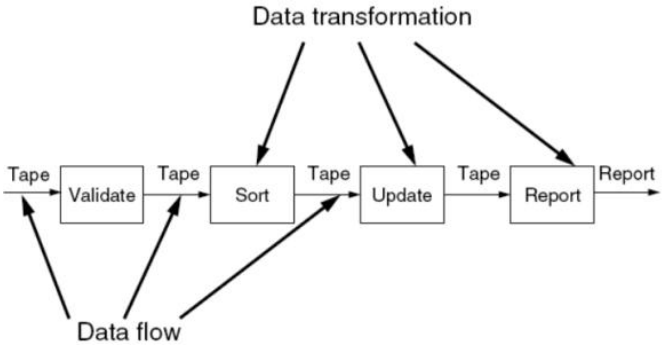
1. 体系结构的概念

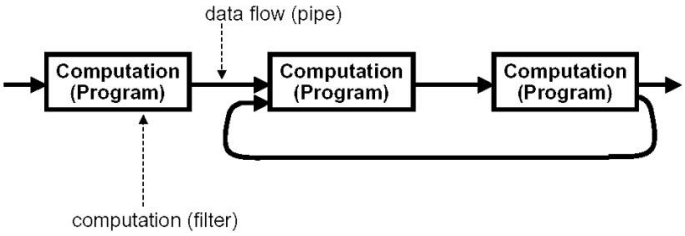
软件体系结构包括构成系统的设计元素的描述，设计元素的交互模式，以及在这些模式中的约束。精简为：组件+连接件+约束。

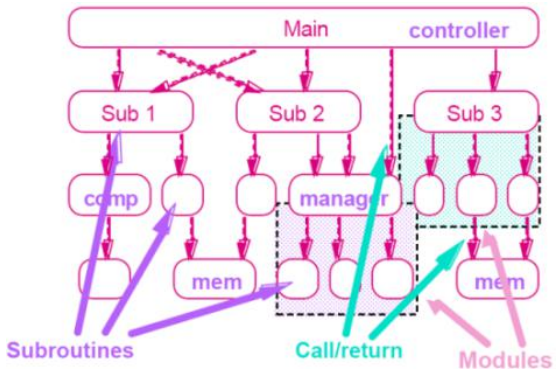
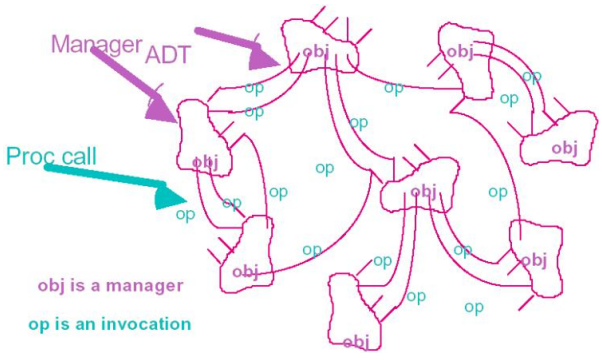
2. 体系结构风格

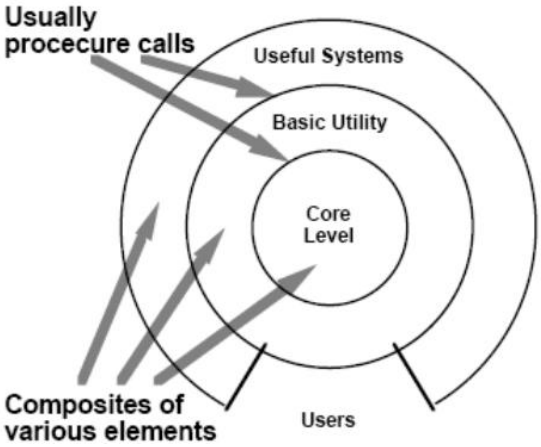
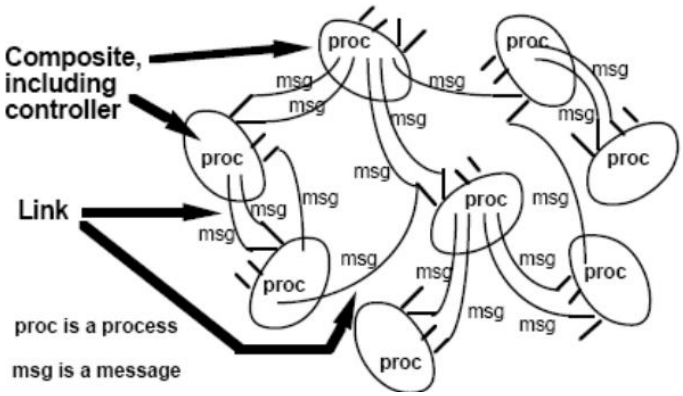
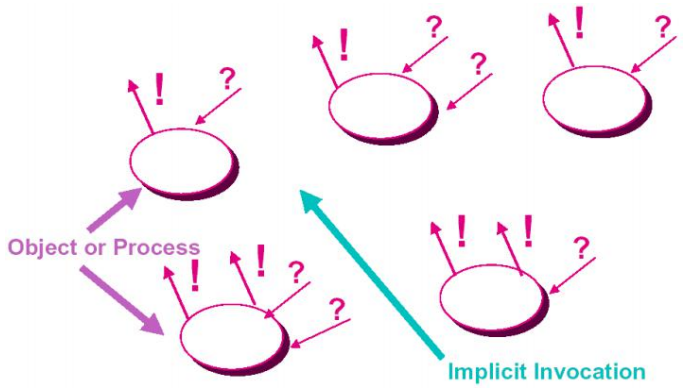
体系结构风格是描述特定系统组织方式的惯用范例，强调组织模式和惯用范例。精简为:构件/连接件集、拓扑和约束。

<table>
<tr><th colspan="2">风格</th><th>特点</th><th>图</th></tr>
<tr><td rowspan="3">数据流系统</td><td rowspan="2">批处理系统</td><td>构件：data transformation
连接件：data flow
约束：每个步骤都是一个独立的程序，每一步必须在前一步结束前才能开始，数据必须是完整的以整体的方式传递

