系统架构设计师

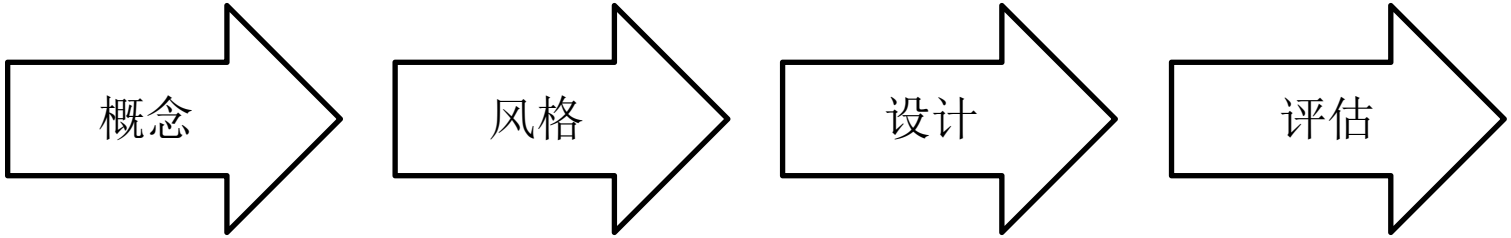
DESIGNER:王川林

直播课2

第三章 架构设计

课程内容

- 软件架构的概念 ★★★
- 软件架构风格 ★★★★★
- 架构描述语音ADL ★★★
- 特定领域软件架构 ★★★
- 基于架构的软件开发方法 ★★★★
- 软件质量属性 ★★★★★
- 软考架构评估 ★★★★★
- 软件产品线 ★★★
- 中间件技术 ★★★
- 典型应用架构 ★★★


概念
风格
设计
评估

需求分析　架构　软件设计

鸿沟

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。
- 软件架构的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质量。
- 软件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。

- 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
- 架构风格定义了用于描述系统的术语表和一组指导构建系统的规则

- 数据流风格：批处理序列、管道—过滤器
- 调用/返回风格：主程序/子程序、面向对象、层次结构
- 独立构建风格：进程通信、事件驱动系统（隐式调用）
- 虚拟机风格：解释器、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

软件架构风格—数据流风格

批处理序列

构件为一系列固定顺序的计算单元，构件之间只通过数据传递交互，每个处理步骤是一个独立的程序，每一步必须在其前一步结束后才能开始，数据必须是完整的，以整体的方式传递。

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,这个过程通常是通过对输入数据流的变换或计算来完成的,包括通过计算和增加信息以丰富数据、通过浓缩和删除以精简数据，通过改变记录方式以转化数据和递增地转化数据等.这里的构件称为过滤器,连接件就是数据传输的管道，将一个过滤器的输出传到另一个过滤器的输入。

早期翻译器就是采用的这种架构.要一步一步处理的,均可考虑采用此架构风格

软件架构风格—调用/返回风格

主程序/子程序

面向对象

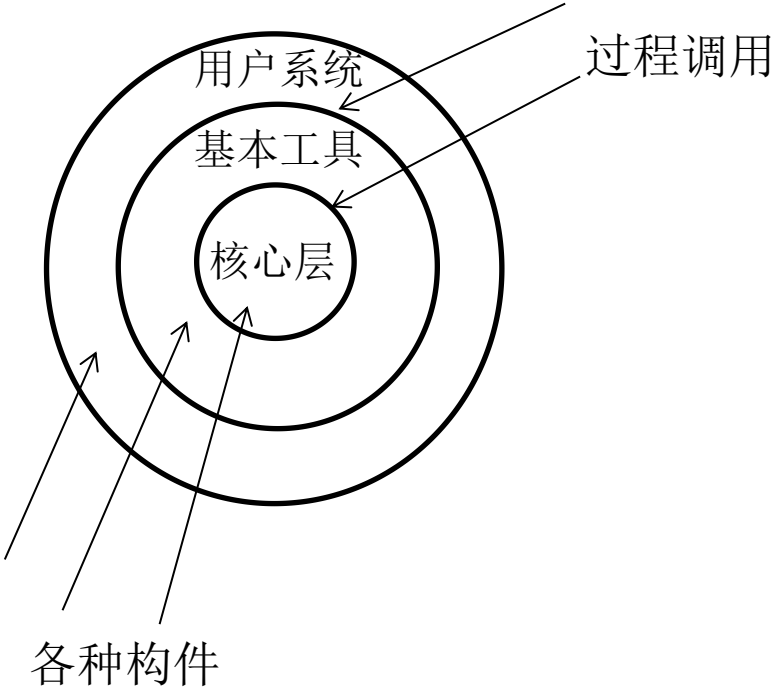
层次结构

单线程控制，把问题划分为若干个处理步骤，构件即为主程序和子程序，子程序通常可合成为模板。过程调用作为交互机制，即充当连接件的角色。通用关系具有层次性，其语义逻辑表现为主程序的正确性取决于它调用的子程序的正确性

构件是对象，对象是抽象数据类型的实例。在抽象数据类型中，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。连接件即是对象间交互的方式，对象是通过函数和过程的调用来交互的

构件组织成一个层次结构，连接件通过决定层间如何交互的协议来定义。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。通过层次结构，可以将大的问题分解为若干个渐进的小问题逐步解决，可以隐藏问题的复杂度。修改某一层，最多影响其相邻的两层（通常只能影响上层）

软件架构风格—调用/返回风格（层次结构）



优点：

1、这种风格支持基于可增加抽象层的设计，允许将一个复杂问题分解成一个增量步骤序列实现。

2、不同的层次处于不同的抽象级别：

越靠近底层，抽象级别越高；

越靠近顶层，抽象级别越低；

3、由于每一层最多只影响两层，同时只要给相邻层提供相同的接口，允许每层用不同的方法实现，同样为软件复用提供了强大的支持

缺点：

1、并不是每个系统都可以很容易的划分为分层的模式。

2、很难找到一个合适的、正确的层次抽象方法

软件架构风格—独立构件风格

进程通信

事件驱动系统（隐式调用）

构件是独立的过程，连接件是消息传递，构件通常是命名过程，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

构件不直接调用一个过程，而是触发或广播一个或多个事件。构件中的过程在一个或多个事件中注册,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。一个事件的触发就导致了另一个模块中的过程调用。这种风格中的构件是匿名的过程,它们之间交互的连接件往往是以过程之间的隐式调用来实现的。主要优点是为软件复用提供了强大的支持,为构件的维护和演化带来了方便;其缺点是构件放弃对系统计算的控制

