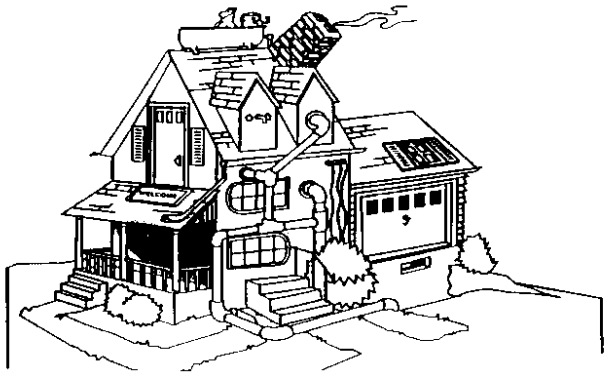
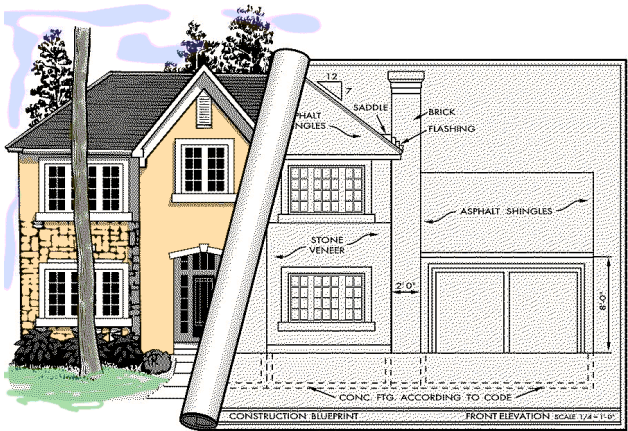
企业IT架构设计及管理
网上银行项目
第一部分：需求 (下)

2017年秋季学期





2012年7月21日，北京遭遇60年不遇的大雨

一个城市的建设，不仅仅是地面上的高楼大厦，城市的建设往往在我们看不到的地方

北京大暴雨故宫无积水 600年排水系统经受考验

2012年07月24日 14:31　来源：今日早报　参与互动(28)　9

61年不遇的京城大暴雨中，近600岁的紫禁城经受住考验

无积水，故宫很淡定

7月21日的北京大雨，故宫没有出现积水。永乐十八年(1420)落成的紫禁城为何能够在61年不遇的大暴雨中不现积水？它的排水系统中又有哪些特点？

沈阳一位古城建筑专家介绍，紫禁城的排水有明暗两套系统。明排水是通过铺地做出泛水，通过各种排水口、吐水嘴排到周边河中，暗排是通过地下排水道将水排到河里，而这条河就是内外金水河。承德避暑山庄也是这样建设的排水系统。归纳起来，一句话，整个系统，通过明暗等手段，到达一个目的：汇总往外排。

城外 依势而建

从地势上看，北京北依燕山，东临渤海，西北高东南低，所以水向东南流。从紫禁城来看，北门神武门地平标高46.05米，南门午门地平标高44.28米，差约2米。其排水设施充分利用了这一地形特点而建。这套排水系统的总特点，是将东西方向的雨水汇流入南北干沟内，然后流入内金水河。

疏通各个宫殿院落的排水系统有干线、支线，有明沟、暗沟、涵洞、流水沟眼等，经过精心测量、规划设计和施工，每年固定时间淘挖养护。值得一提的是，在城外安排了完整的排水系统，减轻了城内的负担。

一个好的建筑也要考虑多方面因素

雨雪、雷电

风力

承重：人，家具...

地基

欧洲最高的住宅－西班牙的“未来的象征”

A 47-story Spanish skyscraper forgets the elevator

By Tyler Falk | August 8, 2013, 1:28 PM PDT

In China they're building the second-tallest skyscraper in the world with the world's fastest elevators, stark contrast with Spain wh[illegible]tors in skyscrapers can't be taken for granted.