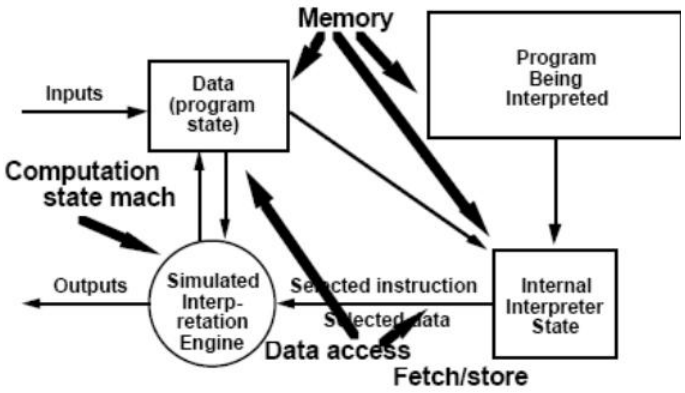
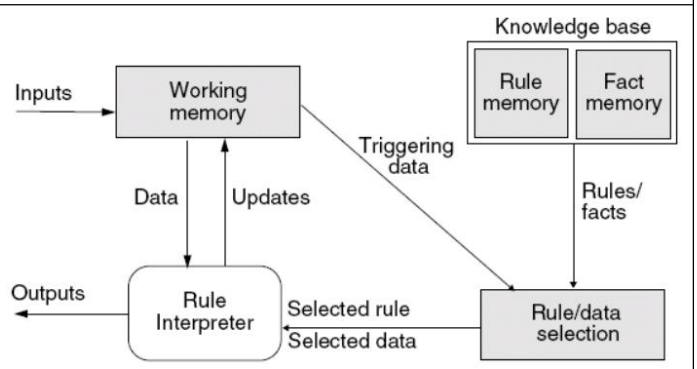
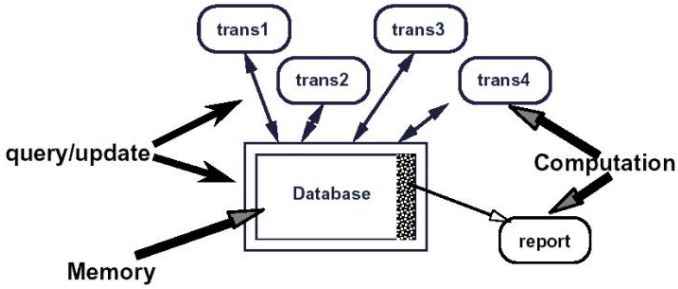
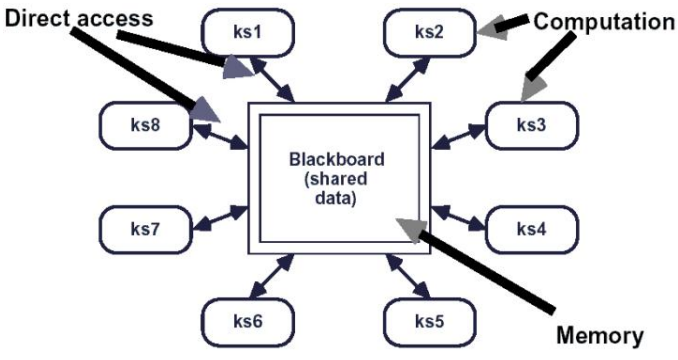
每个处理步骤是一个独立的程序；
每一步必须在前一步结束后才能开始；
数据必须是完整的，以完整的方式传递。</td><td rowspan="2"> 
</td></tr>
<tr><td rowspan="2">管道/过滤器</td><td>构件：过滤器
连接件：管道
约束：过滤器都是各自独立的，相互之间并不存在联系

优点：良好的隐蔽性，高内聚、低耦合；便于设计者理解；支持功能模块的重用；系统易于维护和扩展；支持某些特定属性的分析；支持并发执行。</td></tr>
<tr><td>缺点：不适合于交互性很强的应用；数据传输无通用标准，降低系统性能；需要处理同步问题。</td><td> 
</td></tr>
<tr><td rowspan="3">调用/返回系统</td><td rowspan="2">主/子程序系统</td><td>构件：过程和显式可见的数据
连接件：过程调用和显式数据共享
约束：适用于计算可以被分层定义的应用

优点：逐步分解；
单线程控制；
子程序正确性受主程序正确性影响。</td><td rowspan="2"> 
</td></tr>
<tr><td>缺点：只适用于可以定义为一系列步骤的问题；
子系统结构不清晰。</td></tr>
<tr><td>面向对象系统</td><td>构件：对象
连接件：消息和消息调用
适用于界面与实现分离的系统
优点：对象抽象使得组件和组件之间的操作以黑箱的方式进行；封装性使得细节内容对外部环境得以良好的隐藏。对象之间的访问是通过方法调用来实现的；考虑操作和属性的关联性，封装完成了相关功能和属性的包装，并由对象来对它们进行管理；使用某个对象提供的服务并不需要知道服务内部是如何实现的。</td><td> 
</td></tr>
</table>

		缺点：对象之间的耦合度较紧：一个对象和另一个对象通过过程调用等进行交互，必须知道对象的标识。只要一个对象的标识改变了，就必须修改所有其他明确调用它的对象；必须修改所有显式调用它的其它对象，消除由此带来的一些副作用。例如 A 使用了对象 B，C 也使用了对象 B，则 C 对 B 的使用所造成的对 A 的影响可能是不可预测的；过多的对象难以维护；过多的交互难以维护。	
	分层系统	构件：各层次内部的构件 连接件：层间交互协议 约束：适用于不同服务类别可以被分层管理的 优点：每层为上一层提供服务，使用下一层的服务，智能访问相邻层；大的问题分解为小问题逐步解决，降低复杂度；修改一层最多影响两层。	
		缺点：上层必须知道下层的身份，不能调整层次之间的次序；层层相调，影响性能。	
独立构件	通信进程系统	**Problem:** This pattern is suitable for applications that involve a collection of distinct, largely independent computations whose execution should proceed independently. The computations involve coordination of data or control at discrete points in time. As a result, correctness of the system requires attention to the routing and synchronization of the messages. 组件：收发消息的进程 连接件：消息	
	事件系统隐式	构件：激起事件但不知道最终接收方的进程 连接件：事件－过程绑定，对某事件进行注册了的自动进程调用 控制结构：分权，各个构件对信号接受方并不了解 为软件重用提供了强大的支持。当需要将一个构件加入现存系统中时，只需将它注册到系统的事件中；为改进系统带来了方便。当用一个构件代替另一个构件时，不会影响到其它构件的接口。	
		构件放弃了对系统计算的控制。一个构件触发一个事件时，不能确定其它构件是否会响应它。而且即使它知道事件注册了哪些构件的构成，它也不能保证这些过程被调用的顺序；数据交换的问题。有时数据可被一个事件传递，但另一些情况下，基于事件的系统必须依靠一个共享的仓库进行交互。在这些情况下，全局性能和资源管理便成了问题；既然过程的语义必须依赖于被触发事件的上下文约束，关于正确性的推理存在问题。	