解释器

基于规则的系统

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的进度的数据结构，具有解释器风格的软件中含有一个虚拟机，可以仿真硬件的执行过程和一些关键应用，其缺点是执行效率比较低

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

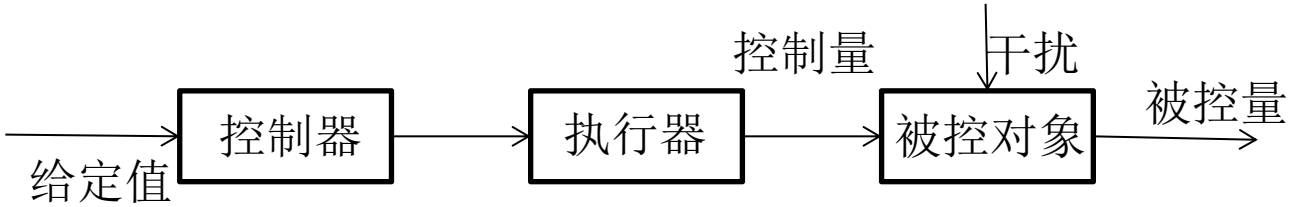
软件架构风格—仓库风格（以数据为中心的风格

- 数据库系统
  - 构件主要有两大类，一类是中央共享数据源，保存当前系统的数据状态；另一类是多个独立处理单元，处理单元对数据元素进行操作
- 黑板系统
  - 包括知识源、黑板和控制三部分。知识源包括若干独立计算的不同单元,提供解决向题的知识。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于解决问题设有确定性算法的软件中(信号处理、问题规划和编译器优化等）
- 超文本系统
  - 构件以网状链接方式相互连接，用户可以在构件之间进行按照人类的联想思维方式任意跳转到相关构件，超文本是一种非线性的网状信息组织方法，它以结点为基本单位，链作为结点之间的联想式关联。超文本系统通常应用在互联网区域

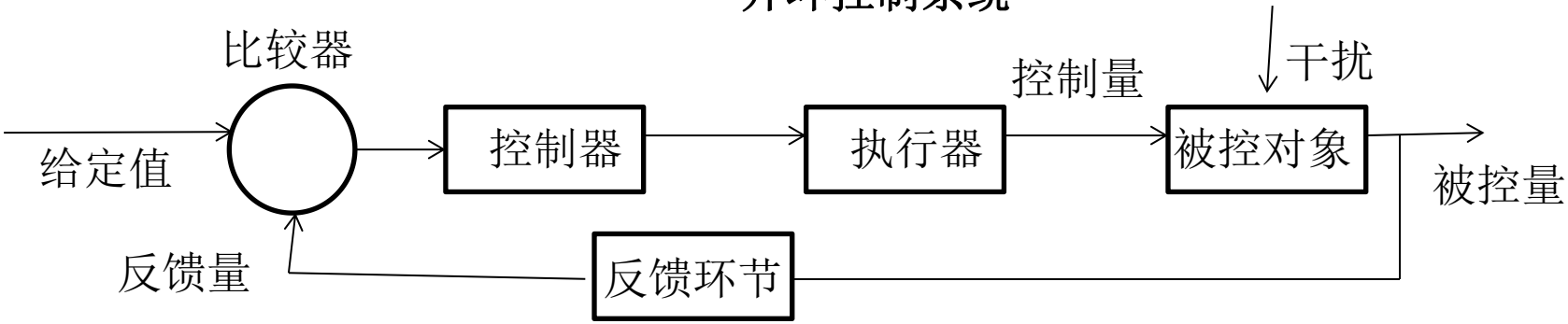
现代集成编译环境一般采用这种架构风格

软件架构风格—闭环控制架构（过程控制）

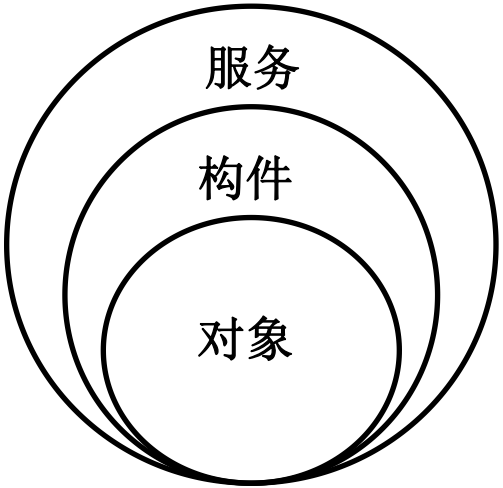
当软件被用来操作一个物理系统时，软件与硬件之间可以粗略地表示为一个反馈循环，这个反馈循环通过接受一定的输入，确定一系列的输出，最终使环境达到一个新的状态。适合于嵌入式系统，涉及连续的动作与状态。