同样，对IT系统的建设也是如此。
项目干系人会关心一些不容易看到的系统能力

系统安全吗？

……

系统稳定吗？

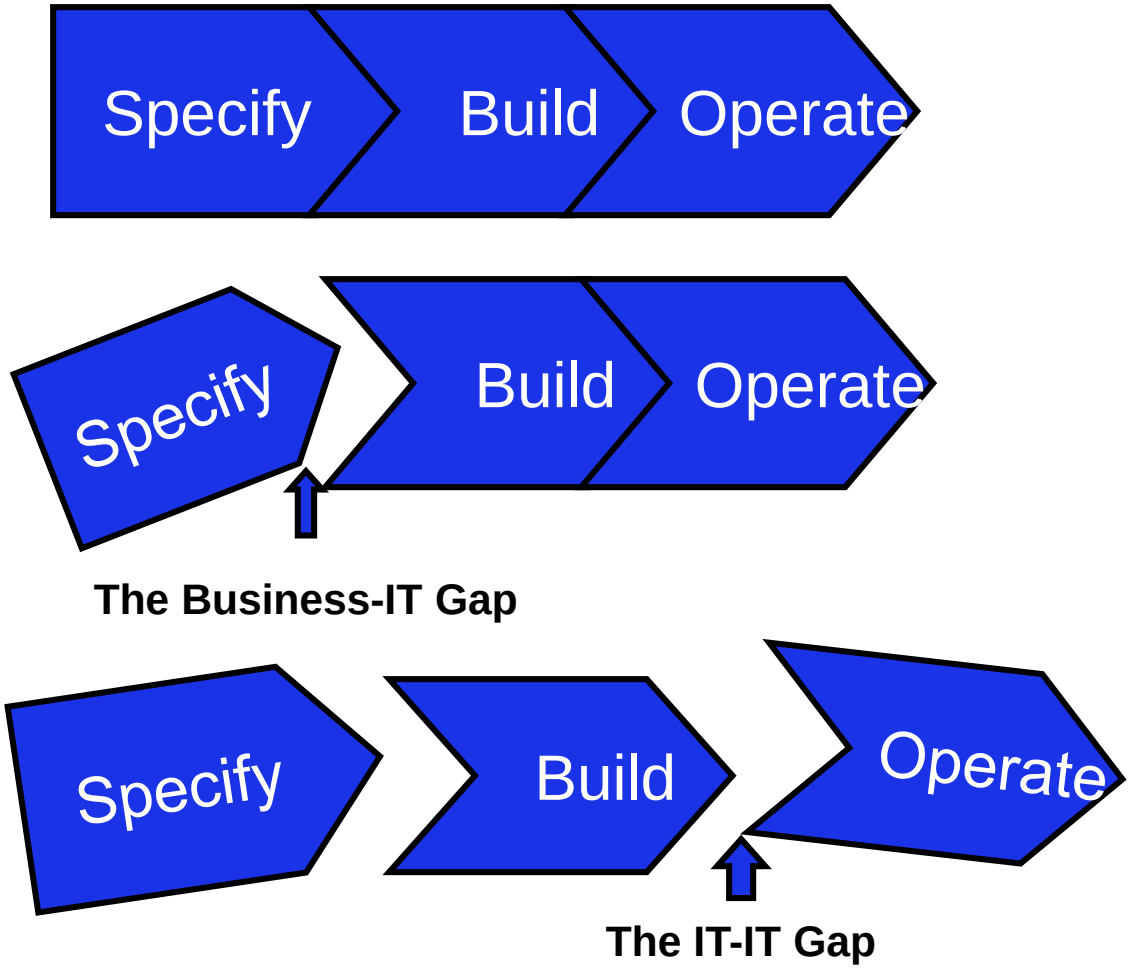
系统能
按时上线吗？

系统好用吗？

已有系统
会受到影响吗？

系统升级容易吗？

非功能需求为什么那么麻烦？



所以，作为一个系统设计的架构师，在讨论系统功能的同时，也要和客户一起，对系统的**非功能需求**做出分析和记录

定义系统需要满足的**期望和特性**

可能是**运行时**（如性能和可用性）

或**非运行**环境的需求（如可扩充性或可维护性）

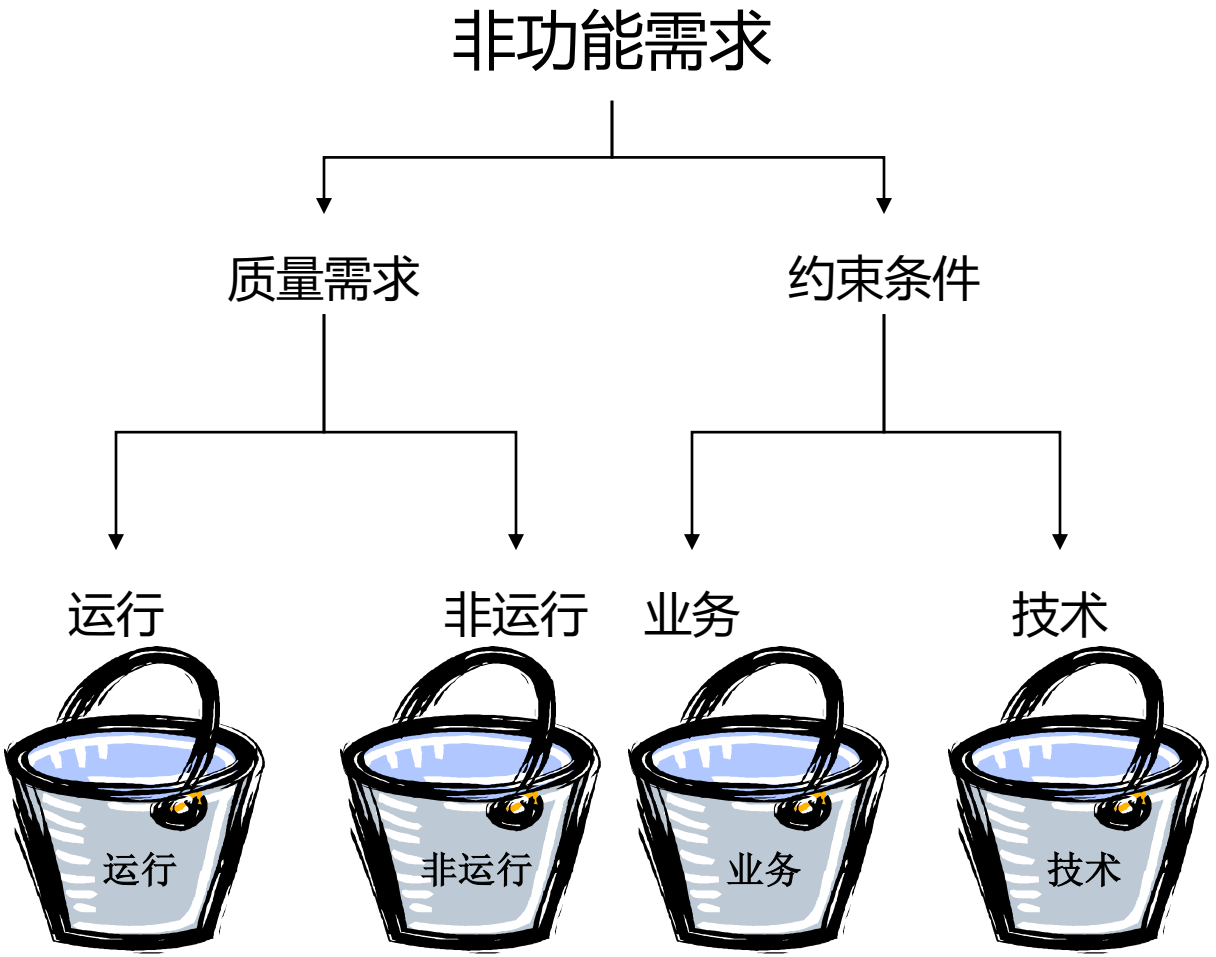
非功能需求

质量需求　　约束条件

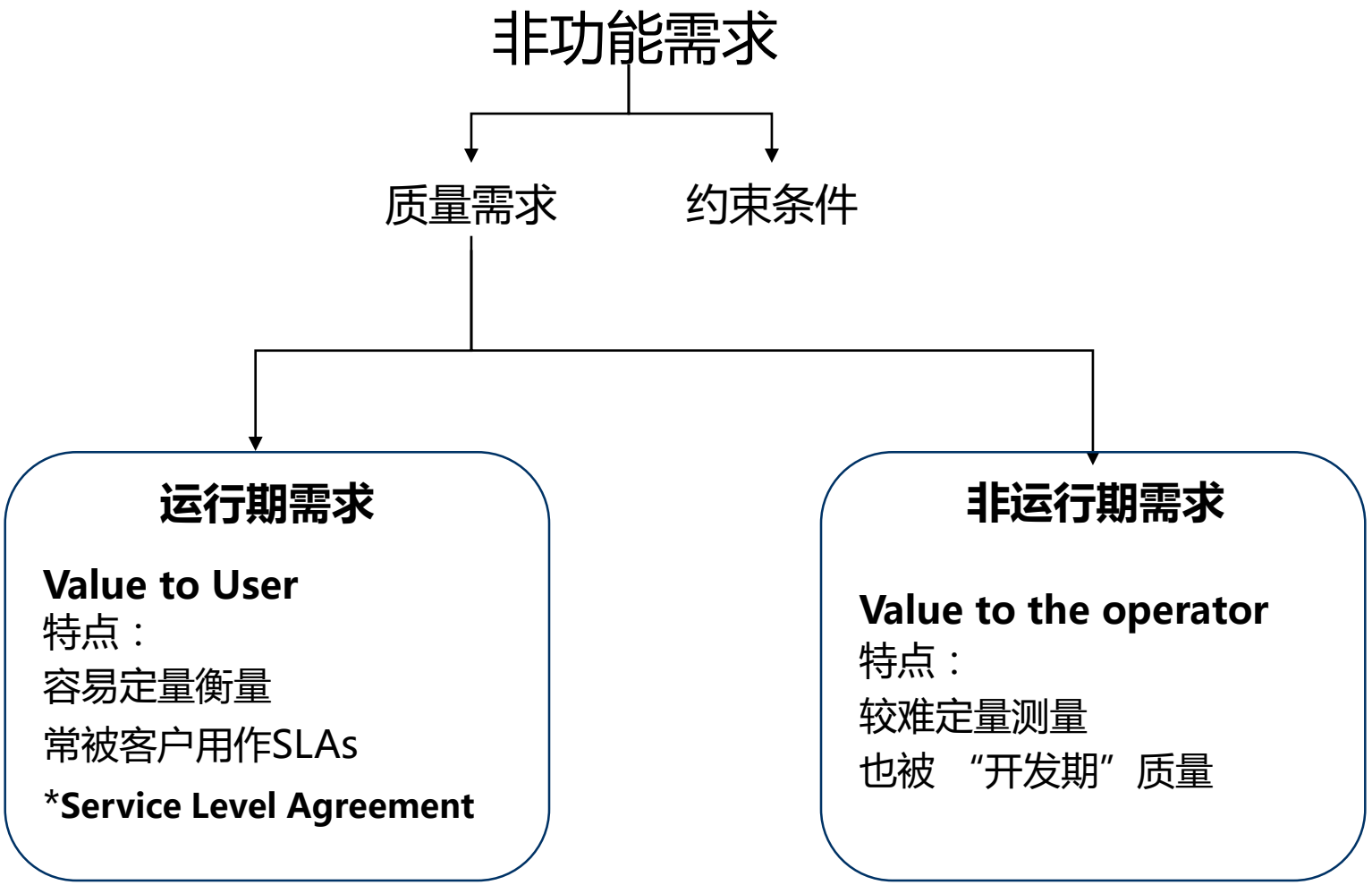
被给定的，在项目生命周期和系统范围内**不能被改变**的事实

其他因素，例如**已确定**的技术、可用的技能和预算费用等

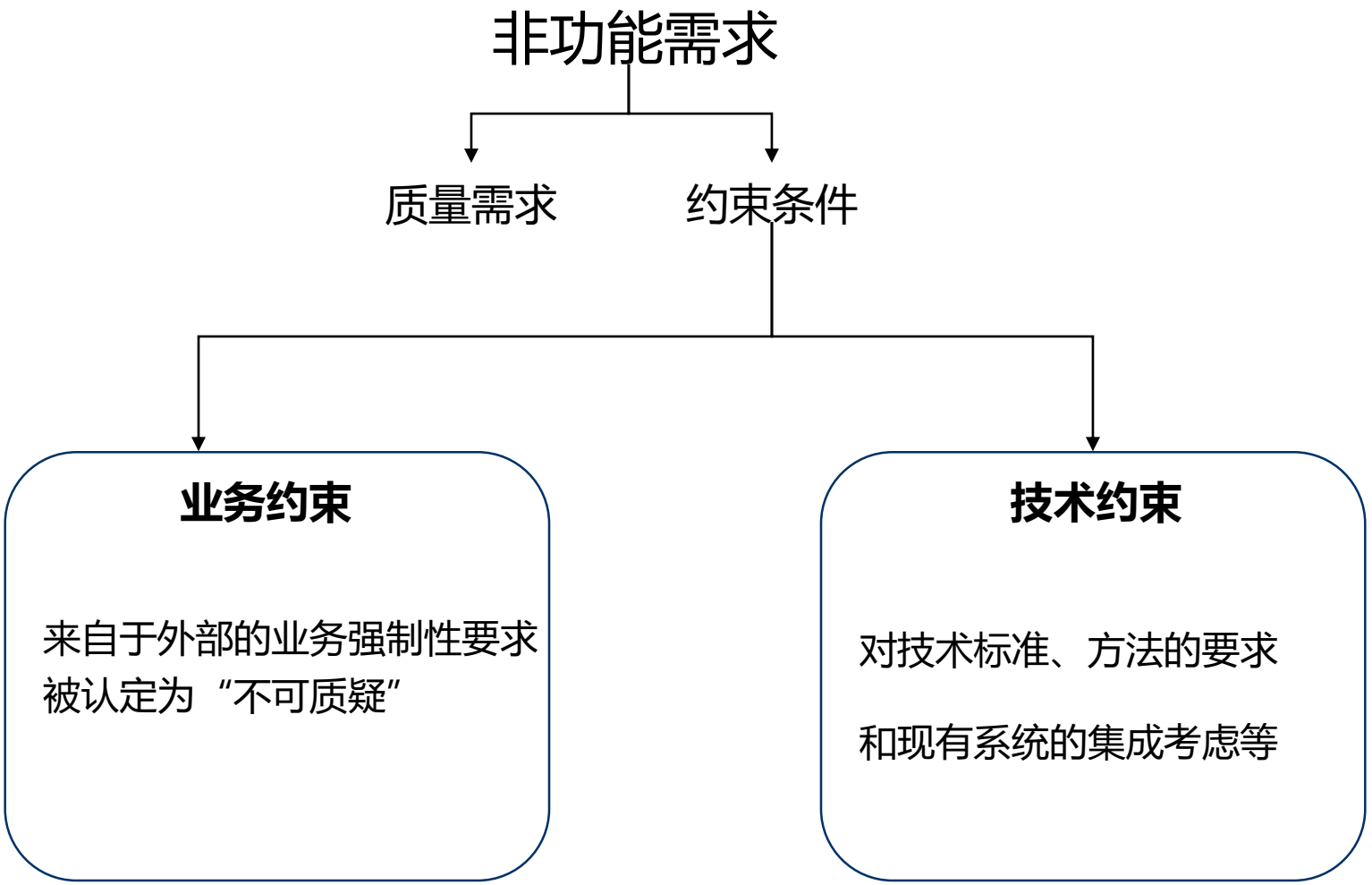
总结来看，一个系统的非功能要求分成以下4类



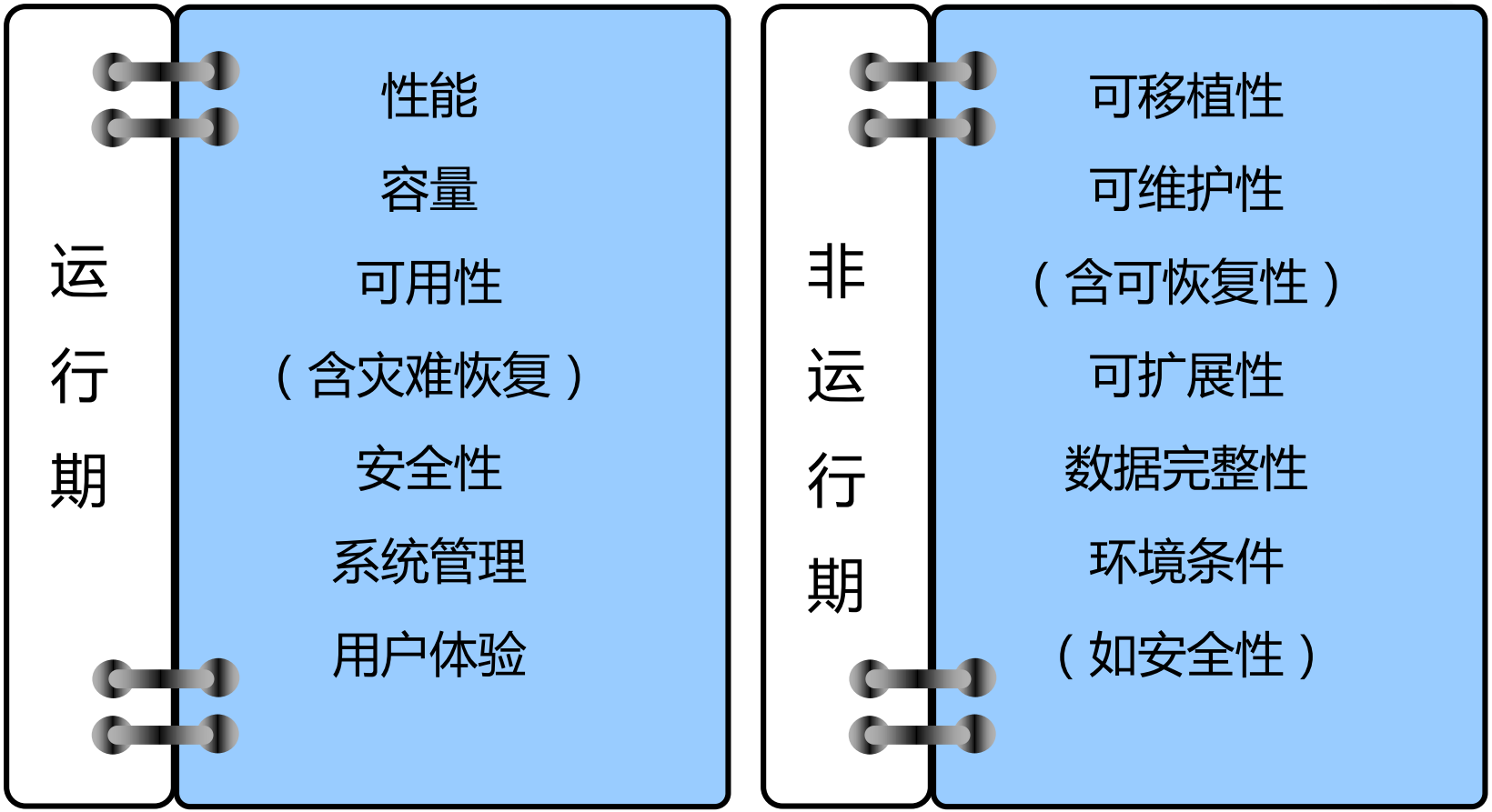
非功能需求之一：
质量需求定义了系统应达到的期望和特性



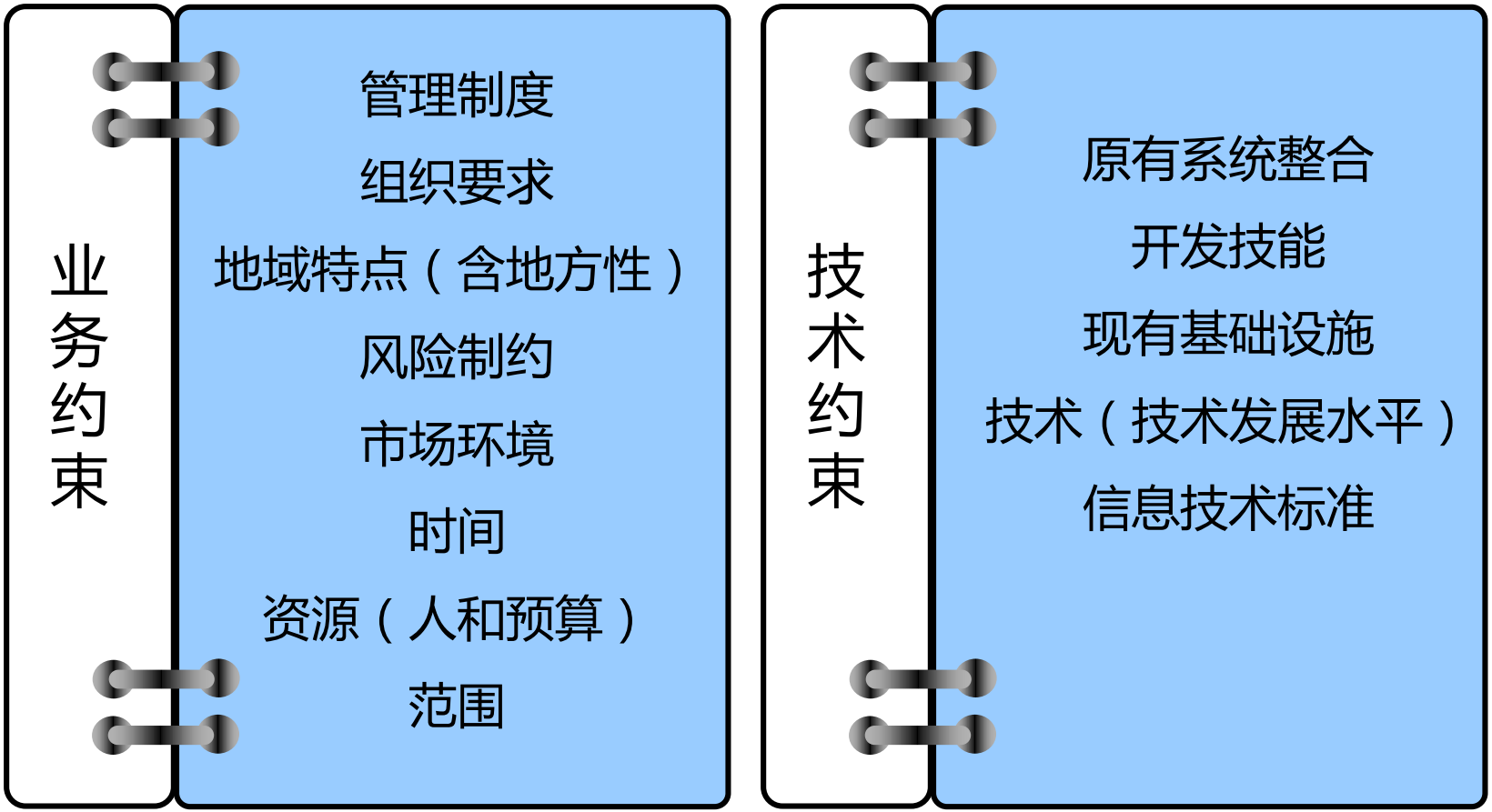
非功能需求之二：
约束性要求是在一个方案上的限制



质量需求都包括哪些方面呢？



约束都包括哪些方面呢？


业务约束
管理制度
组织要求
地域特点（含地方性）
风险制约
市场环境
时间
资源（人和预算）
范围
技术约束
原有系统整合
开发技能
现有基础设施
技术（技术发展水平）
信息技术标准

分组练习: 网上银行非功能需求讨论（10 分钟）

1. 分类

2. 优先级

分组练习：参考答案

类型	编号
性能	1，
容量	2，
可用性（含灾难恢复）	4，5
安全性	7，8，9，11
系统管理	6，20，17
用户体验	13，15，16，

类型	编号
可移植性	10
可维护性（含可恢复性）	18，19
可扩展性	3，
数据完整性	
环境条件（如安全性）	
技术约束	12，14
业务约束	

分组练习：对以下客户的非功能需求进行分析和分类

ID	类型	描述
NFR01	性能	常用页面（首页、首页各栏目）打开速度不高于2秒； 平均响应时间 3~5秒/笔查询 95%的普通查询交易响应速度不超过3秒； 95%的复杂查询交易、账务类交易响应速度不超过5秒 90%的内部系统报表查询不超过10秒；
NFR02	容量	内网：50人并发，外网：10,000人在线 数据量：约500万笔交易/日