<table>
<tr><td rowspan="2">虚拟机</td><td>解释器</td><td>构件：一个解释器引擎和三个存储区
连接器：对存储区数据的访问和过程调用
优点：
可以模拟非本地原生支持的功能
可以在模拟极端条件监测系统
用途广泛

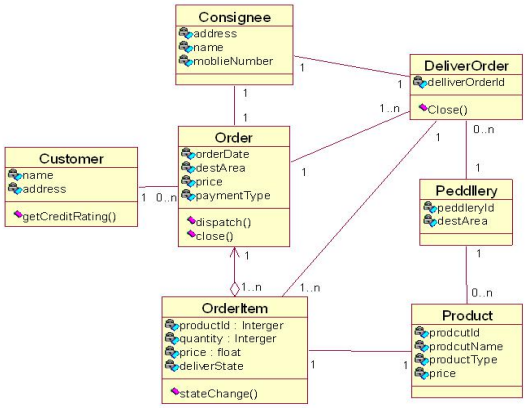
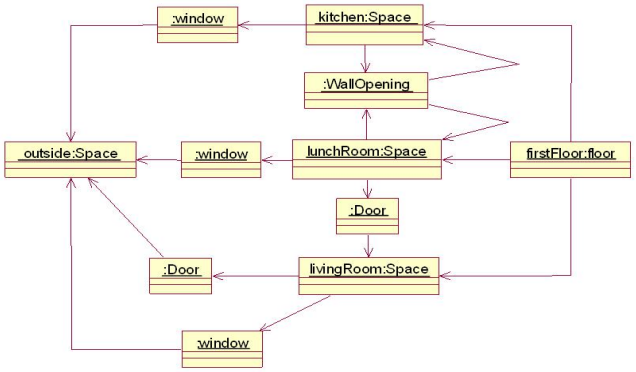
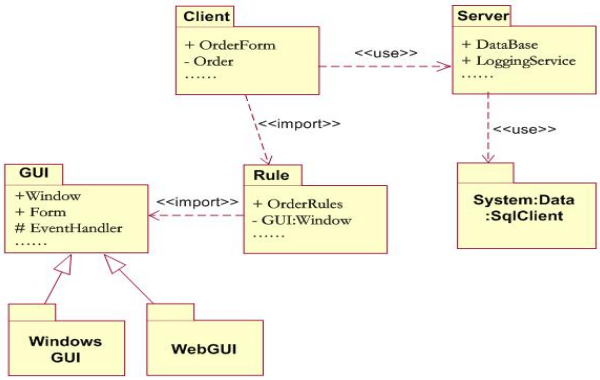
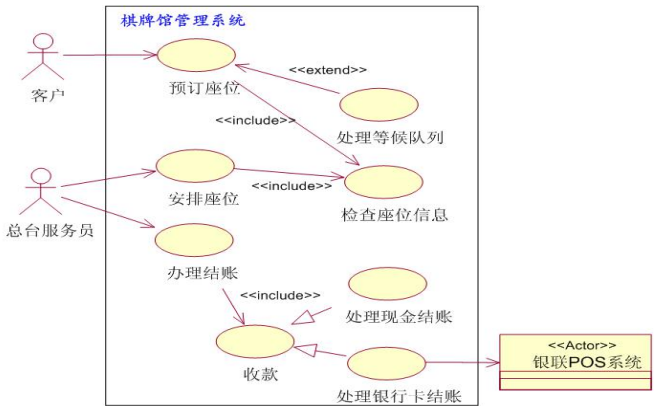
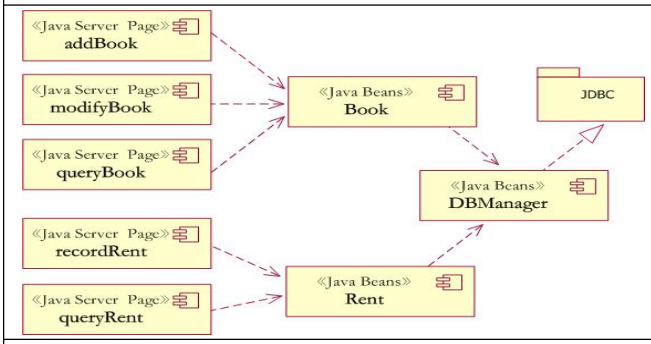
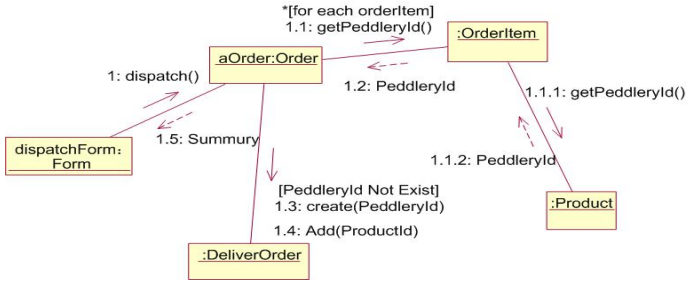
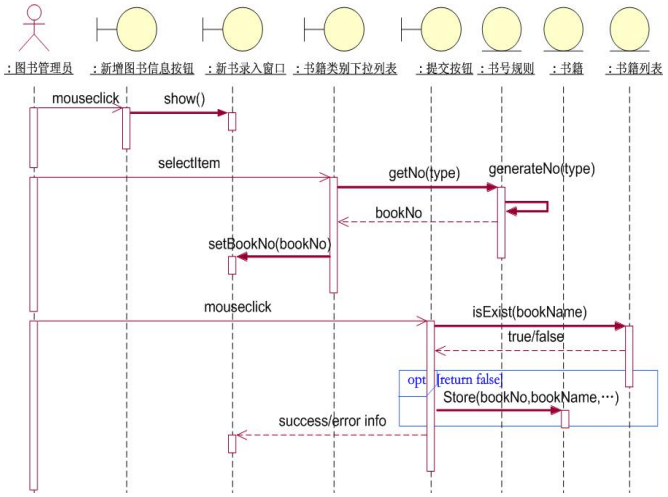
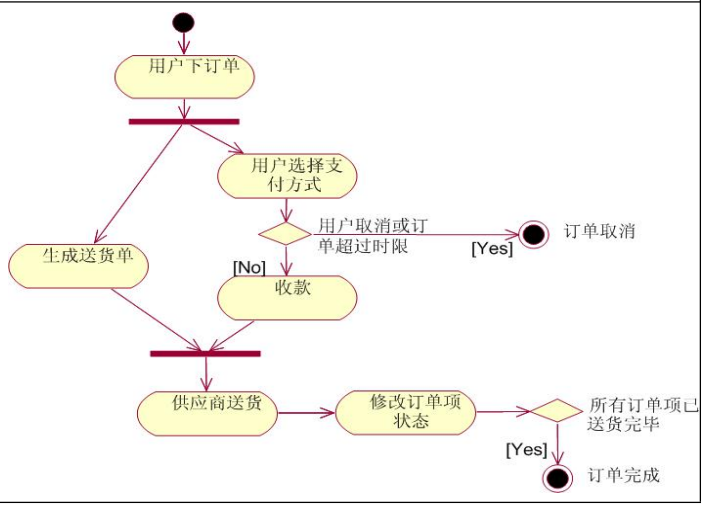
缺点：
效率低
额外软件层的正确性需要验证</td><td> 
</td></tr>
<tr><td>基于规则的系统</td><td>Code to be executed (knowledge base)
Interpretation engine (rule interpreter)
Control state of interpreter (rule/data selection)
Current state of the code (working memory)</td><td> 
</td></tr>
<tr><td rowspan="2">数据为中心的系统</td><td>仓库</td><td>Advantages
– easy to add consumers and producers of data
很容易增加数据的生产者和消费者
Issues
– synchronization（同步）
– configuration and schema management（配置和管理）
– atomicity（原子性）
– consistency（一致性）
– persistence（持久性）
– performance（性能）
构件：一个内存模块，多个纯计算进程
连接件：通过直接访问或过程调用与内存交互的计算单元
约束：适用于核心问题是发布、扩大和维护一个复杂信息中心体的应用</td><td> 
</td></tr>
<tr><td>黑板</td><td>没有直接的算法可解（多种方法都能解决问题，需要多个领域的专门知识协作解决）；
不确定性（数据和解决方法可能错误或者变化，数据中的信噪比会变化，算法借口会变化）；
没有唯一的答案，正确答案会变化。</td><td> 
</td></tr>
</table>