开环控制系统



SOA



松散耦合

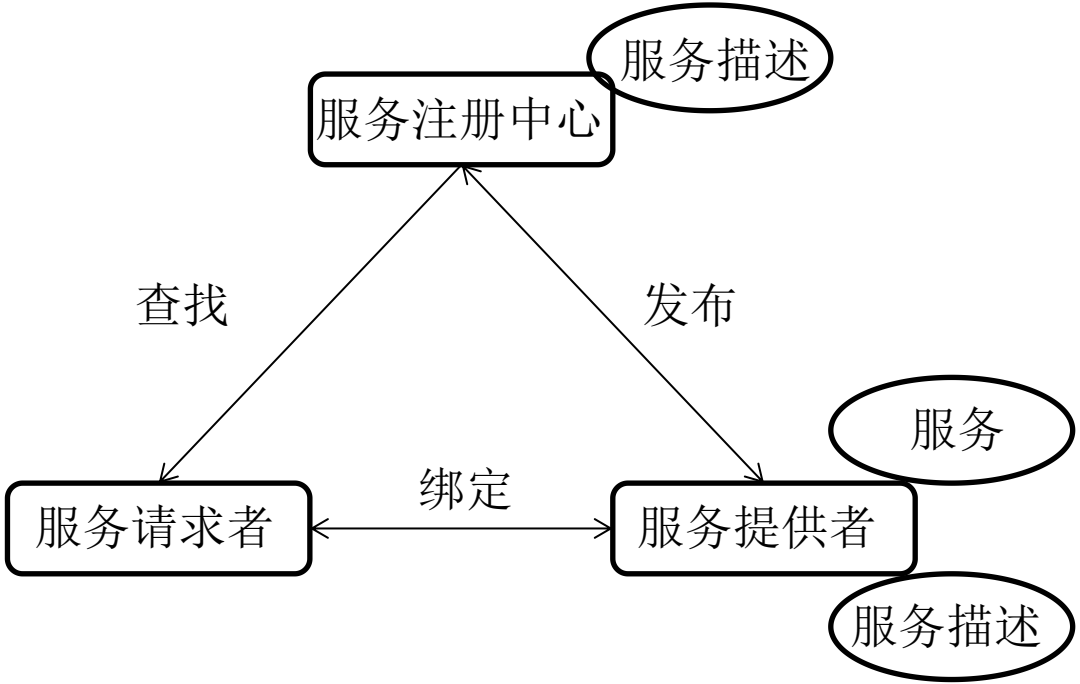
粗粒度

标准化接口

- 服务构件粗粒度，传统构件细粒度居多
- 服务构件的接口是标准的，主要是WSDL接口，传统构件常以具体API形式出现
- 服务构件的实现与语言无关，传统构件绑定各种特定语言
- 服务构件可以通过构件容器提供QoS的服务，传统构件完全由程序代码直接控制

- HTTP+XML进行基于Web通信的技术
- 简单性，缺少严格配置文件
- 只支持几个操作（POST、GET、PUT、DELETE）
- 强调信息本身，称为资源

- 网络上的所有事物都被抽象为资源
- 每个资源对应一个唯一的资源标识
- 通过通用的连接器接口对资源进行操作
- 对资源的各种操作不会改变资源标识
- 所有的操作都是无状态的



