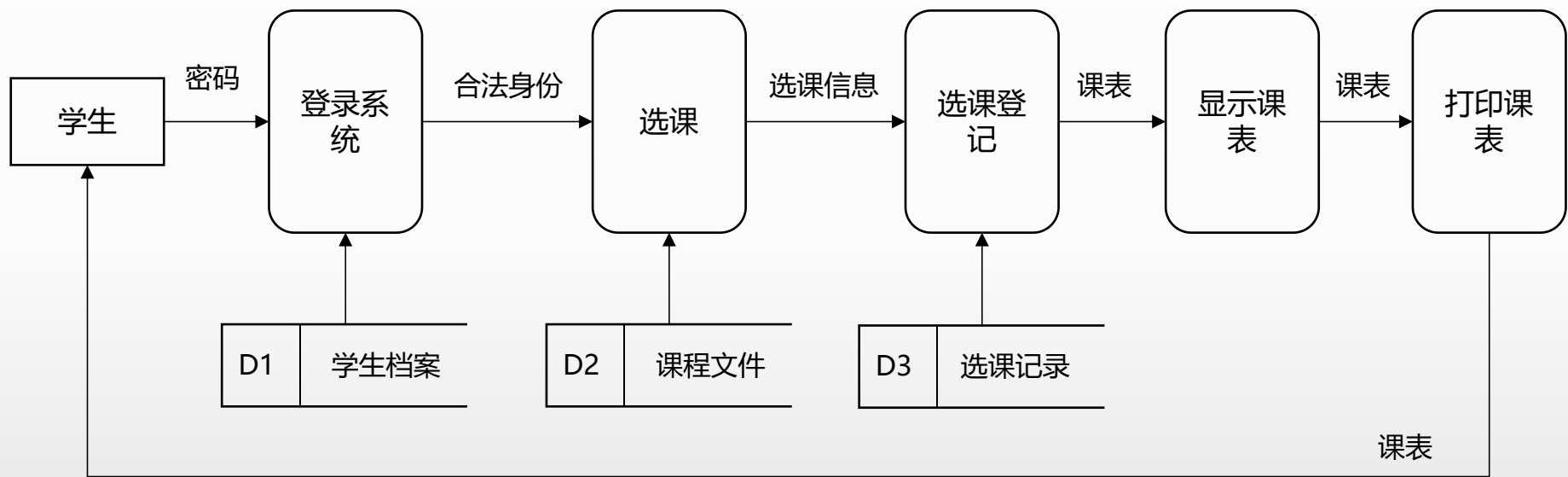
# 系统设计

数据流图与系统流程图的区别：

- (1) 数据流图中的处理过程可并行;系统流程图在某个时间点只能处于一个处理过程。
- (2) 数据流图展现系统的数据流; 系统流程图展现系统的控制流。
- (3) 数据流图展现全局的处理过程，过程之间遵循不同的计时标准;系统流程图中处理过程遵循一致的计时标准。