NFR03	扩展	架构设计需要支持至少5年的运行扩展 无论是水平扩展，还是垂直扩展都不需要修改应用
NFR04	可用	保证7×24小时持续提供正常服务，年宕机故障数不超过2次的持续工作能力 保证重大故障出现时，在30分钟内灾备环境可以启动并提供服务；2个小时，生产环境恢复运行的能力
NFR06	系统管理	整个系统的应用和数据库要能被统一监管；

分组练习：对以下客户的非功能需求进行分析和分类

ID	类型	描述
NFR07	安全	内外网物理隔离 系统需要用户认证、授权、角色管理，可以对系统数据、系统功能的访问权限设置。对登录用户的身份进行认证，并跟踪用户的操作，进行安全审计。 对不同用户的数据访问进行定制，保证数据的安全性与一致性。
NFR08	可移植	由于“网上银行”属于整个多渠道体系的一部分，今后应用和数据可能会采用统一平台（尚未建设），所以网银项目的应用和数据都需要能跨平台、方便移植。
NFR09	标准	网银应用应该采用J2EE标准。
NFR10	技术约束	目前银行已经采购Oracle数据库，在可能的情况下优先考虑Oracle数据库
NFR11	易用	界面要支持中英文，友好，易学、易用。 以用户为本，尽量做到人性化设计。 人性化的操作环境使用户易学易用，全系统操作提示统一。 系统管理人员不需要进行复杂的操作即可完成各种日常系统管理

下面让我们来看一看系统架构设计时需要重点考虑的几个非功能需求

性能-定义

定义 - 在约束条件下一个系统或组件可完成指定功能的程度， 例如速度、精确性、内存使用率

- 一个简单的性能描述，例如“响应时间低于1秒”是不充分的。它必须包含如下元素：
  - 什么是性能的要求（响应时间、交易时间）？
  - 何时交付（通过定义运行时间）？
  - 在哪进行测量（在哪个系统，使用什么工具）？
  - 那种情况下工作负荷才是有效的（测试交易是否可用）？
  - 针对哪一个用户和系统（有多少）？
  - 如何去定义比例以达成既定价值（平均95%）？

*[IEEE-610.12]

性能-定义

定义 - 在约束条件下一个系统或组件可完成指定功能的程度， 例如速度、精确性、内存使用率，应包括以下四个方面：

响应时间

Response Time
在目标系统内完成某个业务流程、交易或批量作业所需要的时间

吞吐量

Throughput：
一个系统或组件在给定的时间内能够完成的某项工作量
如：日总交易量，日新订单量，峰值时每小时的交易量，……

资源使用率

Utilization / Workload
不同时间的系统资源使用情况，
如：峰值时间的CPU使用率，内存使用率，网络带宽使用，……

静态容量

Static Volumetric
系统静态数据容量