底层传输层

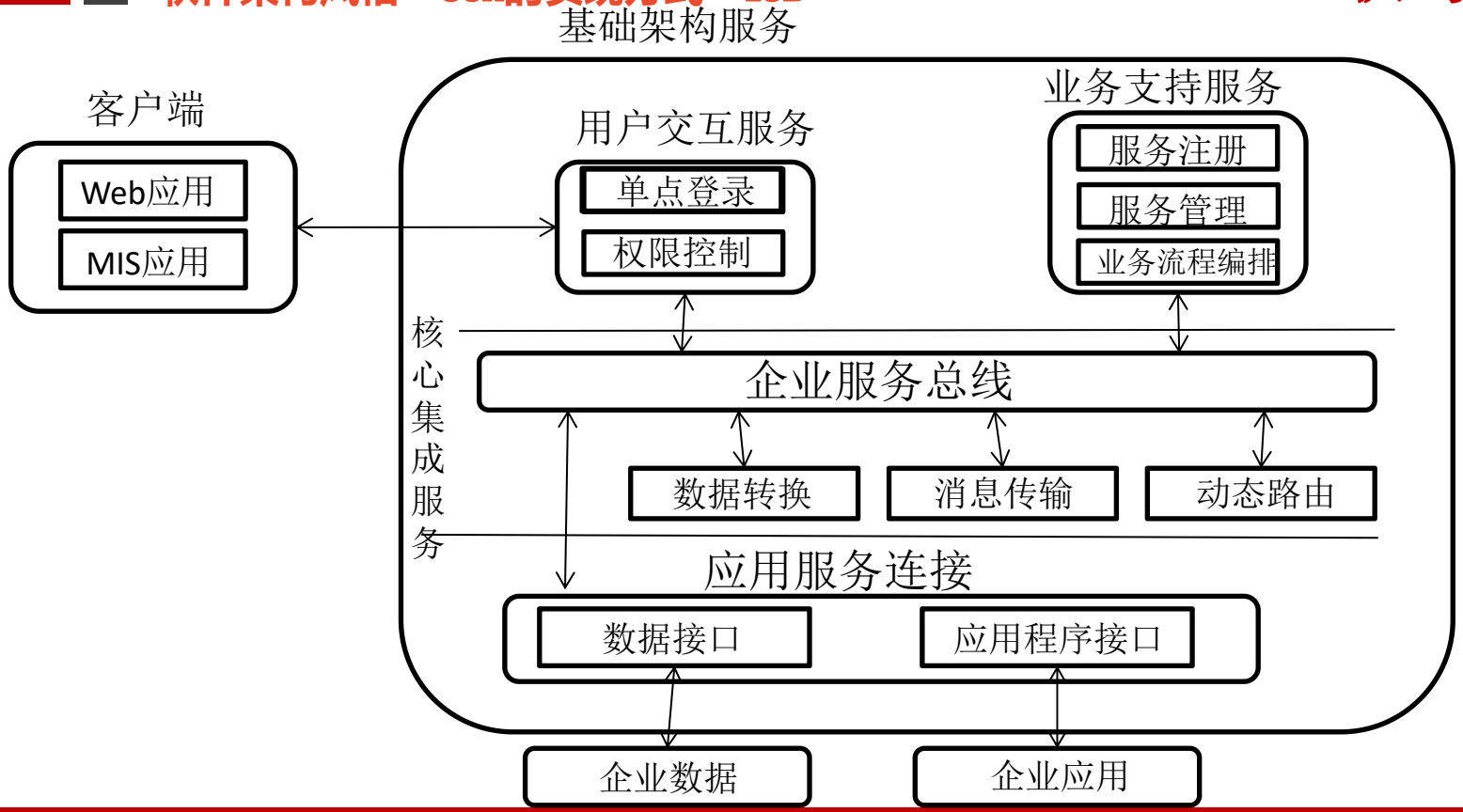
服务通信协议层

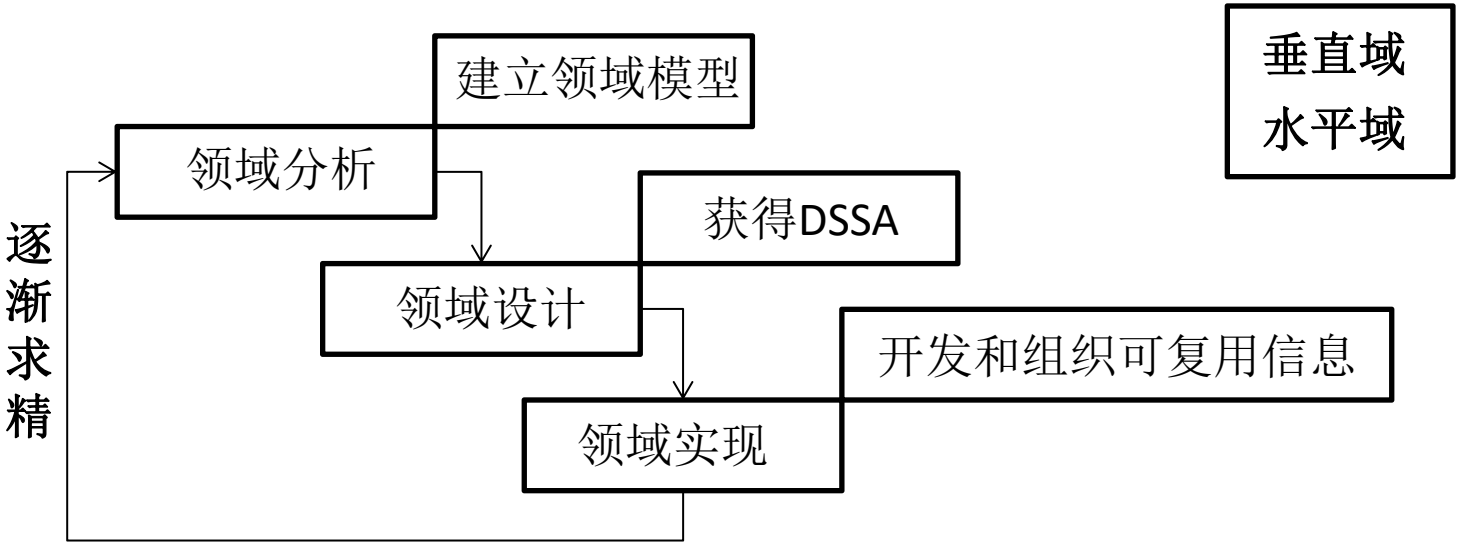
服务描述层

服务层

业务流程层

服务注册层


基础架构服务
客户端
Web应用
MIS应用
用户交互服务
单点登录
权限控制
业务支持服务
服务注册
服务管理
业务流程编排
核心集成服务
企业服务总线
数据转换
消息传输
动态路由
应用服务连接
数据接口
应用程序接口
企业数据
企业应用


建立领域模型
领域分析
获得DSSA
领域设计
开发和组织可复用信息
领域实现
逐渐求精
垂直域
水平域

- ABSD方法是架构驱动，即强调由业务、质量和功能需求的组合驱动架构设计
- 使用ABSD方法，设计活动可以从项目总体功能框架明确就开始，这意味着需求获取和分析还没有完成（甚至远远没有完成），就开始了软件设计
- ABSD方法有三个基础，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藕合技术；第二个基础是通过选择架构风格来实现质量和业务需求：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
- ABSD方法是递归的，且迭代的每一个步骤都是清晰地定义的。因此，不管设计是否完成，架构总是清晰地，这有助于降低架构设计的随意性

1、性能

性能（performance）是指系统的响应能力，即要经过多长时间才能对某个事件做出响应，或者在某段时间内系统所能处理的事件的个数。

代表参数：响应时间、吞吐量　　设计策略：优先级队列、资源调度

2、可靠性

可靠性（reliability）是软件系统在应用或系统错误面前，在意外或错误使用的情况下维持系统的功能特性的基本能力

代表参数：MTTF、MTBF　　设计策略：冗余、心跳线

3、可用性

可用性（availability）是系统能够正常运行的时间比例。经常用两次故障直接的时间长度或在出现故障时系统能够恢复正常的速度来表示。

代表参数：故障间隔时间　　设计策略：冗余、心跳线

4、安全性

安全性（security）是指系统在向合法用户提供服务的同时能够阻止非授权用户使用的企图或拒绝服务的能力。安全性又可划分为机密性、完整性、不可否认性及可控性等特性。

设计策略：追踪审计、信息隐藏

5、可修改性

可修改性（modifiability）是指能够快速地以较高的性能价格比对系统进行变更的能力。通常以某些具体的变更为基准，通过考察这些变更的代价衡量可修改性。

6、功能性

功能性（functionality）是系统所能完成所期望的工作的能力。一项任务的完成需要系统中许多或大多数构件的相互协作。

7、可变性

可变性（changeability）是指系统结构经扩充或变更而成为新体系结构的能力。这种新体系结构应该符合预先定义的规则，在某些具体方面不同于原有的体系结构。当要将某个体系结构作为一系列产品（例如，软件产品线）的基础时，可变性是很重要的。

8、互操作性

作为系统组成部分的软件不是独立存在的，经常与其他系统的自身环境相互作用。为了支持互操作性（interoperation），软件体系结构必须为外部可视的功能特性和数据结构提供精心设计的软件入口。程序和用其他编程语言编写的软件系统的交互作用就是互操作性的问题，这种互操作性也影响应用的软件体系结构。