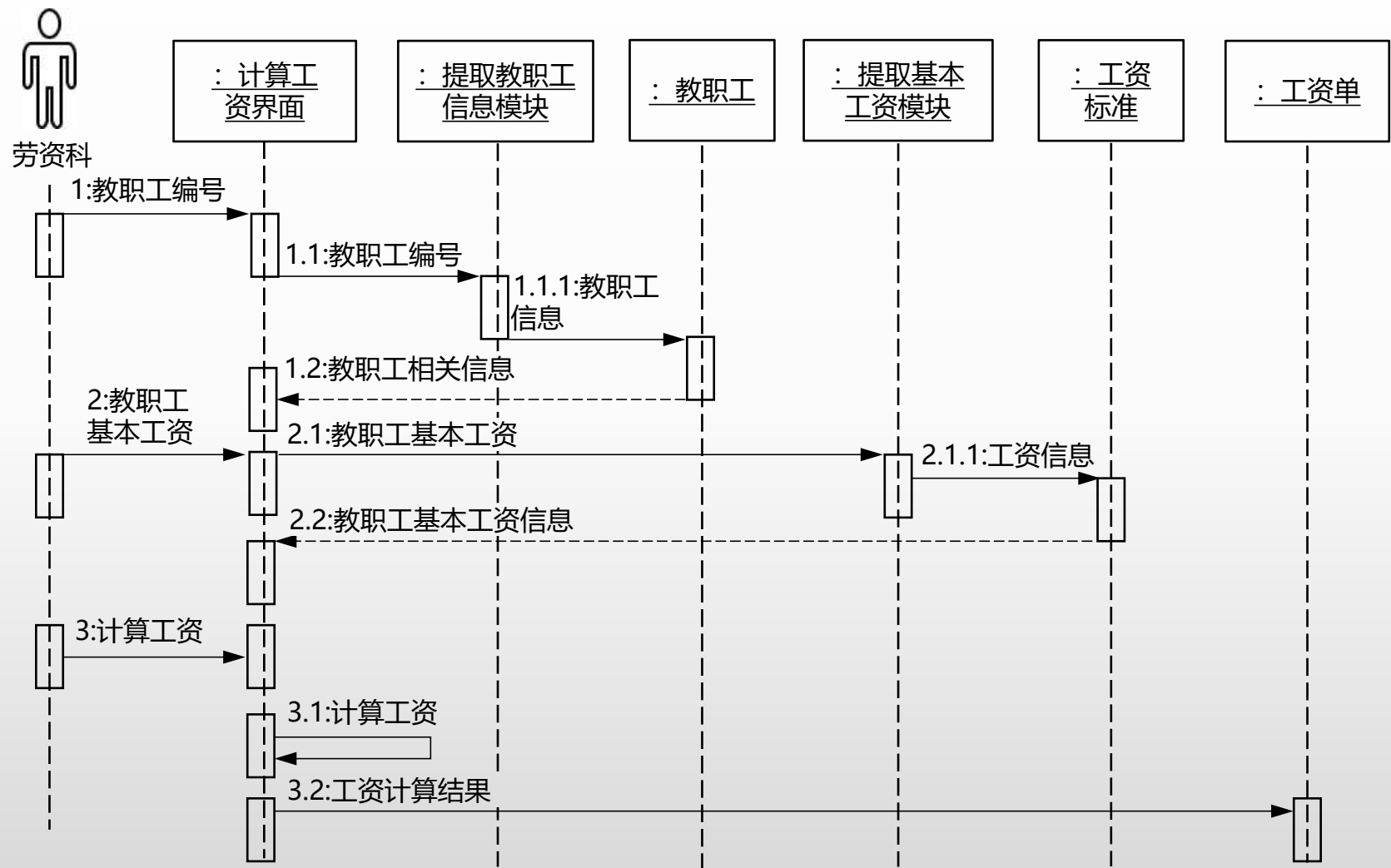


# 系统设计

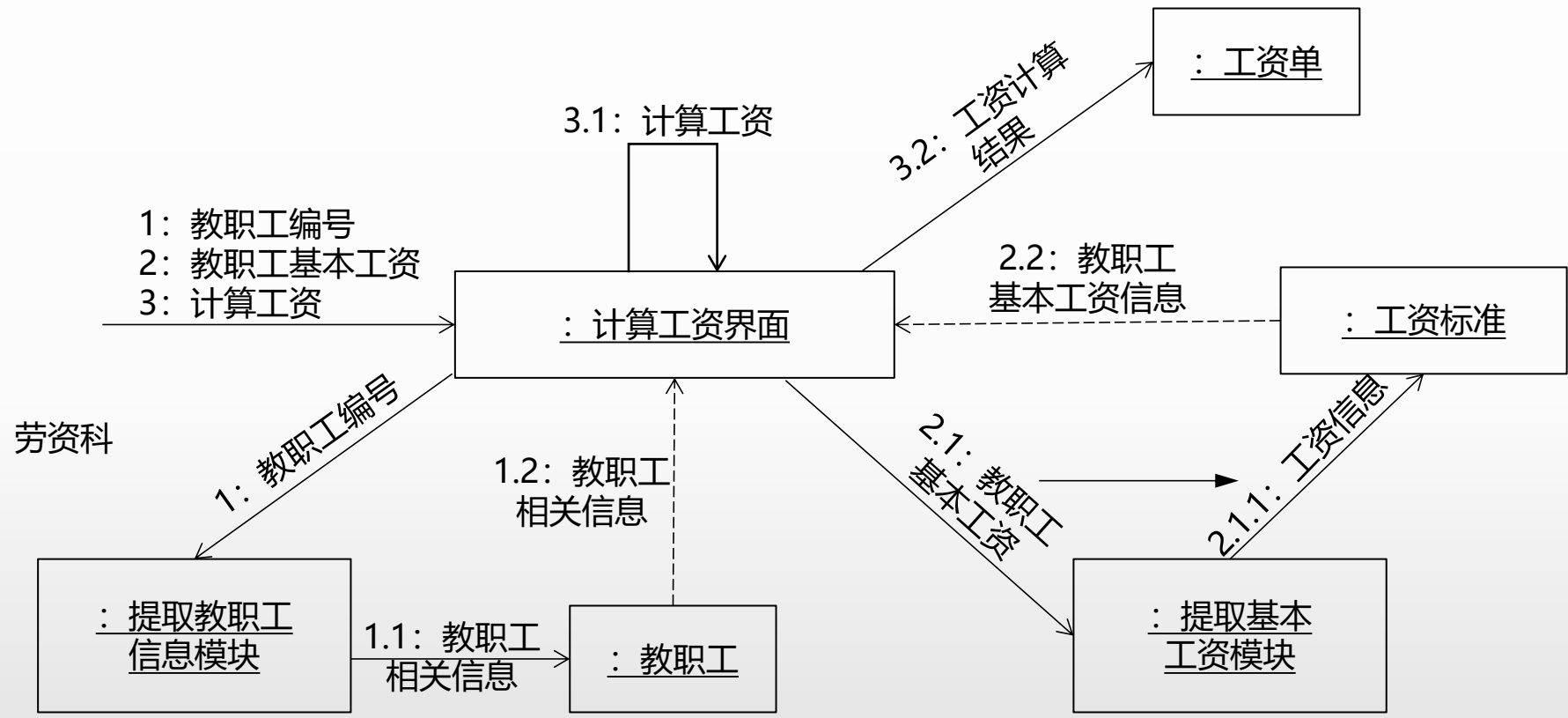


数据流图

# 系统设计

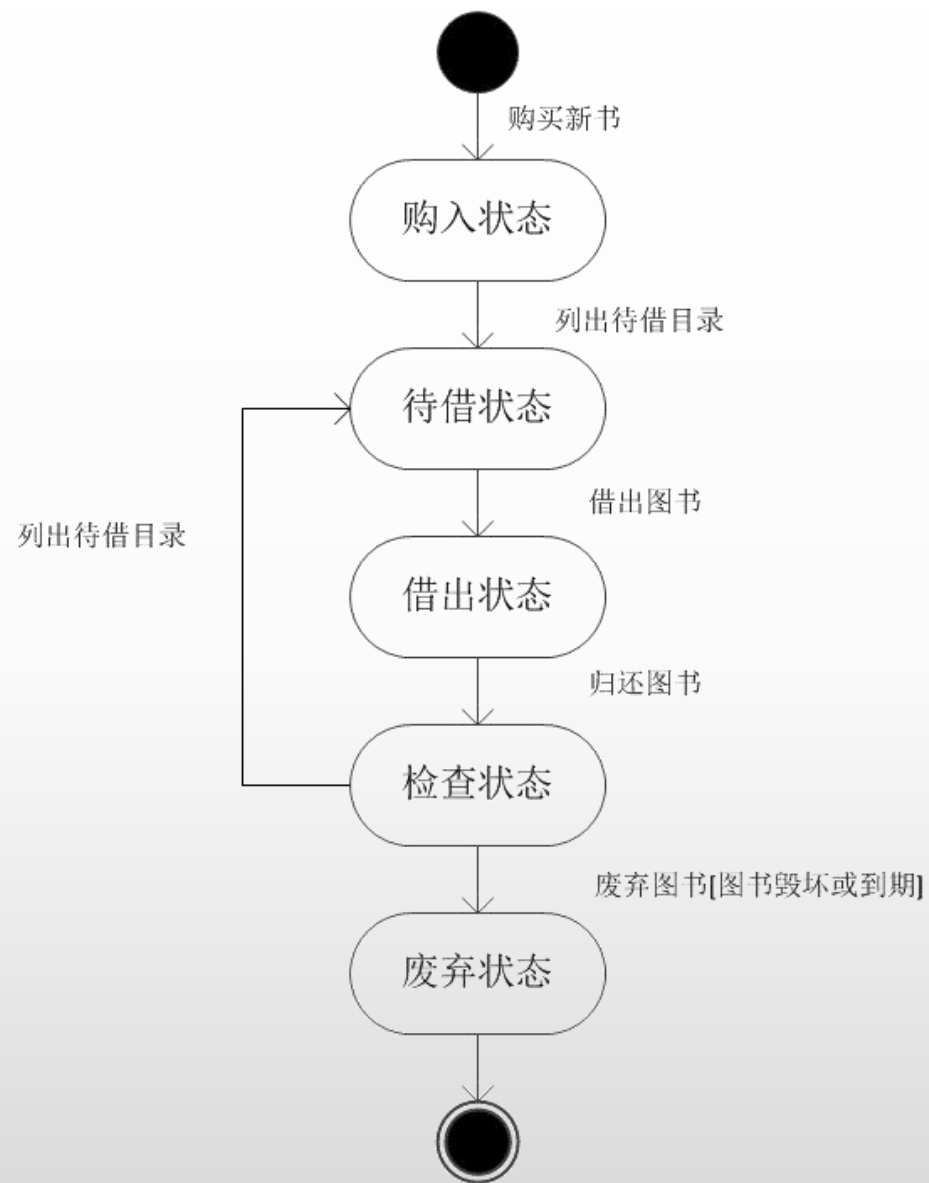


# 系统设计



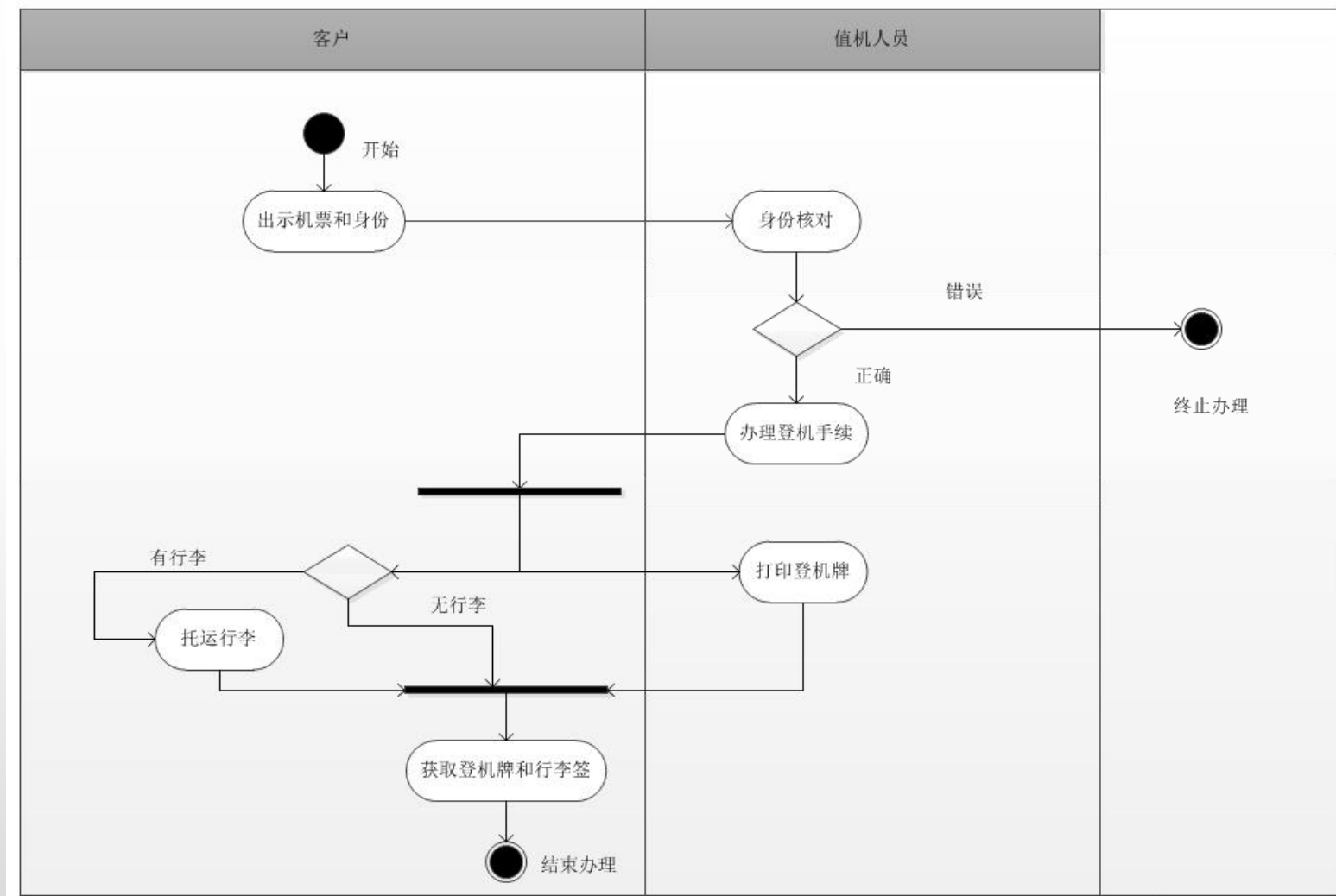
协作图

# 系统设计



状态图

# 系统设计



# ► 架构设计

- 1、软件架构设计主要关注软件**构件的结构**、**属性**和**交互作用**，并通过**多种视图**全面描述特定系统的架构。
- 2、软件体系结构（架构）风格是描述**某一特定应用领域中系统组织方式的惯用模式**。它定义了一个系统家族，即一个体系结构定义一个**词汇表**和**一组约束**。
  - **词汇表**中包含一些**构件和连接件类型**。
  - **约束**指出系统是如何将这些构件和连接件组合起来的。体系结构风格反映了领域中众多系统所共有的**结构**和**语义特性**，并指导如何将各个模块和子系统有效地组织成一个完整的系统。

# 架构设计

## 3、经典架构风格

风格		说明
数据流风格	批处理	每一步处理都是独立的，并且每一步是顺序执行的。只有当前一步处理完，后一步处理才能开始。数据必须是完整的，以整体的方式传递。 如日志分析、计费程序等。不适合用来设计交互式应用系统。
	管道过滤器	每个构件都有一组输入和输出，构件读输入的数据流，经过内部处理，然后产生输出数据流。这个过程通常通过对输入流的变换及增量计算来完成，所以在输入被完全消费之前，输出便产生了。 如传统的编译器、UNIX管道等。
调用/返回风格	主程序子程序	这种风格一般采用单线程控制，把问题划分为若干处理步骤，为主程序的正确性取决于它调用的子程序的正确性。 如开发语言。
	面向对象	数据的表示和它们的相应操作被封装起来，对象的行为体现在其接受和请求的动作中。对象具有封装性，一个对象的改变不会影响其他对象。 如面向对象开发语言。如果一个对象要调用另一个对象，则必须知道它的标识和名称；会产生连锁反应
	层次	每层为上一层提供服务，使用下一层的服务，只能见到与自己邻接的层。在层次结构中，修改某一层，最多影响其相邻的上下两层（通常只能影响上层）。上层必须知道下层的身份，不能调整层次之间的顺序。 如TCP/IP协议。很难找到一种合适和正确的层次划分方法。

# 架构设计

风格	作用	说明
构件风格	进程通信	进程间消息传递的方式可以是点对点、异步或同步方式，以及远程过程（方法）调用等