如：系统总注册用户数，用户的账户数量，……

*[IEEE-610.12]

响应时间

样例

	Target Average End-to-End Response Time (seconds)		
Frequency of Use **Business Transaction Complexity**	**High Frequency** (e.g. > 100 times per day)	**Medium Frequency** (e.g. > 10 times per day but < 100 times per day)	**Low Frequency** (e.g. < 10 times per day)
Simple Transaction	*1 – 2*	*2 - 3*	*3 - 4*
Medium Transaction	*3 - 5*	*4 - 7*	*5 - 10*
Complex Transaction	*6 - 10*	*8 - 15*	*11 - 20*
Very Complex Transaction	*11 - 20*	*16 - 30*	*21 - 40*
Other[1]	*> 20*	*> 30*	*> 40*

吞吐量

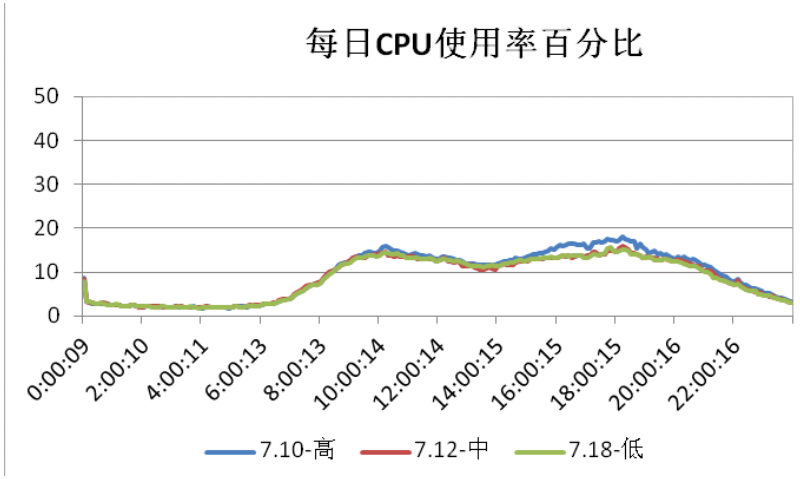
样例

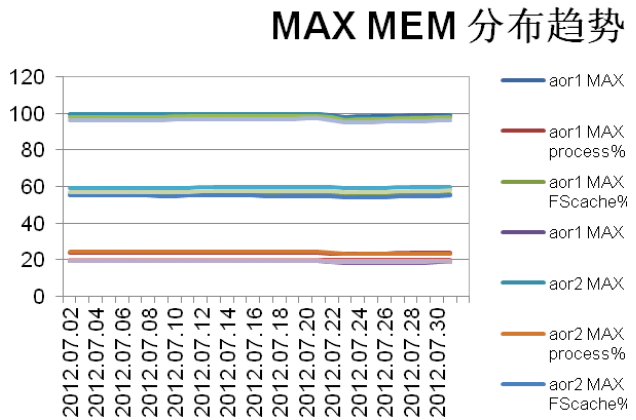
Throughput Requirements	Volume
User Volumetric	
Average number of Concurrent Users	100
Maximum number of Concurrent Users	300
Additional Support/Support Users	500
Total Number of users registered	1500
Average Number of Transaction	
Average Simple Transactions Per Business Transaction	20
Average Business Transactions Per Hour (8 hours per day)	200
Average Business Transactions Per Day	1600
Average Business Transactions Per Minute	4
Average Business Transactions Per Second	6.5
Peak Transaction Volumetric	
Peak Transactions Per Second (Avg * 1.5)	10