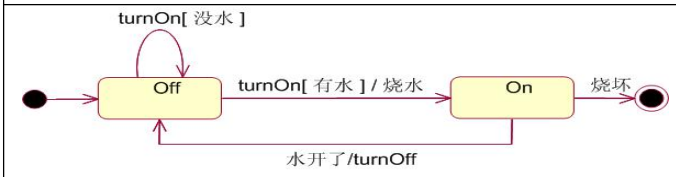
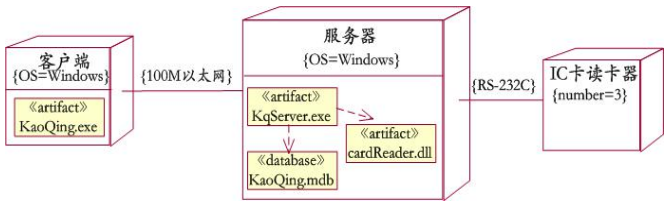
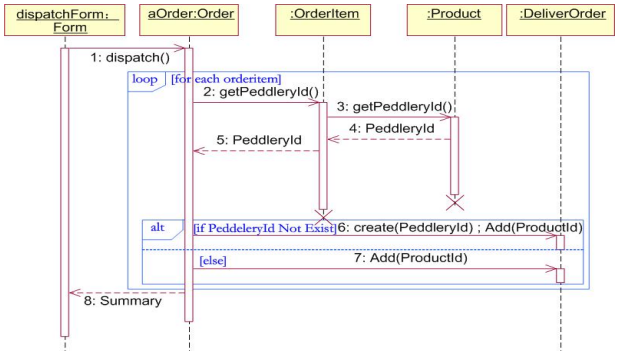
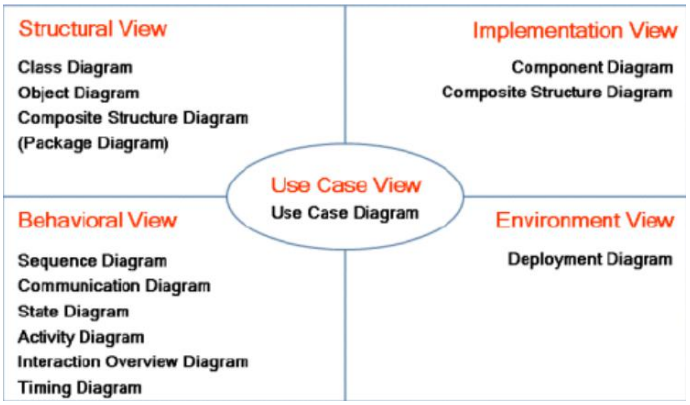
质量属性场景（Quality Attribute Scenario ）

是一个具体的质量属性需求，场景就是风险承担者与系统的交互的简短陈述。
包括六个部分
ƒ 刺激源：可以是一些实体，如人，计算机系统或者其他的刺激源
ƒ 刺激：到达系统的刺激是一个需要被考虑的情况
ƒ 环境：刺激在一定的条件下发生
ƒ 制品：某些制品被激发，可能是整个系统或者其中的一部分
ƒ 反应：反应是刺激到达后所产生的活动
ƒ 反应测量：当反应发生的时候，必须可被测量，从而需求也被检测