	事件驱动	当某个事件被触发时，系统自动调用在这个事件中注册的所有过程。如断点调试、新闻，公众号等的订阅信息。构件放弃了对计算的控制权，完全由系统来决定。
虚拟机风格	解释器	解释器通常包括一个完成解释工作的解释引擎、一个包含将被解释的代码的存储区、一个记录解释引擎当前工作状态的数据结构，以及一个记录源代码被解释执行进度的数据结构。如JVM。
	基于规则的系统	基于规则的系统包括规则集、规则解释器、规则/数据选择器和工作内存。一般用在人工智能领域和DSS中。
仓库风格	数据库系统	构件分为中央共享数据源、独立处理单元。构件控制中央共享数据。如IDE集成开发环境。
	黑板	黑板系统包括知识源、黑板和控制三个部分。知识源包括若干独立计算的不同单元，提供解决问题的知识。知识源响应黑板的变化，也只修改黑板；黑板是一个全局数据库，包含问题域解空间的全部状态，是知识源相互作用的唯一媒介；知识源响应是通过黑板状态的变化来控制的。黑板系统通常应用在对于解决问题没有确定性算法的软件中。如语音识别，信号处理。
	超文本	静态网页
闭环过程控制		将过程输出的指定属性维护在一个特定的参考值（设定值），将事务处理看成输入、加工、输出、反馈、再输入的一个持续的过程模型。实例，空调的温度自动调节器（设定值是温度），巡航系统（设定值是速度）。
C2风格		通过连接件绑定在一起按照一组规则运作的并行构件网络。（构件、连接件）



# 架构设计

## 4、二层 C/S 结构的缺点主要有：

- (1) 服务器的负荷太重，难以管理大量的客户机，系统的性能容易变坏。
- (2) 数据安全性不好。因为客户端程序可以直接访问数据库服务器，那么，在客户端计算机上的其他程序也可想办法访问数据库服务器，从而使数据库的安全性受到威胁。与二层C/S架构相比，在三层C/S架构中，增加了一个应用服务器。可以将整个应用逻辑驻留在应用服务器上，而只有表示层存在于客户机上。这种客户机称为瘦客户机。三层C/S架构将应用系统分成表示层、功能层和数据层三个部分。

层次	功能
表示层	用户接口，检查用户输入的数据，显示输出数据。
功能层	业务逻辑层，是将具体的业务处理逻辑编入程序中。
数据层	对DBMS进行管理和控制。

# 架构设计

5、与传统的二层架构相比，三层C/S架构具有以下优点：

（1）允许合理地划分三层的功能，使之在逻辑上保持相对独立性，从而使整个系统的逻辑结构更为清晰，能提高系统的可维护性和可扩展性。

（2）允许更灵活、有效地选用相应的平台和硬件系统，使之在处理负荷能力上与处理特性上分别适应于结构清晰的三层，并且这些平台和各个组成部分可以具有良好的可升级性和开放性。