资源使用

样例

	XX应用	XXX 应用	XXDB	XXX DB	文件处理子系统
CPU	●	●	●	●	●
IO	●	●	●	●	●
Memory	●	●	●	●	●
Task	●	●	○	○	○



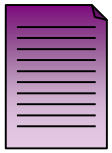


静态容量

样例

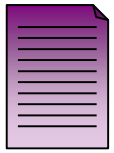
1.1 业务处理特点要求	1. 多少IT transaction/per business tran. 2. 每个交易业务需要多少数据源/数据量需求访问（MB/tran) 3. 那些内容需要记录日志，日志保留周期要求 4. 正常业务中，平均交易失败率，需要涉及额外处理；如再处理、冲正、等机制 5. SQL/IT tran
1.2 数据容量要求指标	系统相关各种类型数据的容量要求，如大小和处理频率要求
1.3 数据保留周期要求指标	系统相关各种类型数据保留周期要求； 及被清除时间/周期，或不同存储级别要求说明， 如**多级存储**需求/以及迁移分类标准
1.4 消息数据处理指标	1. 消息数据存储容量 2. 消息数据大小约定 3. 消息保留周期

性能-设计原理与技术



应用组件模型

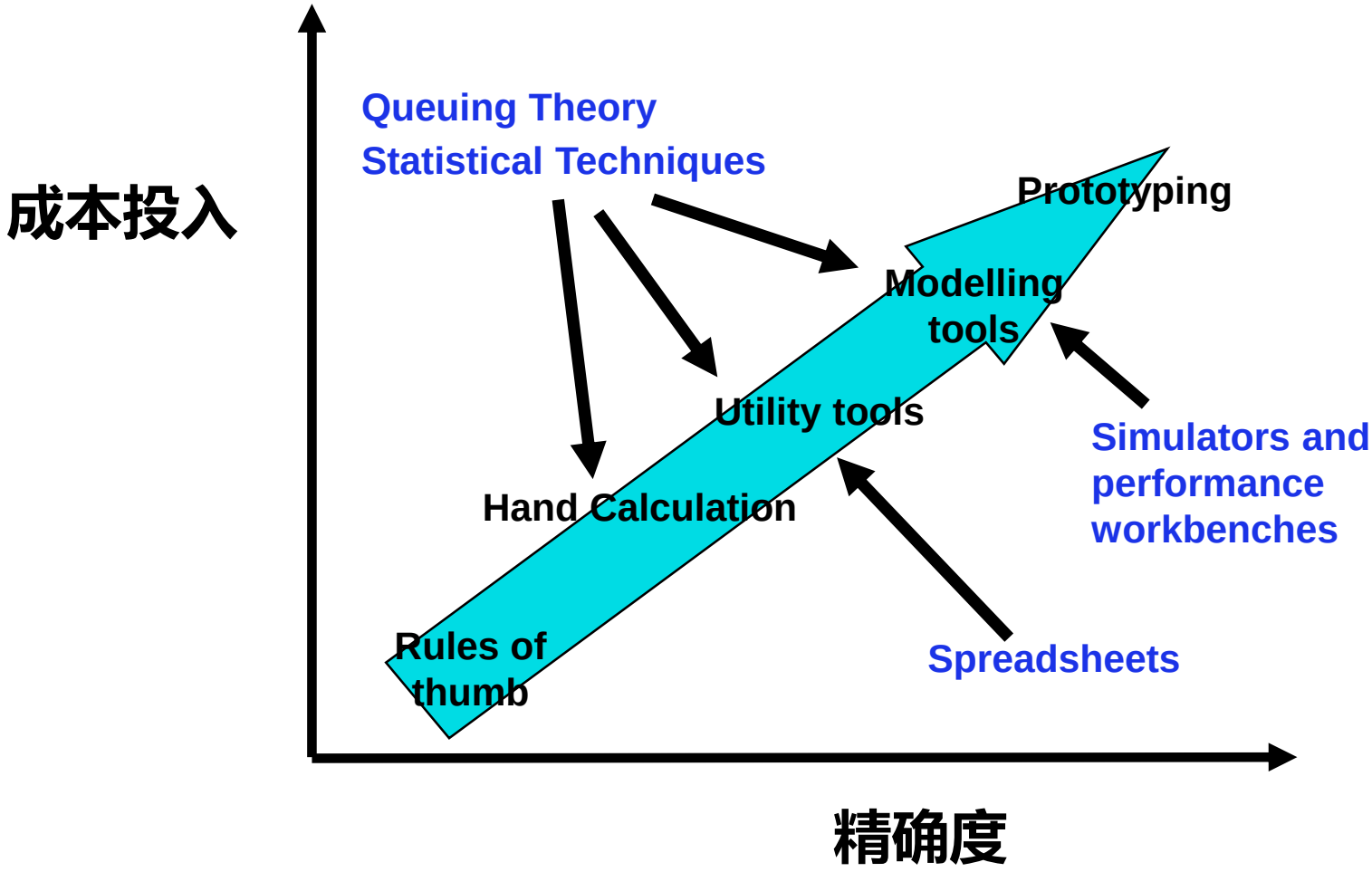
- 优化业务路径
- 优化单元组件性能成本(搜索、加解密...)
- 考虑平行设计
  - (多进程、多线程、异步调用...)
- 避免共用资源冲突
  - (数据库行级锁、表级锁的使用...)
- 减少系统开销
  - (连接池、批量，压缩技术、长连接...)
- 合理的流量控制和问题隔离(避免系统短板)
- 数据/组件管理
  - (缓存数据、数据复制、数据分类存储...)



系统运行模型

- 增加带宽
- 增加服务器配置
- 增加服务器数量
- 增加磁盘
- 增加内存
- 服务器调优
- 改变产品

性能-预估方法


成本投入
Queuing Theory
Statistical Techniques
Prototyping
Modelling tools
Utility tools
Simulators and performance workbenches
Hand Calculation
Rules of thumb
Spreadsheets
精确度

可用性-定义

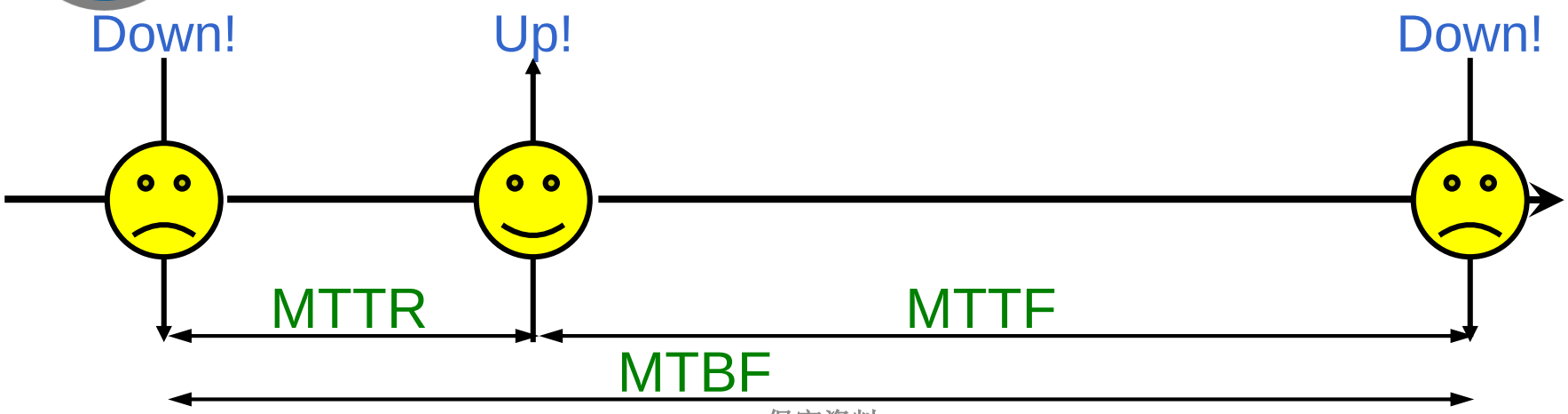
可用性

定义：The degree to which a system or component is operational and accessible when required for use *[IEEE-610.12]

MTTR Mean Time to recovery 平均恢复时间
是一个组件、子系统或系统的还原（包含修复）时间

MTTF Mean Time to Failure 平均无故障时间
是指一个系统从可用到不可用之间的平均总时长

MTBF Mean Time between Failure 平均故障间隔时间
是给定的组件，子系统或系统连续出现错误之间的平均时间



高可用、操作连续性和连续可用性

高可用 (HA)

System is *always* available *whenever it's expected to be.*

- ✓*No unscheduled outages*
- ✓*Scheduled outages permitted.*

操作连续性 Continuous operations (CO)

System is *intended* to be available *all the time:* "24 by 7."