（1）用户提交搜索请求后，系统必须在1秒内显示结果；（性能）

（2）用户信息数据库授权必须保证99.9%可用

（3）系统由MySQL数据库升级为Oracle数据库，必须在1人月内完成；（可修改性）

（4）主服务器出现严重问题无法提供服务时，备用系统10分钟内能接替其工作；（可用性）

（5）需要在3人周内为系统添加一种新的支付方式—支付宝；（可修改性）

（6）视频点播时，超清模式必须保证画面具有1280*720的分辨率；（性能）

（7）主站点断电后，需要在3秒内将访问请求重定向到备用站点；（可用性）

风险点：系统架构风险是指架构设计中潜在的、存在问题的架构决策所带来的的隐患

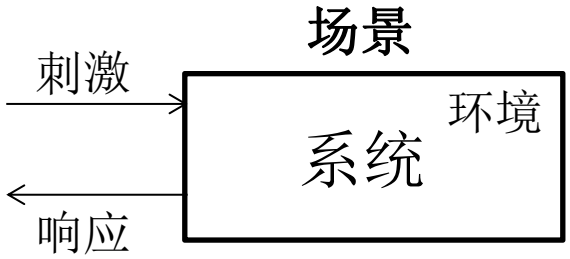
敏感点是指为了实现某种特定的质量属性，一个或多个构件所具有的特性

权衡点是影响多个质量属性的特性，是多个质量属性的敏感点

基于调查问卷（检查表）的方式

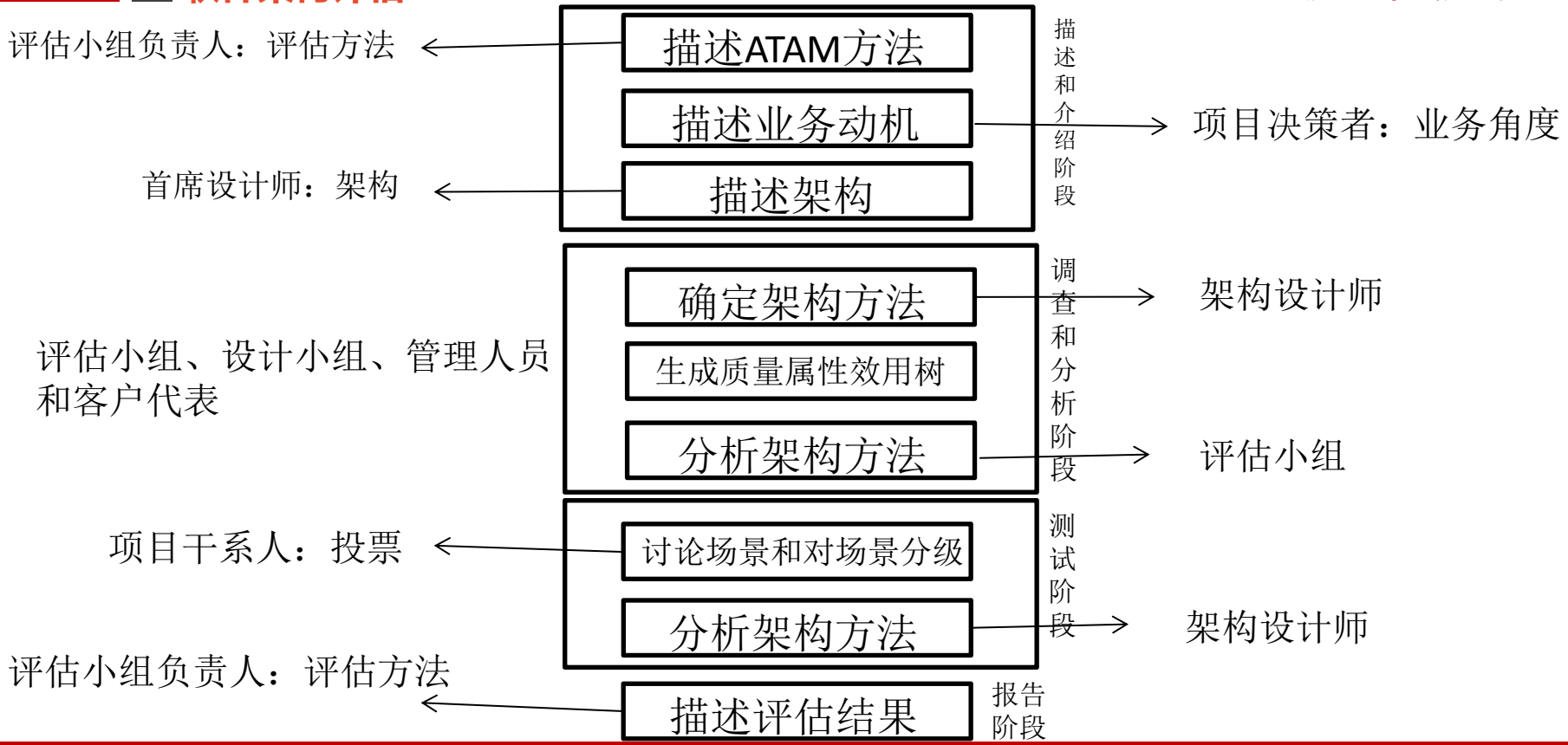
基于度量的方式

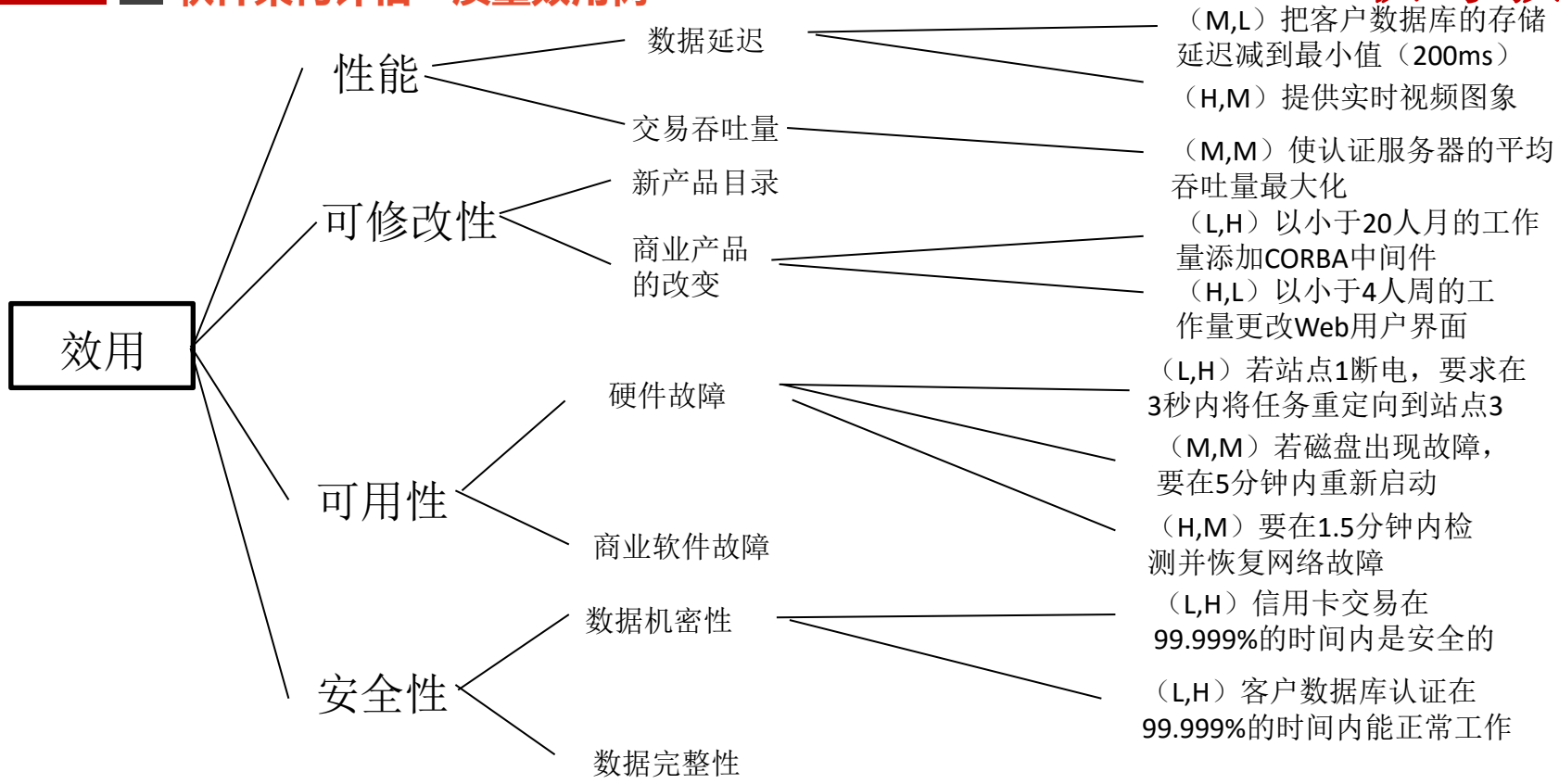
基于场景的方式

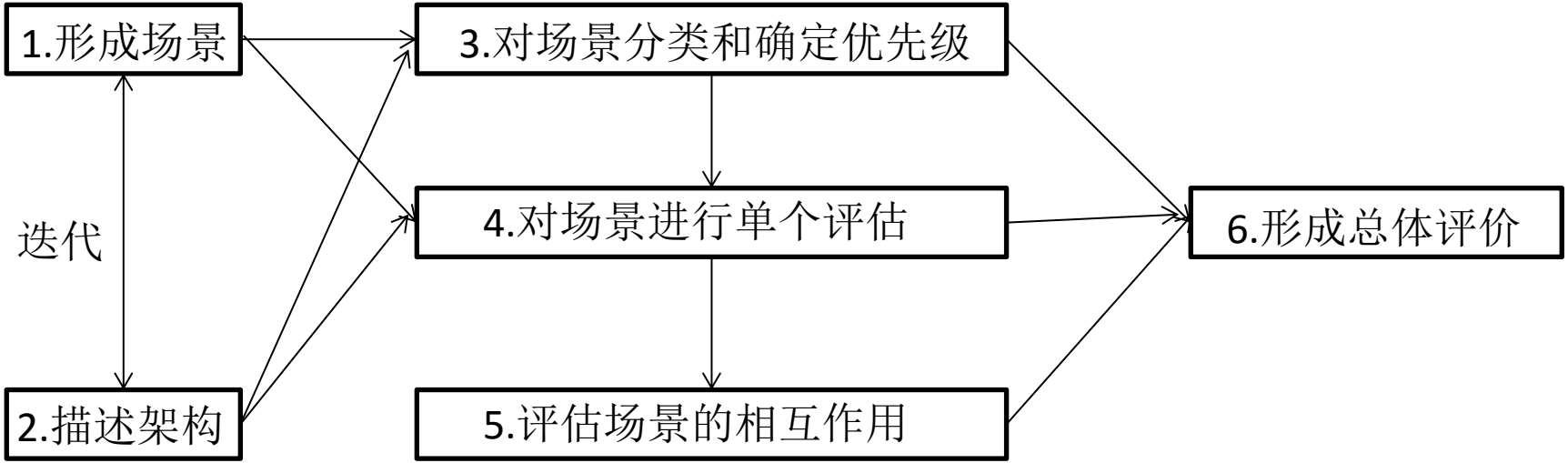


- 确定应用领域的功能和软件架构的结构之间的映射
- 设计用于体现待评估质量属性的场景
- 分析软件架构对场景的支持程度

- 架构权衡分析法（ATAM）
- 软件架构分析法（SAAM）
- 成本效益分析法（CBAM）


评估小组负责人：评估方法
描述ATAM方法
描述业务动机