（3）系统的各层可以并行开发，各层也可以选择各自最适合的开发语言，使之能并行且高效地进行开发，达到较高的性能价格比。对每一层的处理逻辑的开发和维护也会更容易些。

（4）利用功能层可以有效地隔离表示层与数据层，未授权的用户难以绕过功能层而利用数据库工具或黑客手段去非法地访问数据层，这就为严格的安全管理奠定了坚实的基础。

# 架构设计

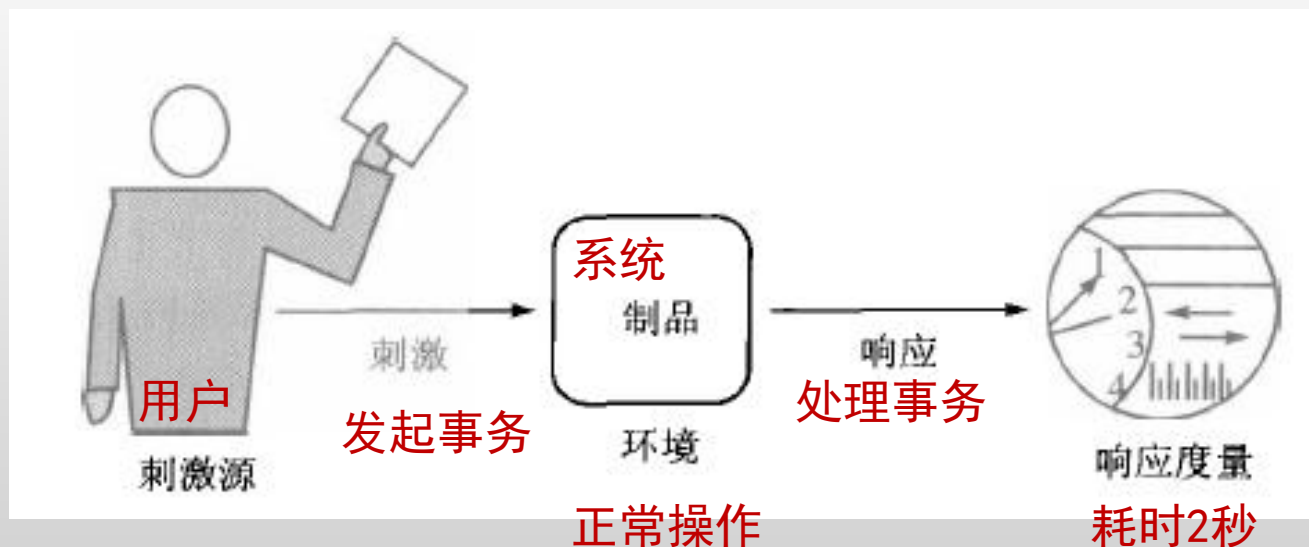
## 6、uml与系统架构

逻辑视图	主要支持 <b>功能性需求</b> ，即在为用户提供服务方面系统所应该提供的功能。逻辑视图来记录设计元素的功能和概念接口，设计元素的功能定义了它本身在系统中的角色，这些角色包括功能、性能等。主要使用类图。
进程（过程）视图	进程架构考虑一些非功能性的需求，如性能和可用性。它解决 <b>并发性、分布性</b> 等问题，以及逻辑视图的主要抽象如何与进程结构相配合在一起，即在哪个控制线程上，对象的操作被实际执行。
实现（开发）视图	描述系统的各部分如何被组织为模块和组件即开发环境中软件的 <b>静态组织结构</b> 。该视图通常包含包图和组件图。
部署视图	把构件部署到一组物理节点上，表示 <b>软件到硬件的映射和分布结构</b> 。展示了抽象部分如何映射到最终部署的系统中。
场景	需求的抽象，可以展示其他视图如何协同工作。

# 架构设计

## 7、质量场景的6个组成部分

- ① 刺激源：谁造成的刺激
- ② 刺激：一个响应系统的情况
- ③ 制品：系统被刺激的部分
- ④ 环境：刺激发生时，系统所处的状态
- ⑤ 响应：刺激所产生的结果
- ⑥ 响应度量指标：如何评估响应



# 架构设计

## 8、质量属性

属性		作用及要点	设计策略
性能		处理任务所需时间或单位时间内的处理量	控制资源的使用、并发机制、增加资源、先来先服务、设定优先级。
可用性		正常运行的时间比例，出现故障多久能启用备用系统。	心跳、Ping/Echo、异常、冗余/故障转移、检查点/回滚、事务、进程监控器。
安全性		系统向合法用户提供服务并阻止非法用户的能力。	用户授权、维护数据机密性与完整性、限制访问、入侵检测系统、追踪审计。
可修改性	可维护性	错误发生后进行局部性修改，对其他构件负面影响最小	信息隐藏、维持现有接口、限制通信路径、使用中介、运行时注册。
	可扩展性	使用新构件、改进或删除原有构件或特性	
	结构重组	重新组织构件及构件关系、灵活配置构件	
	可移植性	多样的环境（硬件平台、语言、操作系统等）	
易用性		在指定条件下使用时，软件产品被理解、学习、使用和吸引用户的能力	

# 架构设计

## 9、质量特性

质量特性	定义	关键词
敏感点	为了实现某种特定的质量属性，一个或多个构件所具有的特性。	对查询请求处理时间的要求将影响系统的数据传输协议和处理过程的设计
权衡点	影响多个质量属性的特征，是多个属性的敏感点	改变业务数据编码方式会对系统的性能和安全性产生影响
风险	不以标准术语出现。某些做法有一些隐患可能导致一些问题。	对系统某业务逻辑的描述尚未达成共识，这可能导致部分业务功能模块的重复，影响系统的可修改性；
非风险	某些做法是可行的、可接受的。	业务处理时间小于30毫秒，则将请求响应时间设定为1秒钟是可以接受的

# 架构设计

10、软件架构评估的方法，按基于的技术手段来看，可以分为三类：基于调查问卷或检查表的方式、基于场景的方式和基于度量的方式。

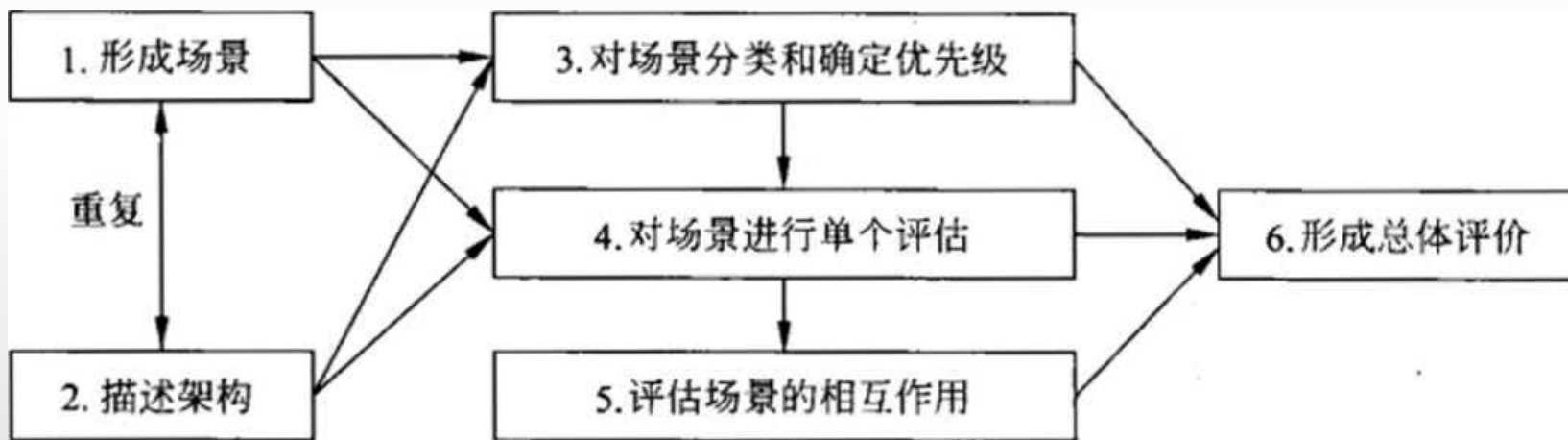
❑ 基于调查问卷或检查表的方式：该方式的关键是要设计好问卷或检查表，它充分利用系统相关人员的经验和知识，获得对架构的评估。其缺点是在很大程度上依赖于评估人员的主观推断。

❑ 基于场景的方式：基于场景的方式包括**架构权衡分析法**（Architecture Tradeoff Analysis Method, **ATAM**）和**软件架构分析方法**（Software Architecture Analysis Method, **SAAM**），以及**CBAM**（Cost Benefit Analysis Method）**成本收益分析方法**。通过分析软件架构对场景（也就是对系统的使用或修改活动）的支持程度，从而判断该架构对这一场景所代表的质量需求的满足程度。