质量属性	意义	策略	
Usability 易用性	便于用户完成期望的事情的能力	运行时战术	a.维持任务的一个模型 b.维持用户的一个模型 c.维持系统的一个模型
		设计时战术	a.模型-视图-控制器 b.表示-抽象-控制 c.Seeheim d.Arch/Slinky
Security 安全 数据完整性，机密性	阻止非法入侵或拒绝服务的能力	抵抗攻击	a.对用户进行身份验证 b.对用户进行授权 c.维护数据的机密性 d.维护完整性 e.限制暴露的信息 f.限制访问 g.在外部用户和提供服务的系统之间设置认证服务器。 h.把要保护的系统置于通讯防火墙之后 i.在某个可信内核的基础上构建系统，由该内核提供安全
		检测攻击	
		从攻击中恢复	a.恢复状态 b.识别攻击者
Testability 可测试性	软件通过测试，易于发现错误的能力		a.提供输入捕获输出 b.内部监控
Availability 可用性 故障处理	系统能够正常运行的时间比例	错误检测	a.砰/回声 b.心跳 c.异常
		错误恢复	a.表决 b.主动冗余 c.被动冗余 d.备件 e.shadow 操作 f.状态再同步 g.检查点/回滚
		错误预防	a.进程监视器 b.从服务中删除 c.事务
Modifiability 可修改性 产品的迭代与改变	快速、低成本修改系统的能力	局部化修改	a.维持语义的一致性 b.预期期望的变更 c.泛化模块 d.限制可能的选择
		防止连锁反应	a.信息隐藏 b.维持现有的接口 添加接口 添加适配器 提供一个占位程序 A
		推迟绑定时间	a.运行时注册—支持即插即用 b.配置文件—启动时设置参数 c.多态—允许方法调用的后期绑定 d.组件更换 –允许载入时间绑定 e.遵守已定义的协议—允许独立进程的运行时绑定
Performance 性能 数据延迟，吞吐量	系统响应能力	控制对资源需求	a.减少处理一个事件所需要的资源：提高计算效率;减少计算开销 b.减少需要同时处理事件的数量：管理事件率;控制采样频率 c.控制资源的使用：限制执行时间;限制队列的大小
		资源管理	a.引入并发 b.维持数据或计算的多个副本 c.增加可用资源
		资源仲裁	a.先进/先出 b.固定优先级 Δ 语义重要性Δ 时限时间单调Δ 速率单调 c.动态优先级调度Δ 轮转 Δ 时限时间最早优先 d.静态调度

UML

类图：系统静态结构，backbone	对象图：运行时刻状态 snapshot
	
包图：package，higher levelunit	用例图：external view，功能的描述，外部视角
	
构件图：物理结构的依赖	通信图：communication 协作，联系
	
顺序图：协作关系，多个对象 1 个用例，随着时间的推移，对象之间是如何交互	活动图：活动约束，识别并行活动
	

<table>
<tr><td>状态图：状态+转移条件</td><td>部署图；软硬件体系结构</td></tr>
<tr><td> 
</td><td> 
</td></tr>
<tr><td>交互图：grafting together of activity diagrams and sequence diagrams</td><td>4+1 视图/分类图从 5 个不同视角来描述软件体系结构</td></tr>
<tr><td> 
</td><td> 
</td></tr>
</table>

效用树 utility tree

效用 – 质量属性 – 刺激 –（权重，风险） 响应

可用性：遗漏，崩溃，超时，无/错响应

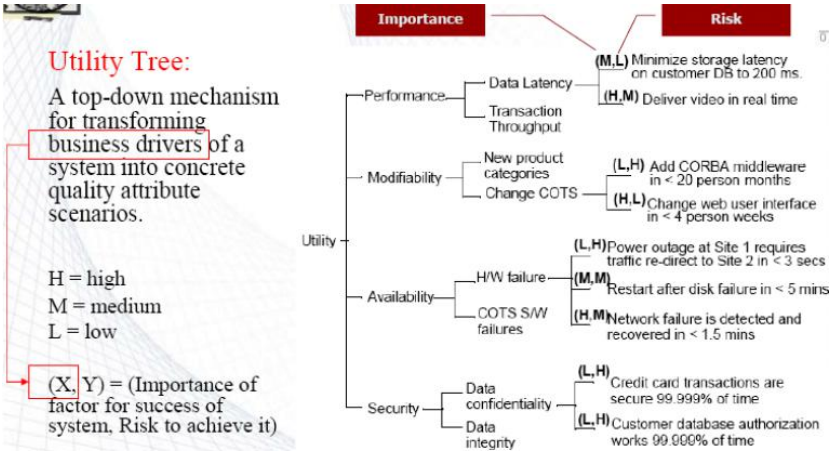
可修改性：希望添加/删除/修改/变更功能,质量属性,容量

性能：周期事件到达，偶发事件到达，随机事件到达

安全：试图显示数据，改变/删除数据，介入系统服务，降低系统服务的可用性

可测试性：分析，构建，设计，类，子系统融合完成；系统呈交

易用性：希望学习系统特性；有效使用系统；最小化错误的影响；适应系统；用着舒服。



评估 evaluation

问法：题中描述的对应哪个：

四个标准的定义

风险（risks）：可能在将来损害某些质量属性的方案

非风险（non-risks）：可以提高质量、帮助实现目标的决策

关键点/敏感点（sensitivity points）：方案中一个小小的变化，就可能对某些质量属性有很大影响

权衡点（tradeoffs）：影响一个以上质量的决策

质量属性

Source	刺激源，可以产生刺激的实体
Stimulus	刺激，一个可以使系统有所动作和反应的情况
Artifact	制品，由刺激产生
Environment	环境，可以满足刺激发生的条件的环境
Response	反应，刺激产生的活动
Response measure	反应测量，反应发生的时候，必须可被测量

Usability（易用性）　　用户完成预期功能的简单程度和系统提供的用户支持种类
1.终端用户
2.用户想要学习如何使用该软件，使用很高效，对错误影响的最小化，对系统的适应，对舒适度的追求
3.系统
4.运行时和设置
5.提供方便快捷并预判用户需求
6.任务执行时间、错误数、问题解决数、用户满意程度、用户的收益、操作成功率、　错误数

Security（安全）　　在提供正常服务的同时对非法操作的抵御能力
特点：机密性，不可抵赖性，合乎逻辑性(Integrity)，真实正确性(Assurance)，正常可用性，可记录审查性
1.hacker或某系统
2.攻击或任何对系统造成威胁的行为
3.系统内数据或提供的服务
4.在线或不在线，在网或不再网，防火墙开启或未开启
5.对合法用户提供正常访问权限，抵御不合法用户并记录、报告该行为
6.抵御攻击和系统恢复的难度

Testability（可测试性）　　可以通过测试发现并证明错误的轻松程度
1.单元测试人员、集成测试人员、系统测试人员、用户；设计测试有其他开发者或其他小组
2.开发里程碑或某部分、单元的完成
3.设计、代码段、系统
4.开发时、设计时，编译时、运行时
5.系统在测试时能按照测试设计的正常运行，测试能捕获系统的响应
6.可执行测试的比例、最长测试链长（测试难度相关）、捕获错误的几率