首席设计师：架构
描述架构
描述和介绍阶段
项目决策者：业务角度
确定架构方法
生成质量属性效用树
分析架构方法
调查和分析阶段
架构设计师
评估小组
评估小组、设计小组、管理人员和客户代表
项目干系人：投票
讨论场景和对场景分级
分析架构方法
测试阶段
架构设计师
评估小组负责人：评估方法
描述评估结果
报告阶段


效用
性能
数据延迟
交易吞吐量
可修改性
新产品目录
商业产品的改变
可用性
硬件故障
商业软件故障
安全性
数据机密性
数据完整性
（M,L）把客户数据库的存储延迟减到最小值（200ms）
（H,M）提供实时视频图象
（M,M）使认证服务器的平均吞吐量最大化
（L,H）以小于20人月的工作量添加CORBA中间件
（H,L）以小于4人周的工作量更改Web用户界面
（L,H）若站点1断电，要求在3秒内将任务重定向到站点3
（M,M）若磁盘出现故障，要在5分钟内重新启动
（H,M）要在1.5分钟内检测并恢复网络故障
（L,H）信用卡交易在99.999%的时间内是安全的
（L,H）客户数据库认证在99.999%的时间内能正常工作

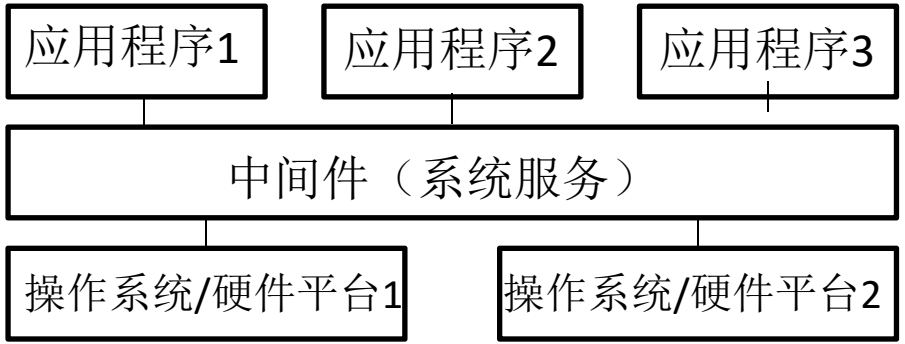

1.形成场景
3.对场景分类和确定优先级
迭代
4.对场景进行单个评估
6.形成总体评价
2.描述架构
5.评估场景的相互作用

- 整理场景
- 对场景进行求精
- 确定场景的优先级
- 分配效用
- 形成“策略-场景-响应级别”的对应关系
- 确定期望的质量属性响应级别的效用
- 计算各架构策略的总收益
- 根据受成本限制影响的投资报酬率选择架构策略