❑ 基于度量的方式：制定一些定量值来度量架构，如代码行数等。要制定质量属性和度量结果之间的映射。

# 架构设计

11、SAAM的主要输入是问题描述、需求说明和架构描述，其分析过程主要包括场景开发、架构描述、单个场景评估、场景交互和总体评估。

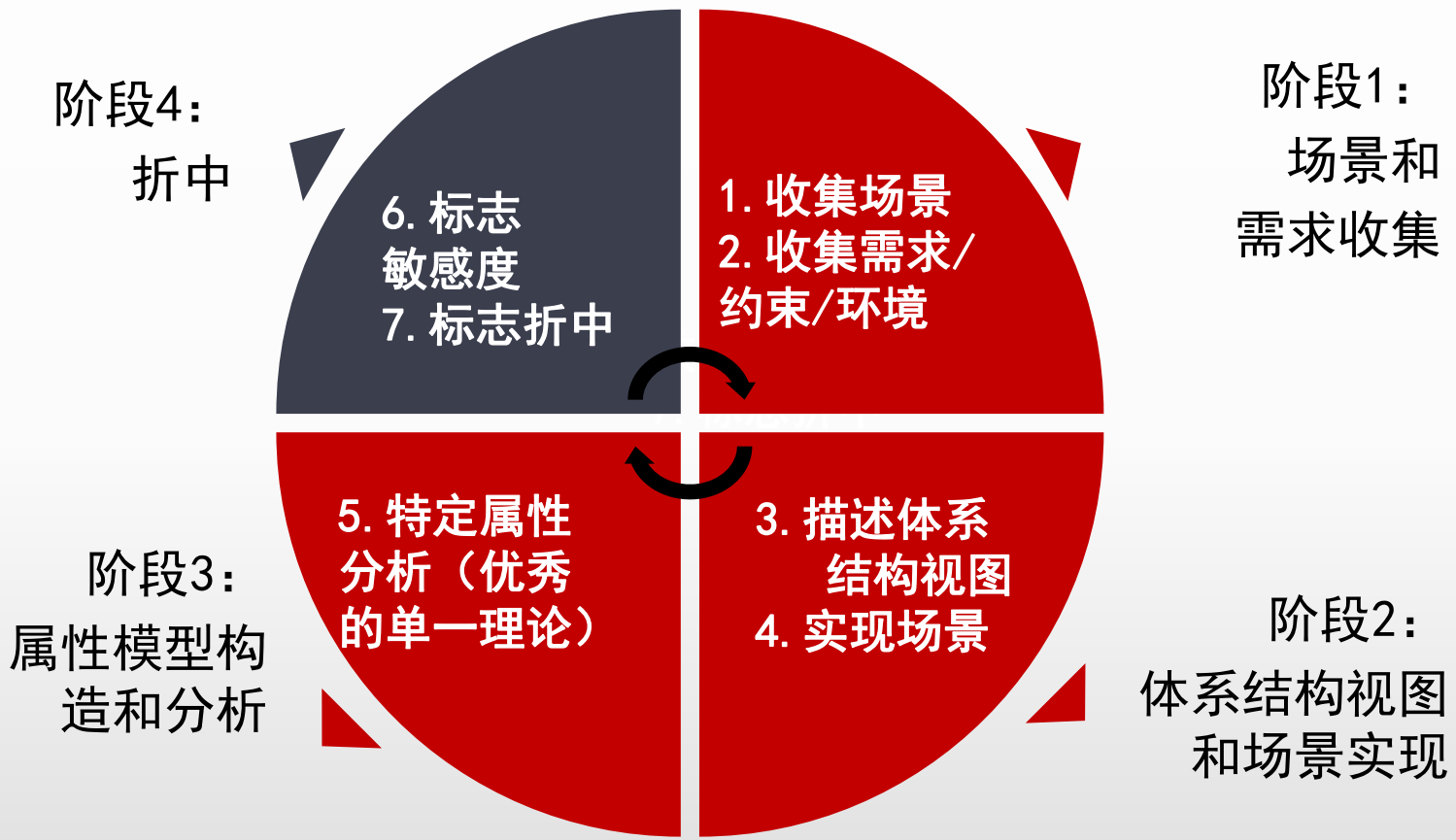




# ► 架构设计

12、架构权衡分析方法 ATAM是一种系统架构评估方法，主要在系统开发之前，针对性能、可用性、安全性和可修改性等质量属性进行评价和折中。ATAM可以分为4个主要的活动阶段，包括需求收集、架构视图描述、属性模型构造和分析、架构决策与折中，整个评估过程强调以属性作为架构评估的核心概念。

# 架构设计



最初的ATAM

# 架构设计





# 架构设计

## 13、特定领域软件架构 (Domain Specific Software Architecture, DSSA)

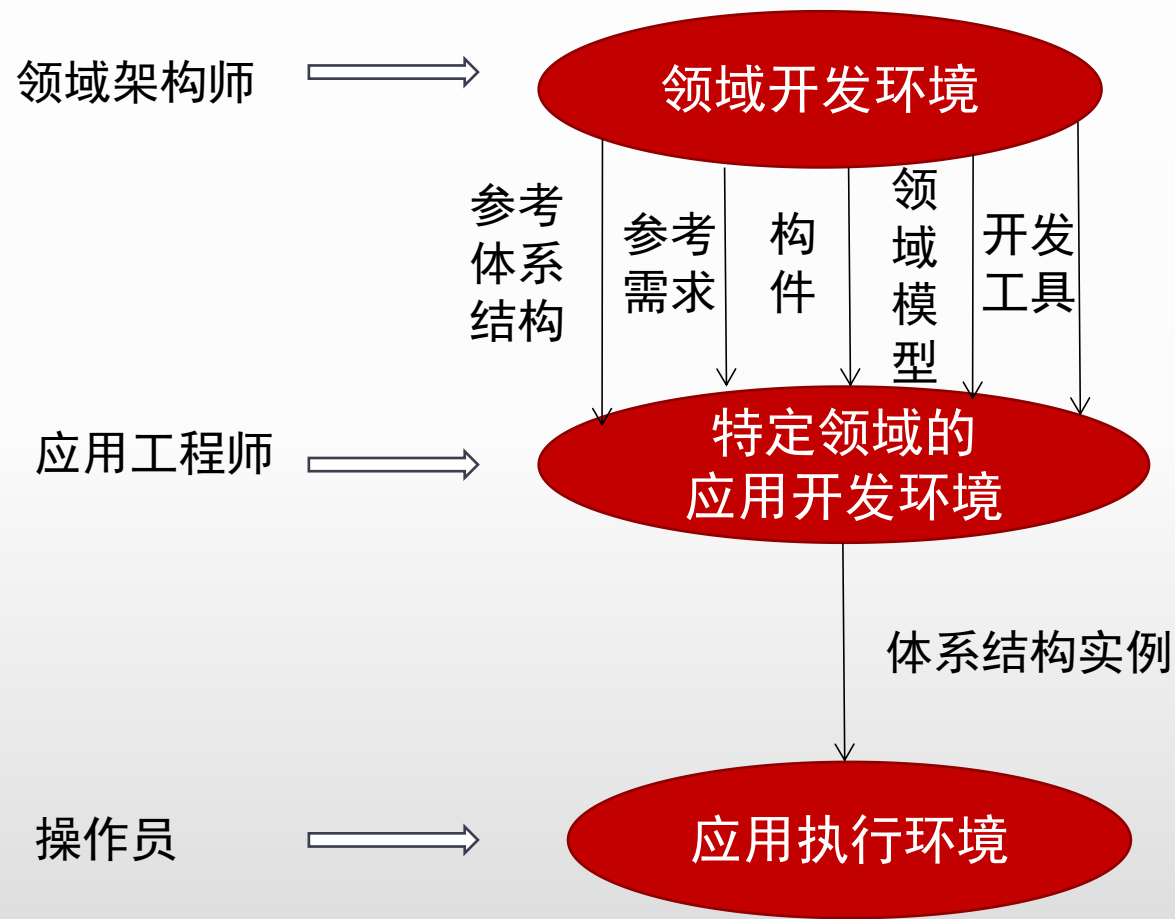
是在一个特定应用领域中，为一组应用提供组织结构参考的标准软件体系结构。

➤ 从功能覆盖的范围角度有两种理解DSSA中领域的含义的方式。

(1) 垂直域：定义了一个特定的系统族，包含整个系统族内的多个系统，结果是在该领域中可作为系统的可行解决方案的一个通用软件体系结构。

(2) 水平域：定义了多个系统和多个系统族中功能区域的共有部分。在子系统级上涵盖多个系统族的特定部分功能。

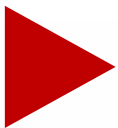
# 架构设计



# 架构设计

基本活动	主要目标
领域分析	获得领域模型。领域模型描述领域中系统之间共同的需求，即领域需求。
领域设计	获得特定领域软件架构。DSSA描述领域模型中表示需求的解决方案。
领域实现	依据领域模型和DSSA开发和组织可重用信息，并对基础软件架构进行实现。

参与角色：领域专家、领域分析人员、领域设计人员、领域实现人员



# 架构设计

14、基于体系结构（架构）的软件设计(Architecture-Based Software Design, ABSD)方法。ABSD方法是体系结构驱动，即指构成体系结构的商业、质量和功能需求的组合驱动的。在基于体系结构的软件设计方法中，采用**视角与视图**来描述软件架构，采用**用例**来描述功能需求，采用**质量场景**来描述质量需求。