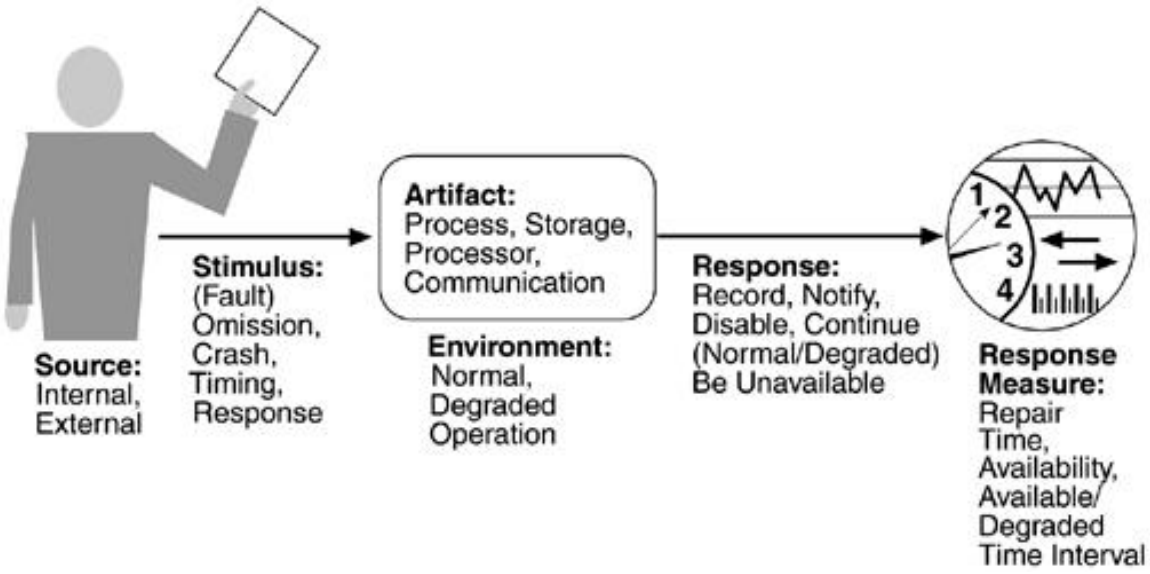
Availability（可用性）　　计算机在任意时刻正常工作的概率
1.系统内外的错误潜在标志
2.错误：输入缺漏，崩溃，超时，错误响应
3.高可靠性资源：CPU，内存，信道，进程
4.当前的运行环境，正常状态，低级模式
5.记录错误日志，通知用户和系统，开启低级模式保持基本运转，关闭扩展系统，当机维护
6.可用性百分比，修复时间

Modifiability（可修改性）　　系统易被修改的能力，以适应新的需求，采用新的算法、数据结构的能力。
1.有意图做出修改的实体，开发人员，管理员，用户
2.改变项
3.被改变的部分，功能，平台，接口，交互系统等
4.系统可被修改的状态，开发时，编译时，建立时，初始化时，运行时
5.修改者必须对其修改有足够的理解，修改、测试并部署之

6.花费的时间和金钱

Performance（性能）　　系统响应时间，硬件资源占用率
1.系统内部或外部
2.事件的产生，产生模式有：周期性的，随机性的，偶然性的
3.系统服务
4.正常状态，紧急状态，过载状态等
5.系统对事件的处理，有可能会因此切换状态
6.处理时常，系统变动情况，处理频度，异常种类

Availability general scenarios



Modifiability General Scenario

Portion of Scenario	Possible Values
Source	End user, developer, system administrator
Stimulus	Wishes to add/delete/modify/vary functionality, quality attribute, capacity
Artifact	System user interface, platform, environment; system that interoperates with target system
Environment	At runtime, compile time, build time, design time
Response	Locates places in architecture to be modified; makes modification without affecting other functionality; tests modification; deploys modification
Response Measure	Cost in terms of number of elements affected, effort, money; extent to which this affects other functions or quality attributes

Performance General Scenarios

Portion of Scenario	Possible Values

Source	One of a number of independent sources, possibly from within system
Stimulus	Periodic events arrive; sporadic events arrive; stochastic events arrive
Artifact	System
Environment	Normal mode; overload mode
Response	Processes stimuli; changes level of service
Response Measure	Latency, deadline, throughput, jitter, miss rate, data loss

Security General Scenarios

Portion of Scenario	Possible Values
Source	Individual or system that is correctly identified, identified incorrectly, of unknown identity who is internal/external, authorized/not authorized with access to limited resources, vast resources
Stimulus	Tries to display data, change/delete data, access system services, reduce availability to system services
Artifact	System services; data within system
Environment	Either online or offline, connected or disconnected, firewalled or open
Response	Authenticates user; hides identity of the user; blocks or allow access to data and/or services; grants or withdraws permission to access data and/or services; records access/modifications or attempts to access/modify data/services by identity; stores data in an unreadable format; recognizes an unexplainable high demand for services, and informs a user or another system, and restricts availability of services
Response Measure	Time/effort/resources required to circumvent security measures with probability of success; probability of detecting attack; probability of identifying individual responsible for attack or access/modification of data and/or services; percentage of services still available under denial-of-services attack; restore data/services; extent to which data/services damaged and/or legitimate access denied

Testability General Scenarios

Portion of Scenario	Possible Values

Source	Unit developer, Increment integrator System verifier, Client acceptance tester System user
Stimulus	Analysis, architecture, design, class, subsystem integration completed; system delivered
Artifact	Piece of design, piece of code, complete application
Environment	At design time, at development time, at compile time, at deployment time
Response	Provides access to state values; provides computed values; prepares test environment
Response Measure	Percent executable statements executed Probability of failure if fault exists Time to perform tests Length of longest dependency chain in a test Length of time to prepare test environment