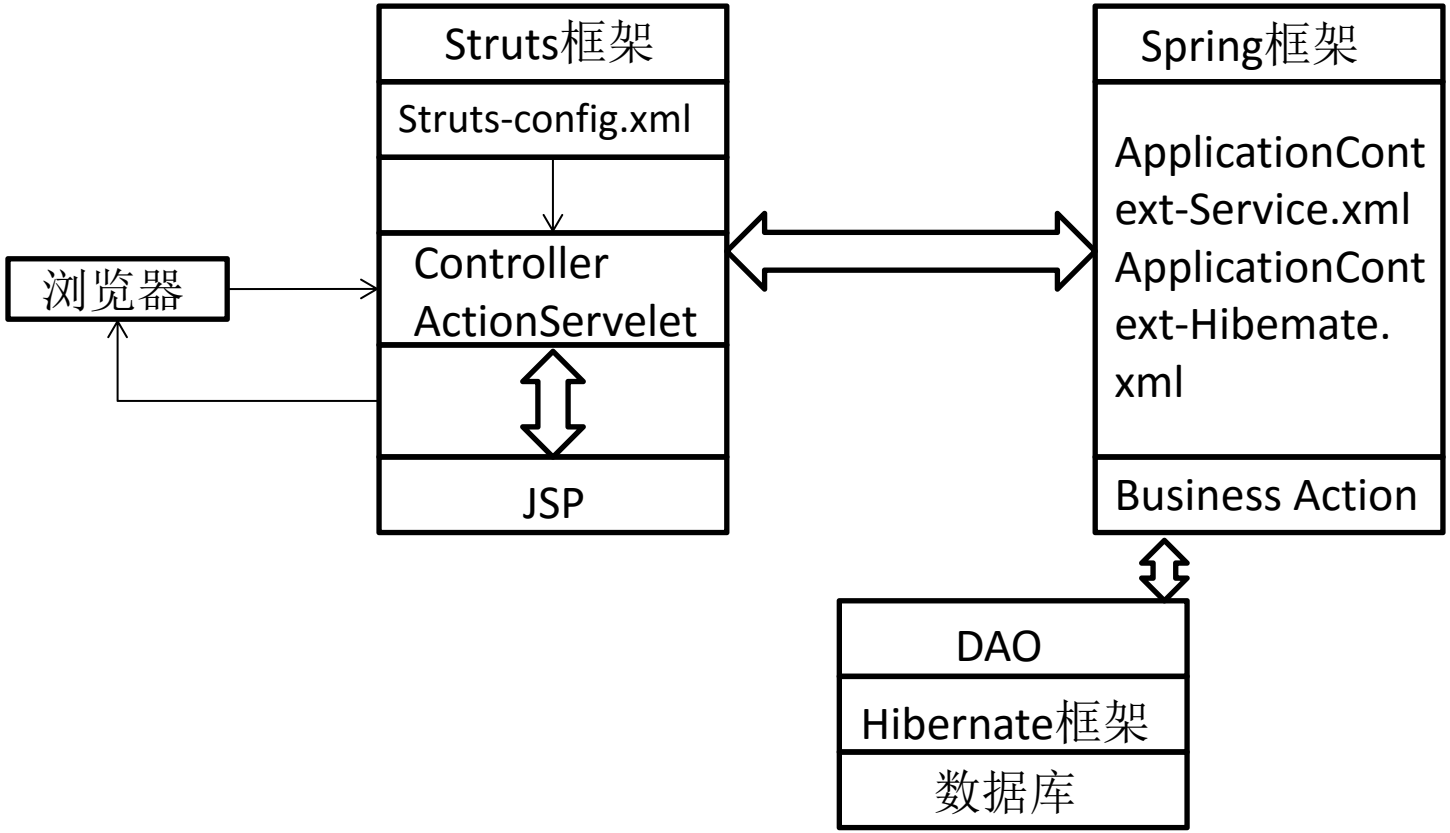
	演化方式	革命方式
基于现有产品	基于现有产品架构设计产品线的架构，经演化现有构件，开发产品线构件	核心资源的开发基于现有产品集的需求和可预测的、将来需求的超集
全新产品线	产品线核心资源随产品新成员的需求而演化	开发满足所有预期产品线成员的需求的核心资源

- 将现有产品演化为产品线
- 用软件产品线替代现有产品集
- 全新软件产品线的演化
- 全新软件产品线的开发

中间件是一种独立的系统软件或服务程序，可以帮助分布式应用软件在不同的技术之间共享资源



- 负责客户机与服务器之间的连接和通信，以及客户机与应用层之间的高效率通信机制
- 提供应用层不同服务之间的互操作机制，以及应用层与数据库之间的连接和控制机制
- 提供多层构架的应用开发和运行的平台，以及应用开发框架，支持模块化的应用开发
- 屏蔽硬件、操作系统、网络和数据库的差异
- 提供应用的负载均衡和高可用性、安全机制与管理功能，以及交易管理机制，保证交易的一致性
- 提供一组通用的服务去执行不同的功能，避免重复的工作和使应用之间可以协作


Struts框架
Struts-config.xml
Controller
ActionServelet
JSP
浏览器
Spring框架
ApplicationCont
ext-Service.xml
ApplicationCont
ext-Hibemate.
xml
Business Action
DAO
Hibernate框架
数据库

Struts是一个基于J2EE平台的MVC框架，主要采用Serviet和JSP技术来实现。在Struts中，M由实现业务逻辑的JavaBean构成，C由ActionServiet和Action来实现，V由一组JSP文件构成

Spring通过RMI或Web Service远程访问业务逻辑，允许自由选择和组装各部分功能，还提供和其他软件集成的接口。Spring本身是个容器，管理构件的生命周期、构件的组态。依赖注入等，并可以控制构件在创建时以原型或单例模式来创建

Hibernate是一个对象关系映射框架，提供了Java对象到数据库表之间的直接映射，它对JDBC进行了非常轻量级的对象封装，使得Java程序员可以使用对象编程思维来操作数据库。在Hibernate中，ORM机制的核心是一个XML文件，该文件描述了数据库模式是怎么与一组Java类绑定在一起的

Model（模型）是应用程序中用于处理应用程序数据逻辑的部分。通常模型对象负责在数据库中存取数据
View（视图）是应用程序中处理数据显示的部分。通常视图是依据模型数据创建的。
Controller（控制器）是应用程序中处理用户交互的部分。通常控制器负责从视图读取数据，控制用户输入，并向模型发送数据

J2EE体系结构中：
视图（View）：JSP
控制（Controller）：Servlet
模型（Model）：Entity Bean、Session Bean

历年真题

阅读以下关于软件架构评估的叙述，在答题纸上回答问题 1 和问题 2.