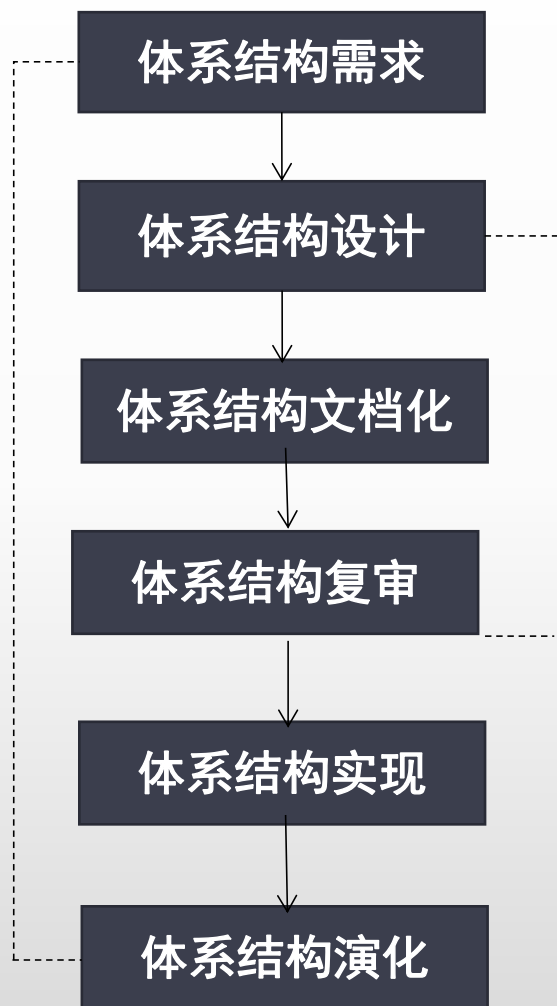
- ABSD方法是一个**自顶向下**，**递归细化**的方法，软件系统的体系结构通过该方法得到细化，直到能产生软件构件和类。
- ABSD方法有三个基础：
  - ① **第一个基础是功能的分解**。在功能分解中，ABSD方法使用已有的基于模块的内聚和耦合技术。
  - ② **第二个基础是通过选择体系结构风格**来实现质量和商业需求。
  - ③ **第三个基础是软件模板的使用**。软件模板利用了一些软件系统的结构。

# 架构设计

传统的软件开发过程可以划分为从概念直到实现的若干个阶段，包括问题定义、需求分析、软件设计、软件实现及软件测试等。如果采用传统的软件开发模型，软件体系结构的建立应位于需求分析之后，概要设计之前。传统软件开发模型存在开发效率不高，不能很好地支持软件重用等缺点。ABSDM模型把整个基于体系结构的软件过程划分为体系结构需求、设计、文档化、复审、实现和演化等6个子过程，得到细化，直到能产生软件构件和类。