Usability General Scenarios

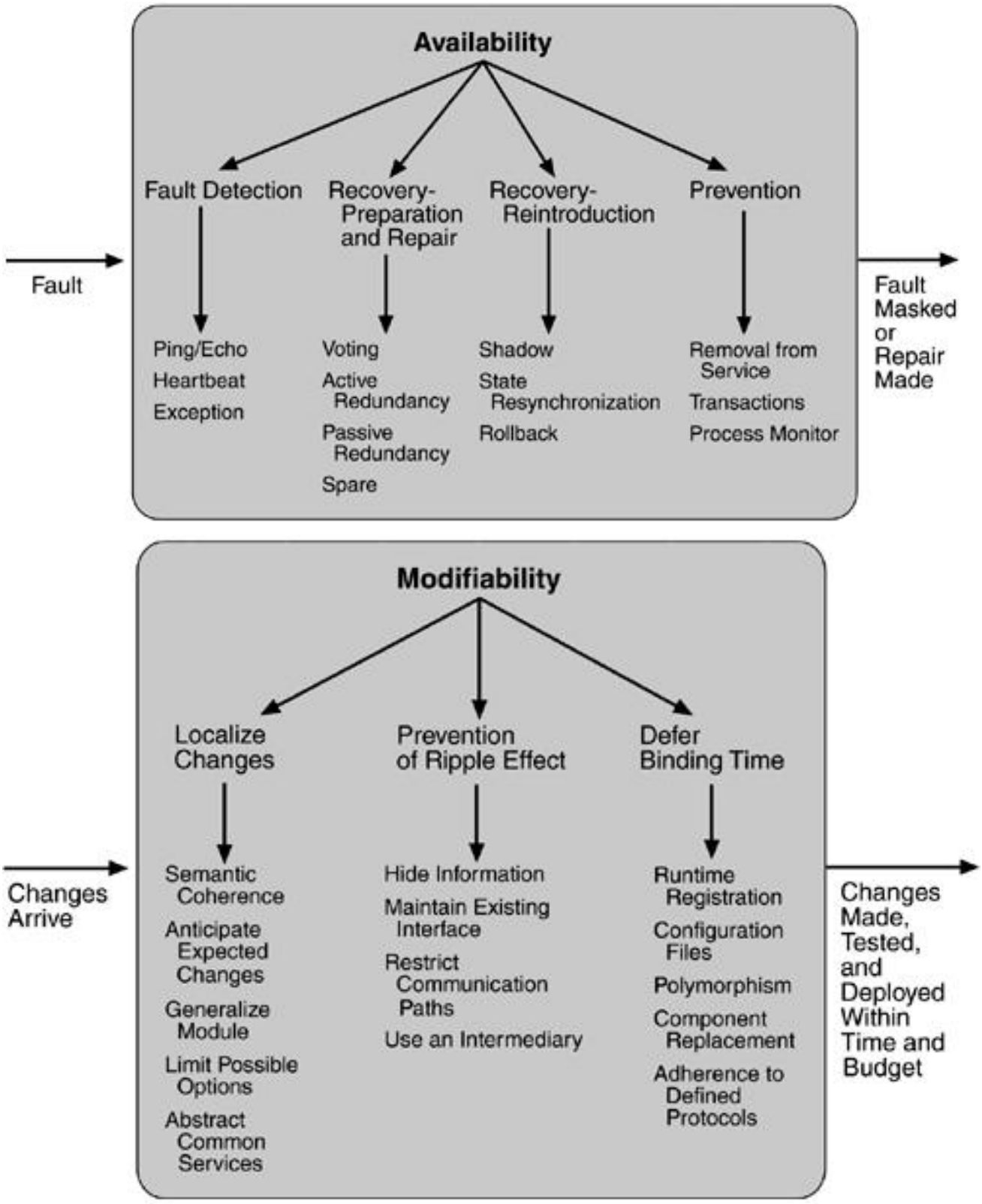
Portion of Scenario	**Possible Values**
Source	End user
Stimulus	Wants to， learn system features; use system efficiently; minimize impact of errors; adapt system; feel comfortable
Artifact	System
Environment	At runtime or configure time
Response	to support "learn system features": help system is sensitive to context; interface is familiar to user; interface is usable in an unfamiliar context to support "use system efficiently": aggregation of data and/or commands; reuse of already entered data and/or commands; support for efficient navigation within a screen; distinct views with consistent operations; comprehensive searching; multiple simultaneous activities to "minimize impact of errors": undo, cancel, recover from system failure, recognize and correct user error, retrieve forgotten password, verify system resources to "adapt system": customizability; internationalization to "feel comfortable":display system state; work at the user's pace
Response Measure	Task time, number of errors, number of problems solved, user satisfaction, gain of user knowledge, ratio of successful operations to total operations, amount of time/data lost