Although unscheduled outages may occur, *no scheduled outages*

连续可用性 Continuous availability (= HA + CO)

Business is *always* available *all the time.*

	HA	CO	CA
Scheduled outages	Y	N	N
Unscheduled outages	N	Y	N

可用性-设计原理

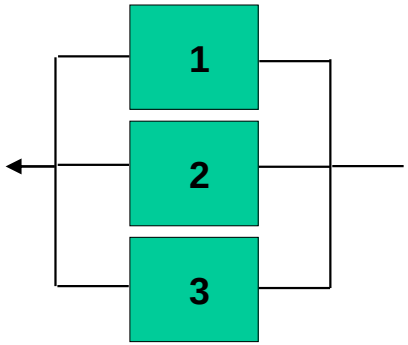
- 相互连接的组件组成一个链，为保证可用性每一个都依赖于前一个组件
- 总的可用性永远比组件链中的任何一个组件的可用性要低

Availability = A^1 x A^2 x A^3

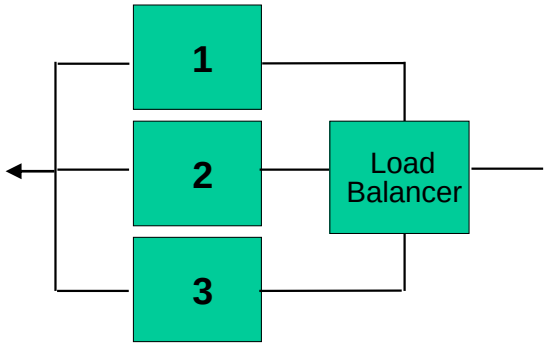
Availability = 0.95 x 0.95 x 0.95 = **86%**

可用性-设计原理

- 经由复制实现组件冗余



- 总可行性比单一组件可行性要高

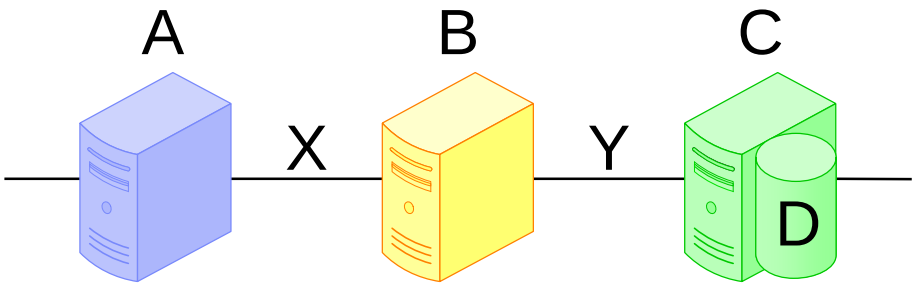


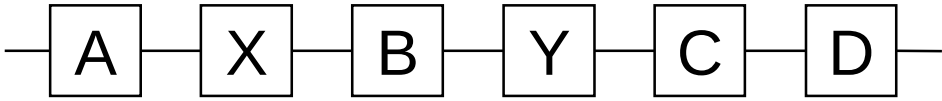
Availability = 1-[(1-A(1))x(1- A(2))x (1-A(3))]