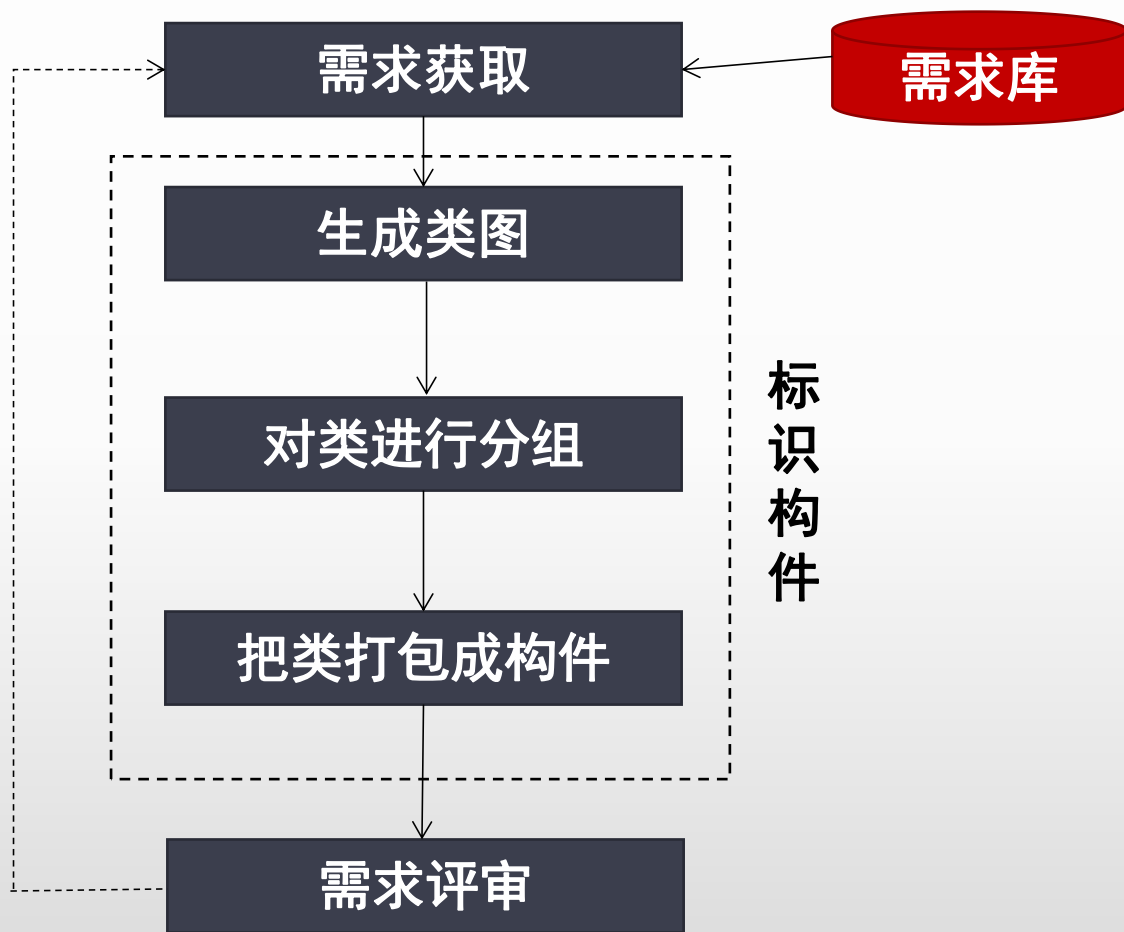


# 架构设计



体系结构开发模型

# 架构设计



# 架构设计

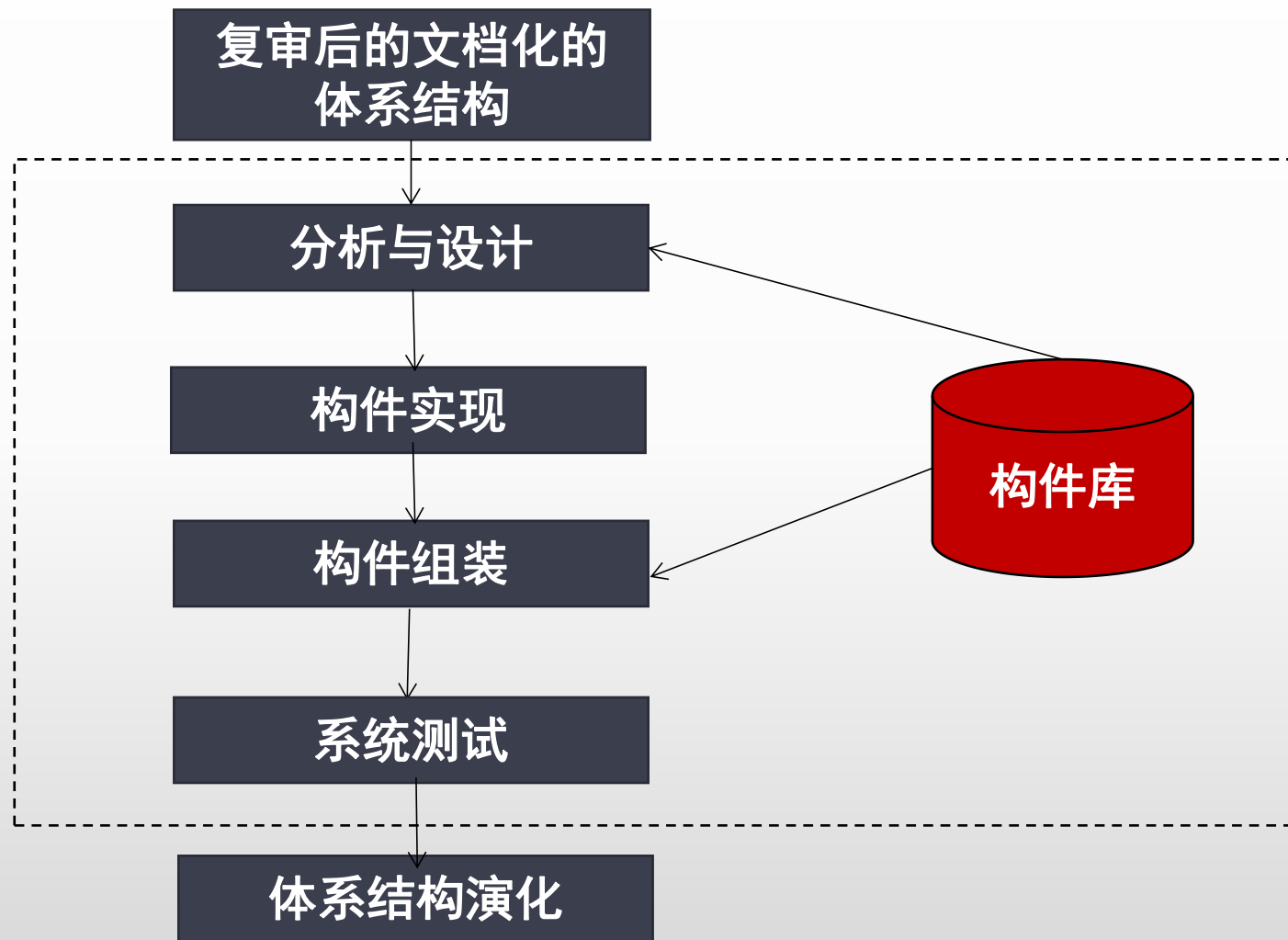


体系结构设计过程

# 架构设计

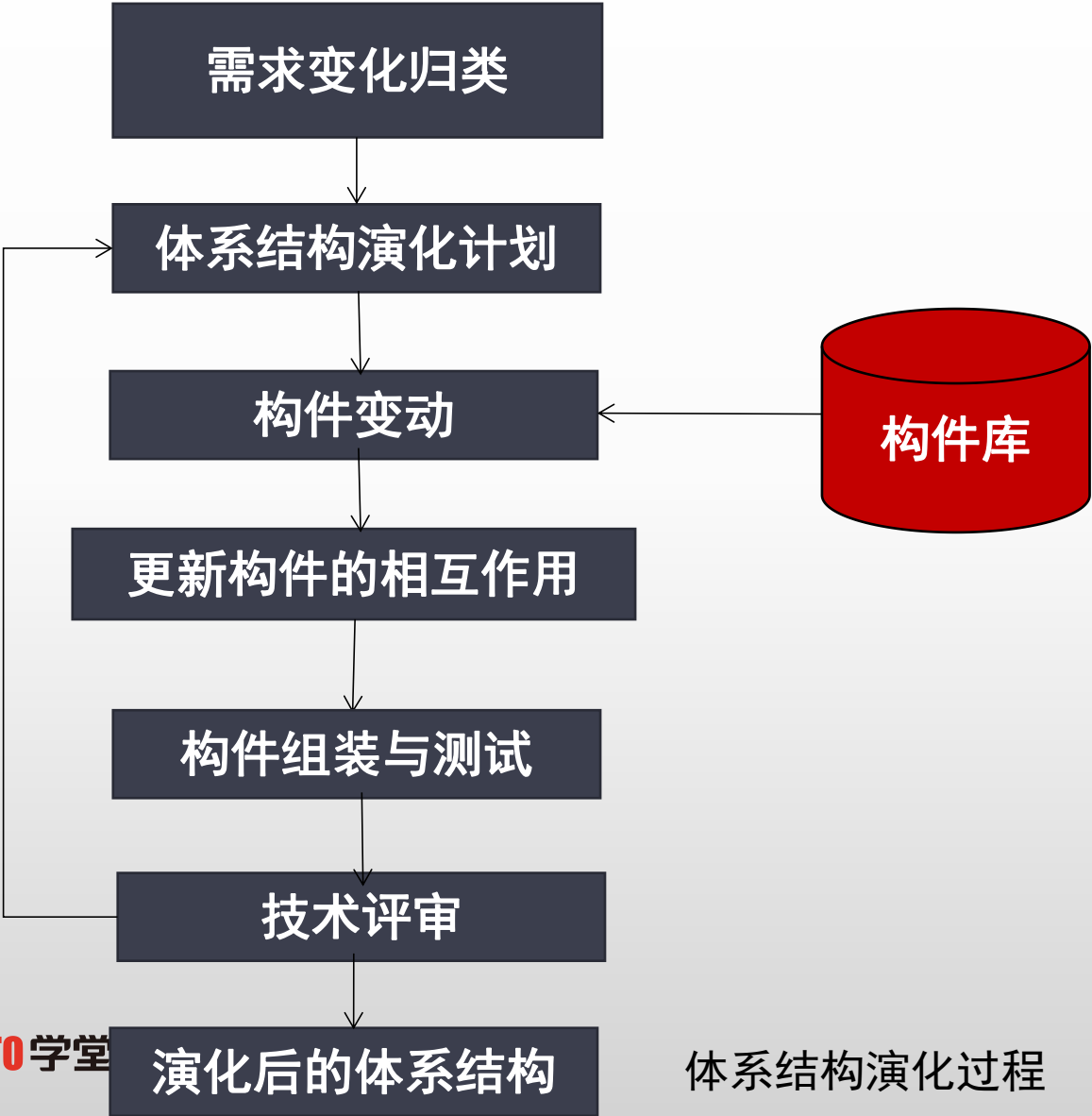
- 体系结构文档化过程的主要输出结果是**体系结构规格说明**和测试体系结构需求的**质量设计说明书**这两个文档。
- 体系结构复审是一个迭代过程。目的是标识出**潜在的风险**尽早发现体系结构设计中的**缺陷**和**错误**。从这个方面来说，在一个主版本的软件体系结构分析之后，要安排一次由外部人员（**用户代表**和**领域专家**）参加的复审。

# 架构设计

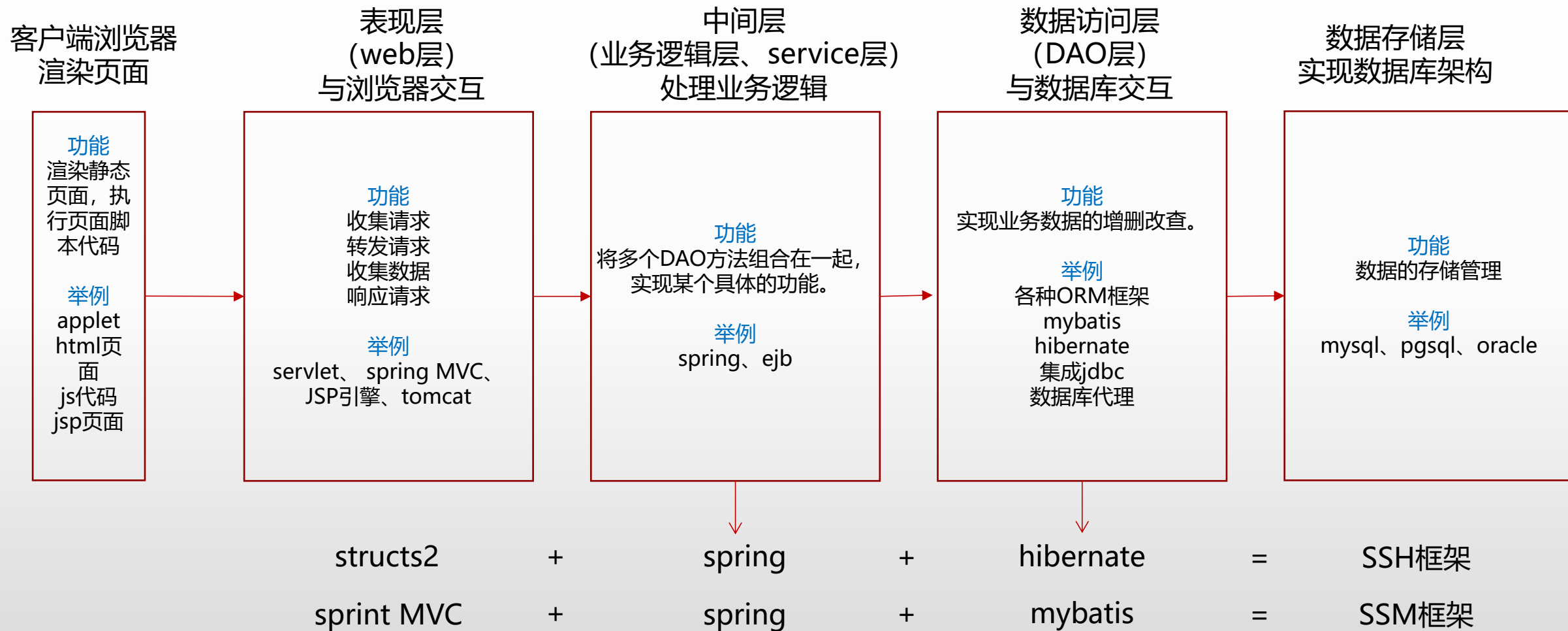


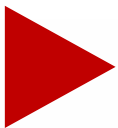


# 架构设计



# Web系统设计



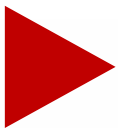


# Web系统设计

前端技术	描述
HTML/CSS/JS	前端页面包含的元素，被浏览器渲染成用户看到的五光十色的网页内容。
JSP页面	其实是HTML页面。由表现层JSP引擎使用JSP模板文件+后端数据组装而成。
Ajax	一种基于js的创建交互式网页应用的网页开发技术，无需重新加载整个网页的情况下，能够异步更新部分网页的技术。
Jquery	基于JS语言，集合Ajax技术开发出来的JS库，封装JS和Ajax的功能，提供函数接口，大大简化了Ajax，JS的操作。
Applet	采用Java编程语言编写的小应用程序，该程序可以包含在 HTML中，含有Applet的HTML网页代码中部带有<applet> 和</applet>这样一对标记，当支持Java的网络浏览器遇到这对标记时，就将下载相应的小应用程序代码并在本地计算机上执行该Applet。