【说明】

某单位为了建设健全的公路桥梁养护管理档案，拟开发一套公路桥梁在线管理系统。在 系统的需求分析与架构设计阶段，用户提出的需求、质量属性描述和架构特性如 下:

(a) 系统用户分为高级管理员、数据管理员和数据维护员等三类；

(b) 系统应该具备完善的安全防护措施，能够对黑客的攻击行为进行检测与防御；

(c) 正常负载情况下，系统必须在 0.5 秒内对用户的查询请求进行响应；

(d) 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；

(e) 系统的用户名不能为中文，要求必须以字母开头，长度不少于 5 个字符；

(f) 更改系统加密的级别将对安全性和性能产生影响；

(g) 网络失效后，系统需要在 10 秒内发现错误并启用备用系统；

(h) 查询过程中涉及到的桥梁与公路的实时状态视频传输必须保证画面具有 1024*768 的分辨率， 40 帧 /秒的速率；

(i) 在系统升级时，必须保证在 10 人月内可添加一个新的消息处理中间件；

(j) 系统主站点断电后，必须在 3 秒内将请求重定向到备用站点；

(k) 如果每秒钟用户查询请求的数量是 10 个，处理单个请求的时间为 30 毫秒，则系统应保证在 1 秒内完成用户的查询请求；

(l) 对桥梁信息数据库的所有操作都必须进行完整记录；

(m) 更改系统的 Web 界面接口必须在 4 人周内完成；

(n) 如果"养护报告生成"业务逻辑的描述尚未达成共识，可能导致部分业务功能 模块规则的矛盾，影响系统的可修改性

(O) 系统必须提供远程调试接口，并支持系统的远程调试。

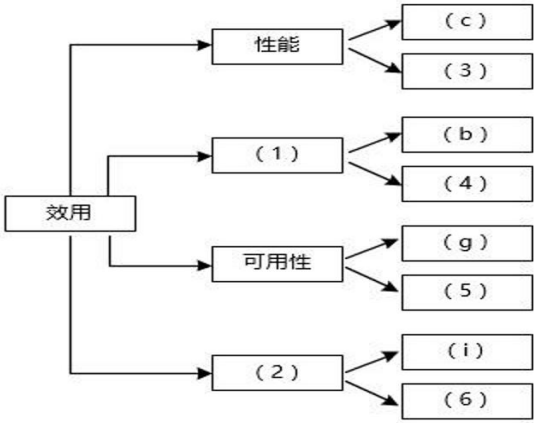
在对系统需求，质量属性描述和架构特性进行分析的基础上，系统的架构师给出了三个候选的架构设计方案，公司目前正在组织系统开发的相关人员对系统架构进行评估。

【问题 1】 (12 分)

在架构评估过程中，质量属性效用树 (utility tree) 是对系统质量属性进行识别和优先级

历年真题

排序的重要工具。请给出合适的质量属性，填入图 1-1 中 (1)、(2) 空白处；并选择题干描述的 (a)~(o)，填入(3) ~(6) 空白处，完成该系统的效用树。



(1) 安全性

(2) 可修改性

(3) (h)

(4) (l)

(5) (j)

(6) (m)

历年真题

【问题 2】(13 分)

在架构评估过程中，需要正确识别系统的架构风险、敏感点和权衡点，并进行合理的架 构决策。请用 300 字以内的文字给出系统架构风险、敏感点和权衡点的定义，并从题干(a)~(o) 中分别选出 1 个对系统架构风险、敏感点和权衡点最为恰当的描述。

系统架构风险是指架构设计中潜在的、存在问题的架构决策所带来的隐患。 敏感点是指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。 权衡点是影响多个质量属性的特性，是多个质量属性的敏感点。风险点: （n）敏感点:

（d） 权衡点: （f）

历年真题

●软件架构风格是描述某一特定应用领域中系统组织方式的（1）模式。

(1) A. 常用　　B. 惯用　　C. 经典　　D. 复用

答案B，解析：架构章节的定义，记不得的同学，看PPT。

●以下不属于软件架构评评估方法的是（2）。

(2) A. ATAM　　B. ABSD　　C. SAAM　　D. CBAM

答案B，解析：ABSD（基于架构的软件设计），是一种软件开发方法，非评估方法。

●以下不属于特定领域软件架构（DSSA）建立过程中涉及的技术角色的是（3）。

(3) A. 领域架构师　　B. 应用工程师　　C. 系统分析师　　D. 操作员

答案C，解析：DSSA建立过程：领域开发环境涉及领域架构师； 领域特定的应用开发环境涉及应用工程师，应用执行环境涉及操作员。

历年真题

● 某公司拟开发一个 VIP 管理系统，系统需要根据不同商场活动，不定期更新 VIP 会员的审核标准和 VIP 折扣系统。针对上述需求，采用（49）架构风格最为合适。

(49)A.规则系统　　B.过程控制　　C.分层　　D.管道-过滤器

答案A，**解析**:审核标准、折扣都是属于自定义规则，故选规则系统。

某公司拟开发一个新闻系统，该系统可根据用户的注册兴趣，向用户推送其感兴趣的新内容，该系统应该采用（50）架构风格最为合适。

(50)A.事件驱动系统　　B.主程序-子程序　　C.黑板　　D.管道-过滤器

答案A，**解析**：推送内容，不是直接联系属于隐式调用，故选事件驱动系统。

● 某公司拟开发一个扫地机器人。机器人的控制者首先定义清洁流程和流程中任务之间的关系，机器人接受任务后，需要响应外界环境中触发的一些突发事件，根据自身状态进行动态调整，最终自动完成任务。针对上述需求，该机器人应该采用（51）架构风格最为合适。

(51)A. 面向对象　　B. 主程序-子程序　　C. 闭环　　D. 管道-过滤器

答案C，**解析**：本题前段自定义可用虚拟机风格，后段带反馈属于闭环风格。

● 某企业内部现有的主要业务功能已封装成为 Web 服务。为了拓展业务范围，需要将现有的业务功能进行多种组合，形成新的业务功能。针对业务灵活组合这一要求，采用（52）架构风格最为合适。

(52)批量处理　　B. 面向对象　　C. 黑板　　D. 解释器

答案D，**解析**：业务灵活组合就是自定义的意思，选虚拟机风格，解释器属于此风格。