Availability = 1 - [(1-0.95) x (1-0.95) x (1-0.95)] = 99.99%

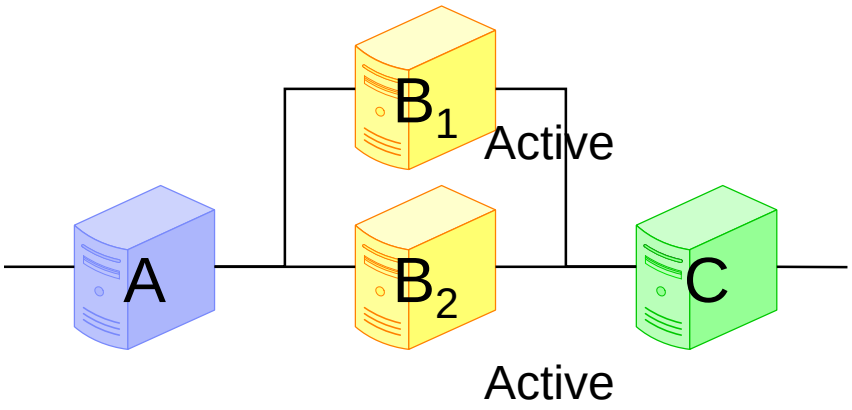
可用性-设计模式

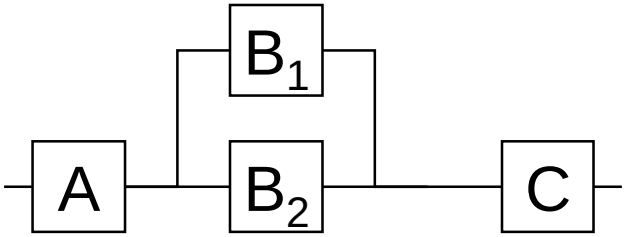
1. 反模式 串行模式

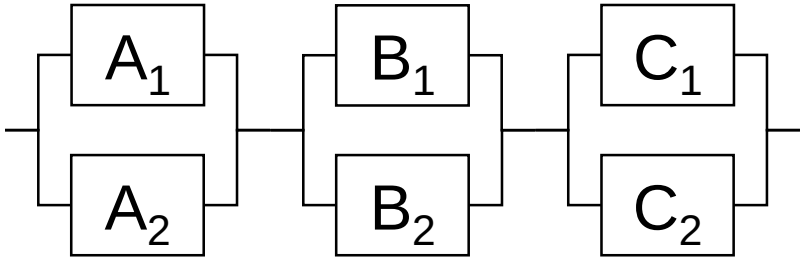




2. Active-Active/均衡模式

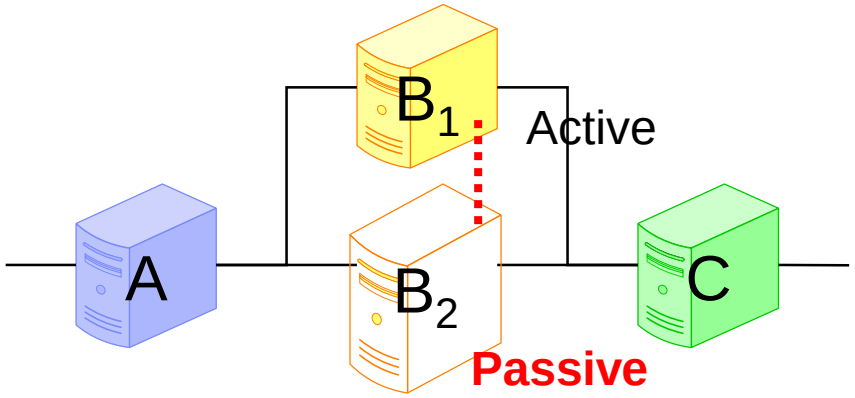


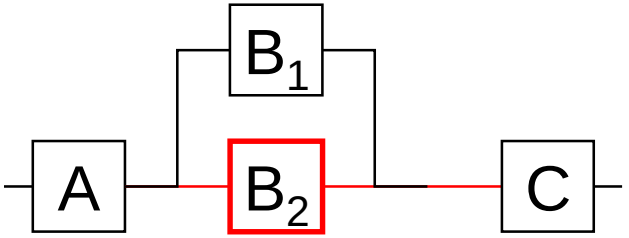




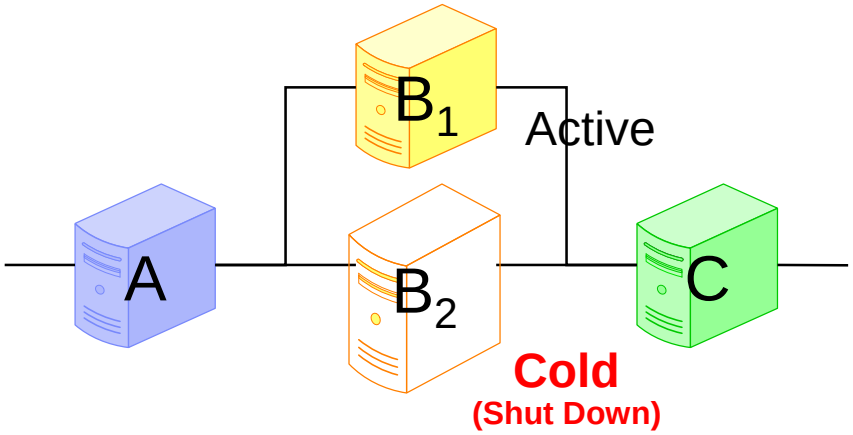
可用性-设计模式

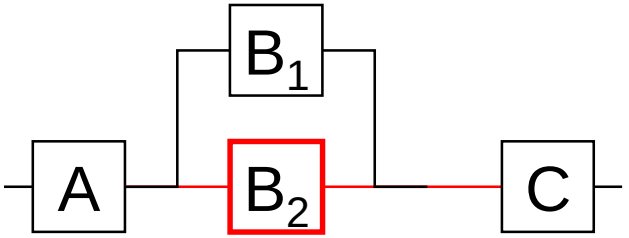
3. Active-Passive /热备模式





4. Active-Passive /冷备模式





安全性-定义

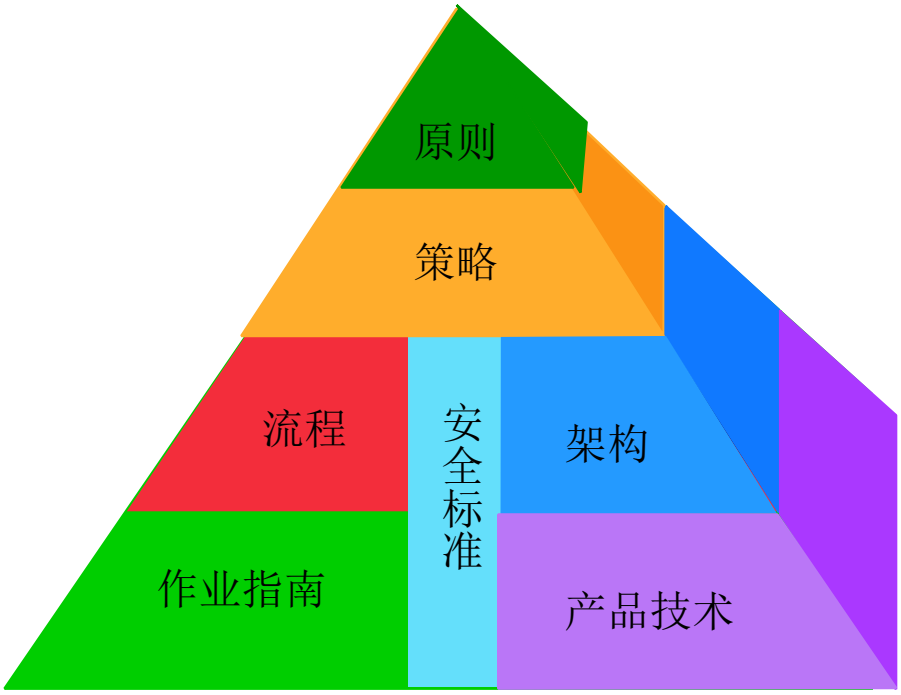
- 定义：用来保护IT系统对抗恶意使用的能力，同时仍允许正常使用
- 安全性包括：
  - 安全：降低或排除危险、焦虑、风险和不信任
  - 保护：抵御攻击（来自内部和外部）和诈骗（包含资产误用、身份冒充）。保护有形资产（IT系统和应用、储存或运送的信息）和无形资产（如名誉）
  - 保证：确保正确和可靠的操作方式，强化所有者及身份认证

安全性-公共标准

- 审核：审核、组件保护、资源利用率
- 访问控制：数据保护、组件保护、安全性管理、组件访问、密码支持、识别与认证、通讯、信任路径/渠道
- 流量管理：通讯、密码支持、数据保护、组件保护、信任路径/渠道、隐私
- 识别/证书：密码支持、数据保护、组件保护、识别与授权、组件访问、安全管理、信任路径/渠道
- 方案整合：密码支持、数据保护、组件保护、资源利用率、安全管理

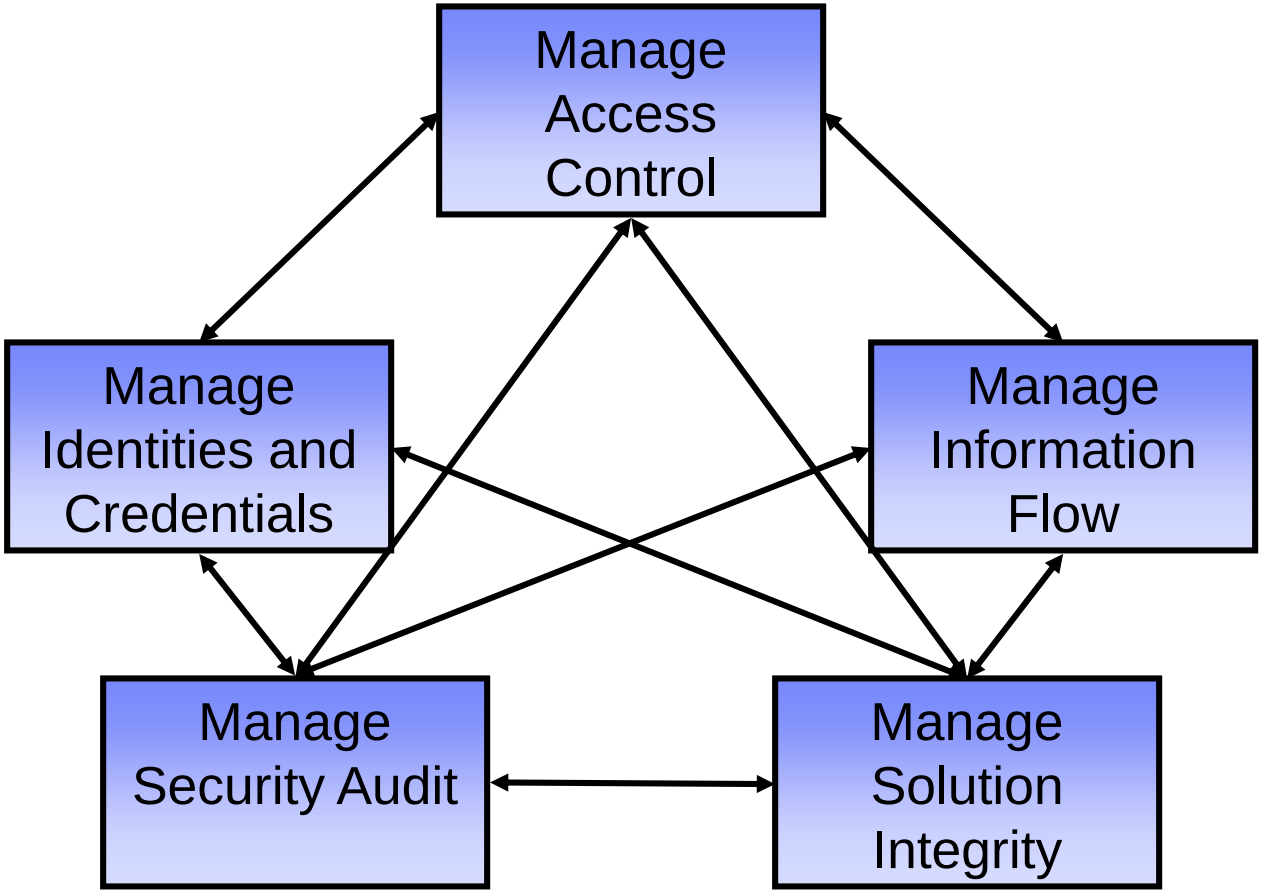
- 请看：http://www.commoncriteriaportal.org