表现层技术	描述
struts2	基于MVC设计模式的Web应用框架，高危安全漏洞很多，已被淘汰。
springMVC	spring的一个子模块，基于MVC设计模式的Web应用框架。
servlet	基于servlet接口规范实现的类，运行在servlet容器中。当servlet容器接收到前端发来的请求后，就转发给响应的servlet类来处理。servlet类中可以编写业务代码，也可以运用MVC设计模式。
tomcat	既是Servlet容器（运行servlet类的对象），又是Web服务器（接收请求、处理请求（委托给servlet类）、响应请求）
JSP引擎	当业务代码从后端接收到数据后，JSP引擎负责将JSP模板和后端数据结合处理成JSP页面（最终通过应用服务器发送给前端）
JSP模板	尚未融合后端数据的HTML文件，所有元素的值都是空的，等待JSP引擎将后端数据填入。成为最终的JSP页面/HTML代码。

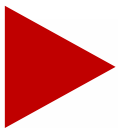




# Web系统设计

应用层技术	描述
docker	开源的应用容器引擎，让开发者打包他们的应用以及依赖包到一个轻量级、可移植的容器中，然后发布到任何流行的Linux 机器上，也可以实现虚拟化。完全使用沙箱机制，性能开销极低。
k8s	Kubernetes是一个开源的，用于管理云平台中多个主机上的容器化的应用，Kubernetes的目标是让部署容器化的应用简单并且高效，Kubernetes提供了应用部署，规划，更新，维护的一种机制。
spring	开放源代码的J2EE应用程序框架，是针对bean的生命周期进行管理的轻量级容器，可以单独应用于构筑应用程序，也可以和Struts、Webwork、Tapestry等众多Web框架组合使用，并且可以与 Swing等桌面应用程序AP组合。
EJB	EJB （EJB容器）是用于开发和部署多层结构的、分布式的、面向对象的 Java 应用系统的跨平台的构件体系结构。已被淘汰。

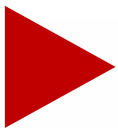
数据访问/持久层技术	描述
hibernate	ORM框架，sql语句高度封装，自动生成，加快开发速度。但无法优化。
mybatis	ORM框架，半自动化，sql手工配置，自行管理对象和关系模式的映射关系，可以优化，比较灵活。
jdbc	java语言中用来规范客户端程序如何来访问数据库的应用程序接口，提供了诸如查询和更新数据库中数据的方法。 jdbc
EJB	



# Web系统设计

数据库	技术名词	描述
SQL 大部分可支持事务	MySQL	一种开源的关系型数据库管理系统，关系数据库将数据保存在不同的表中，体积小、速度快、总体拥有成本低，尤其是开放源码这一特点，一般中小型和大型网站的开发都选择 MySQL 作为网站数据库。
	PostgreSQL	简称pgsql，开源的对象-关系型数据库管理系统。与mysql互有优劣区别。
	Oracle	甲骨文公司的一款关系数据库管理系统，非开源，商业化产品。
	SQL Server	Microsoft开发和推广的关系数据库管理系统。非开源，起源于windows服务器。
NoSQL 不支持事务	MongoDB	MongoDB 是一个基于分布式文件存储的数据库。支持的数据结构非常松散，是类似json的bson格式，可以存储比较复杂的数据类型。Mongo最大的特点是它支持的查询语言非常强大，其语法有点类似于面向对象的查询语言。
	Redis	开源的使用ANSI C语言编写、可基于内存亦可持久化的日志型key-value数据库。
	leveldb	google实现的高效KV数据库。在十亿级别的数据量下还有着非常高的性能。核心算法是LSM。
	HBase	可伸缩、分布式的大数据存储列式数据库，可以用在数据的实时与随机访问的场景下，拥有模块化与线性的可伸缩性，并且能够保证读写的严格一致性。

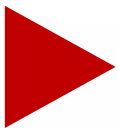
缓存	描述
memcached	分布式的高速缓存系统，缺乏认证和安全机制，只能当做缓存系统使用。多线程，协议简单，基于libevent的时间处理，不互相通信的分布式。



# Web系统设计

web服务器	描述
apache	快速，可靠，可通过简单的API扩展功能，将动态语言（python/php/perl等）解释器编译到服务器中。
nginx	轻量级高性能的web服务器，因它的稳定性、丰富的功能集、简单的配置文件、低资源消耗和并发能力强而闻名。
tomcat	既是Servlet容器（运行servlet类的对象），又是Web服务器（接收请求、处理请求（委托给servlet类）、响应请求）
Jetty	与tomcat相比，可以同时处理大量连接并且长时间保持这些连接。
IIS	在windows操作系统平台下的web服务器。

消息队列	描述
ActiveMQ	ActiveMQ 是Apache出品，流行的、能力强劲的开源消息总线。
RabbitMQ	流行的开源消息队列系统，用erlang语言开发。是AMQP（高级消息队列协议）的标准实现
Kafka	是一种高吞吐量、高性能的分布式发布订阅消息系统，支持消息分区、支持hadoop并行数据加载。
RocketMQ	阿里开源的消息中间件，纯Java开发，具有高吞吐量、高可用性、适合大规模分布式系统应用的特点。
Redis	自带发布-订阅机制，可用作消息转发构件。



# Web系统设计

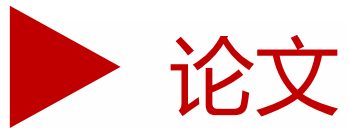
分布式协调构件	描述
ZooKeeper	为分布式应用提供一致性服务的软件，提供的功能包括配置维护、域名服务、分布式同步、组服务等。
etcd	一个高可用的键值存储系统，主要用于配置共享和服务发现。
Nacos	更易于帮助构建云与安生应用的动态服务发现、配置和服务管理平台，提供注册中心、配置中心和动态DNS三大功能。

其他技术	描述
DNS	因特网上作为域名和IP地址互相映射的一个分布式数据库，能够使用户更方便的访问互联网，而不用去记住能够被机器直接读取的IP数串。通过主机名，最终得到该主机对应的IP地址的过程叫做域名解析（或主机名解析）。
CDN	Content Delivery Network，即内容分发网络。CDN是构建在现有网络基础之上的智能虚拟网络，依靠部署在各地的边缘服务器，通过中心平台的负载均衡、内容分发、调度等功能模块，使用户就近获取所需内容，降低网络拥塞，提高用户访问响应速度和命中率。
keepalive	用来管理并监控LVS集群系统中各个服务节点的状态，后来又加入了可以实现高可用的VRRP功能。因此，Keepalived除了能够管理LVS软件外，还可以和VIP漂移技术搭配，作为其他服务（例如：LVS、Nginx、MySQL等）的高可用解决方案软件。

# ▶ 案例分析

## 2022年系统架构设计师案例方向

案例一	案例二	案例三	案例四	案例五
1、经典架构风格对比 2、架构评估/非功能性需求	1、结构化建模 •补充DFD图、ER图 2、面向对象建模 •用例图、状态图、活动图	嵌入式系统设计	Mysql+分布式缓存技术Redis	Web架构设计 • 层次式架构



论文

## 2022年系统架构设计师论文方向

- “三高” ——高并发、高可用、高性能
- 分布式架构技术
- 系统分析与设计方法

# 51CTO 学堂

技术成就梦想