安全性-信息安全不止是技术的实现，更是管理的范畴



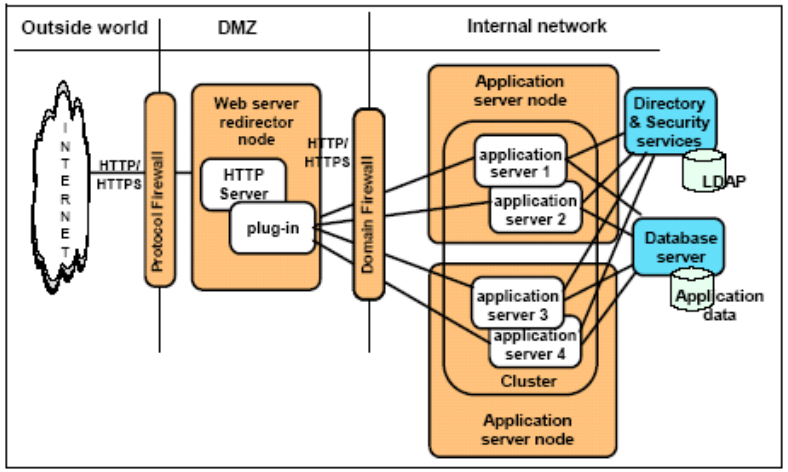
35

安全性-设计模式



可扩展性-定义

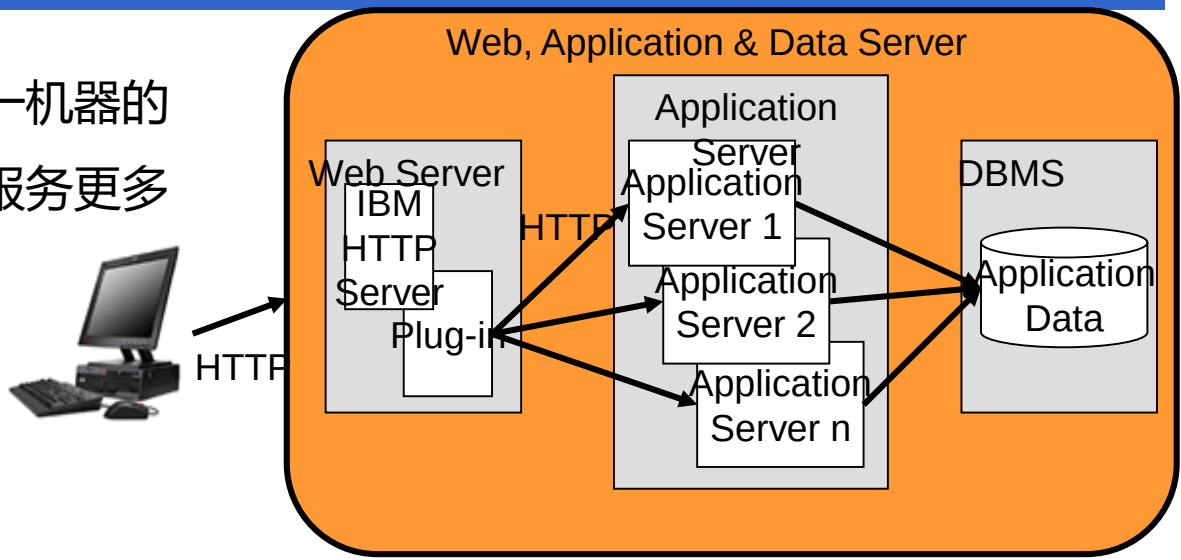
- 定义：通过增加更多的资源以响应用户增长能力
- 它和性能不同，它的目的不是为了增加性能，而是当面临更大的吞吐量时（例如增加用户和交易负荷）仍维持性能



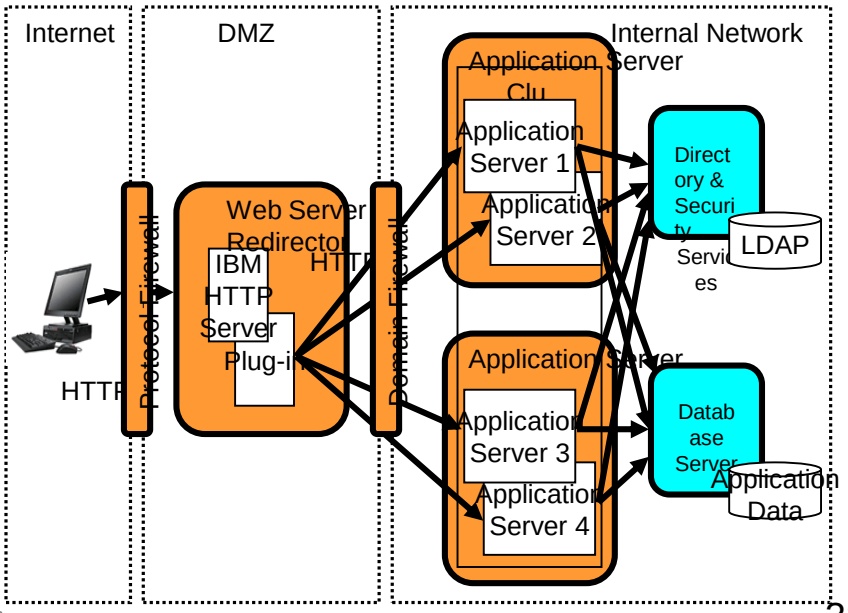
可扩展性往往很难预测！

可扩展性-设计原理

- **垂直扩充**是指通过增加单一机器的额外资源使一应用程序能服务更多的需求，例如：
  - 增加内存
  - 使用更快的CPU



- **水平扩充**是指通过增加更多的计算机共同处理应用程序以实现扩充
  - 需要一个通过负载均衡实现计算机之间的需求分配
  - 允许你通过增加或移除计算机来改变性能



原有系统整合-定义

- 原有系统整合（企业整合）可被定义为：

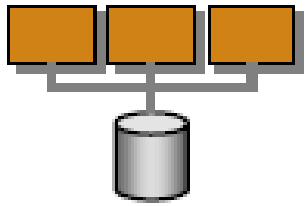
  “在不同区域的跨平台下，提供在多个运行的应用上实现公共的业务流程和数据分享的能力”

- 简单的说，它与计算机系统、人和业务的连接有关。
- 整合是困难的，因为以下因素：
  - 任何两个系统是不同的
  - 人，商业，系统等因素
  - 变化让集成更难
  - 没有通用的标准

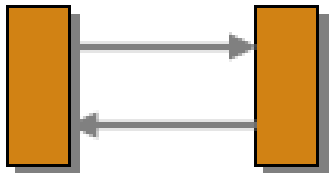
原有系统整合-设计模式

- 文件传输
  - 利：所有平台
  - 弊：并实时传输
- 共享数据库
  - 利：保证数据一致性
  - 弊：并未进行行为整合
- 远程过程调用
  - 利：当需要时可调用业务功能
  - 弊：特定语言，不可靠，速度慢
- 消息
  - 利：根据需要进行数据交换
  - 弊：需要消息系统

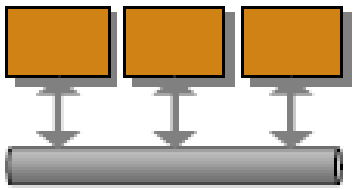
文件传输



共享数据库



远程过程调用



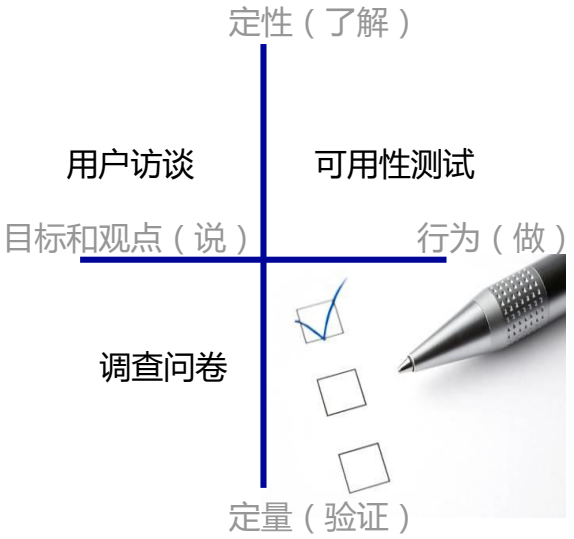
消息

*See [Hope]

用户体验-定义和分析

用户体验

用户体验：结合用户的操作特性，知觉特性，认知心理特征，设计产品，使产品更符合用户的习惯，经验和期待。

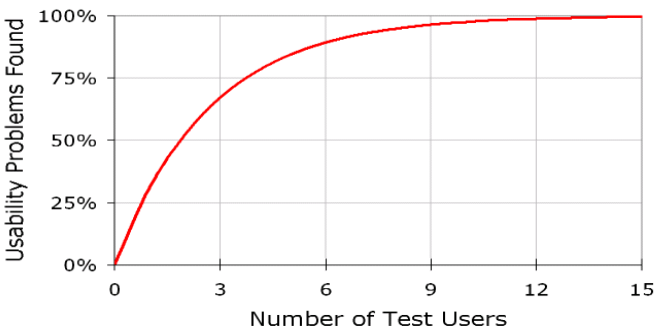


用户体验的维度

- 可用性
- 内容性
- 视觉性
- 功能性

用户体验分析方法

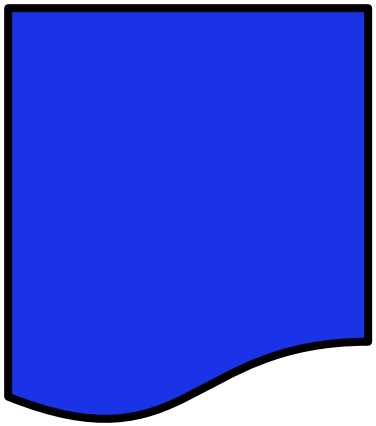
- 用户问卷调研概况
- 人口统计学信息
- 职业结构与收入状况
- 功能使用情况
- 用户体验评价
- 用户类型划分
- 用户模型详述



Tom Landauer 和 Jakob Nielsen
的可用性问题数量与参与测试的用户数量关系

*See [Hope]

考虑了系统的非功能需求，我们应该把这些需求正确的记录下来，为未来系统设计做好准备



非功能需求模板

Thank You !