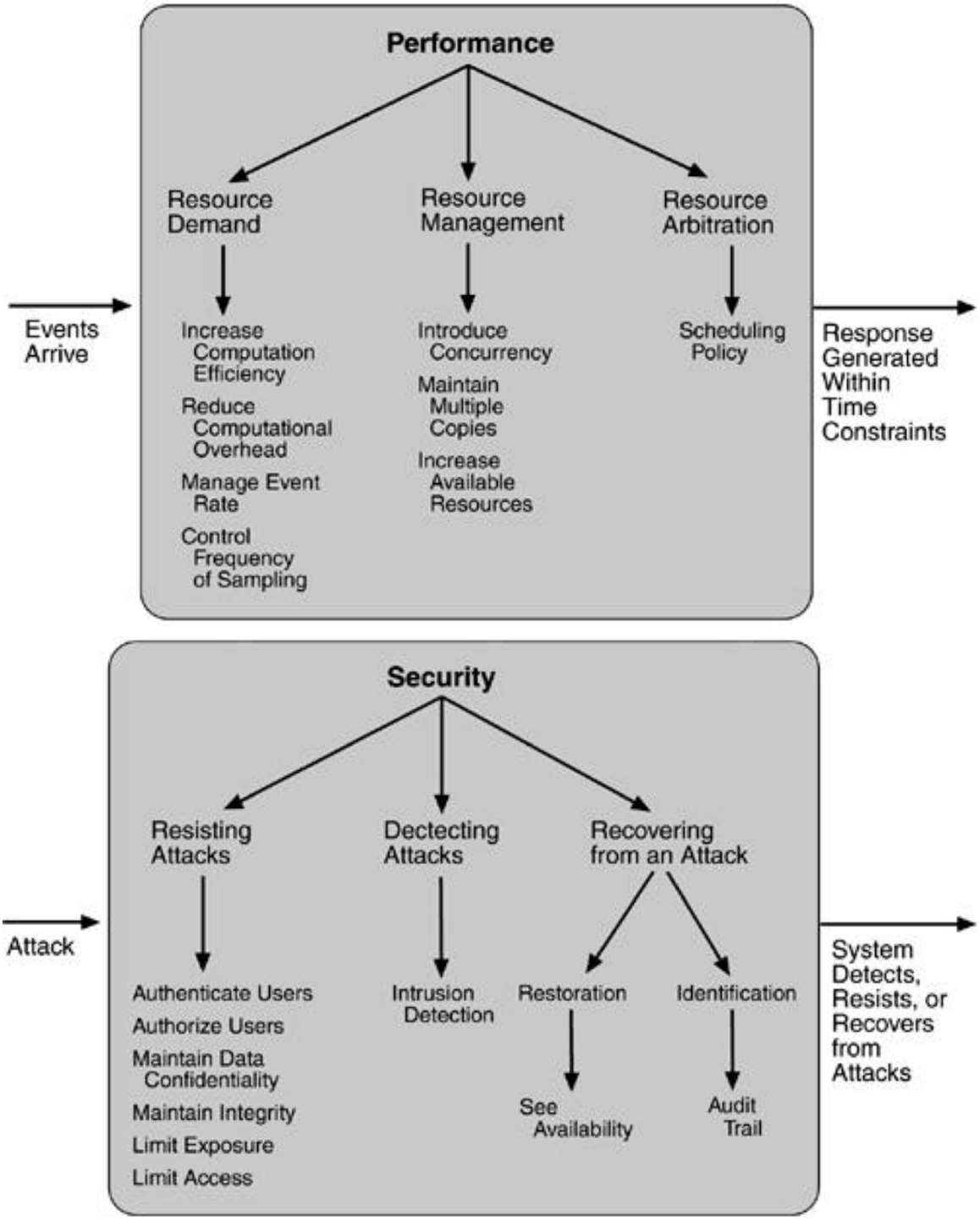
Quality Attribute Stimuli

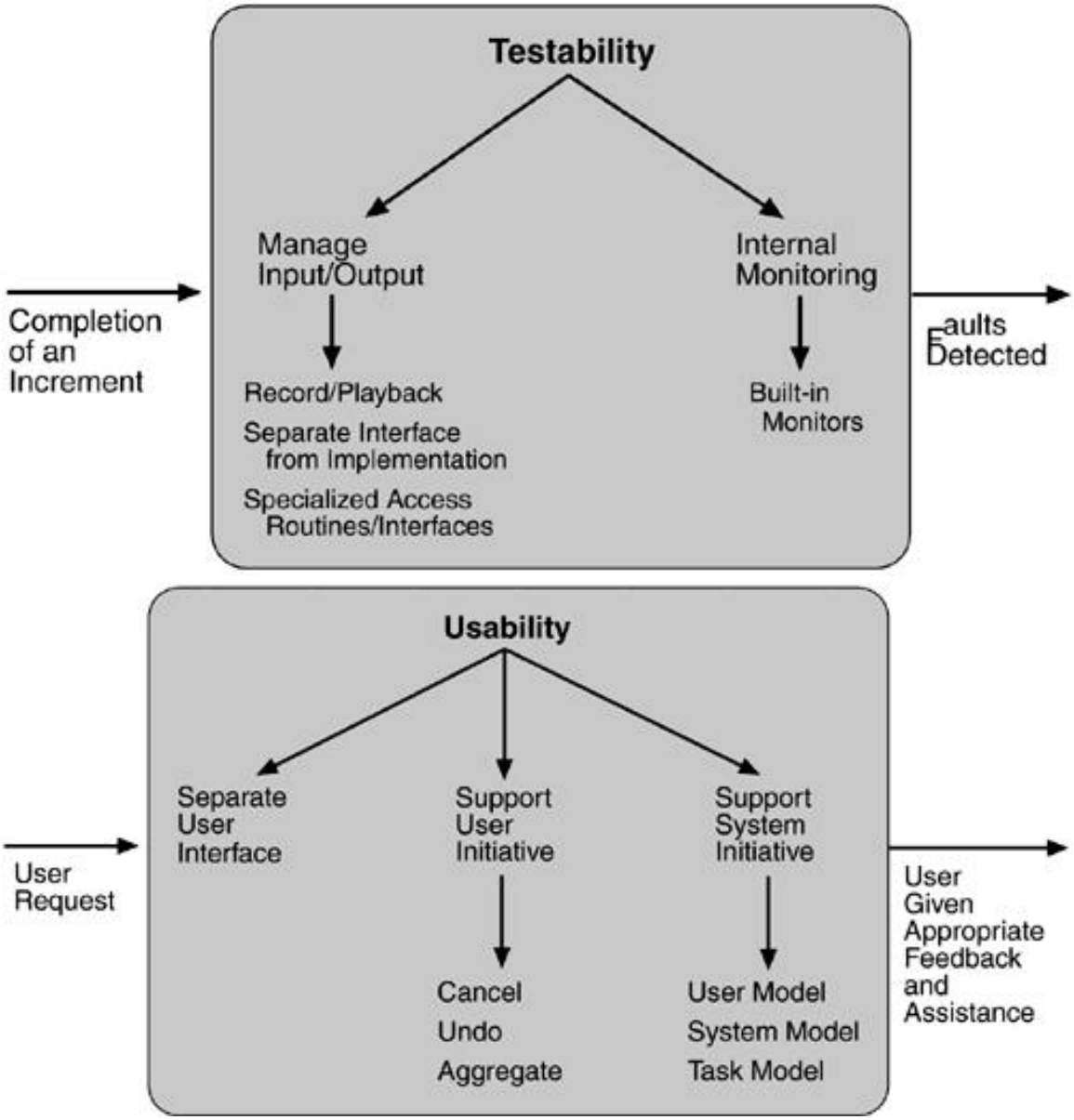
Quality Attribute	Stimulus
Availability	Unexpected event, nonoccurrence of expected event
Modifiability	Request to add/delete/change/vary functionality, platform, quality attribute, or capacity
Performance	Periodic, stochastic, or sporadic
Security	Tries to display, modify, change/delete information, access, or reduce availability to system services
Testability	Completion of phase of system development
Usability	Wants to learn system features, use a system efficiently, minimize the impact of errors, adapt the system, feel comfortable

Other

Quality Attribute	Stimulus
Availability	Unexpected event, nonoccurrence of expected event
Modifiability	Request to add/delete/change/vary functionality, platform, quality attribute, or capacity
Performance	Periodic, stochastic, or sporadic
Security	Tries to display, modify, change/delete information, access, or reduce availability to system services
Testability	Completion of phase of system development
Usability	Wants to learn system features, use a system efficiently, minimize the impact of errors, adapt the system, feel comfortable


Availability
Fault Detection
Recovery-Preparation and Repair
Recovery-Reintroduction
Prevention
Fault
Ping/Echo
Heartbeat
Exception
Voting
Active Redundancy
Passive Redundancy
Spare
Shadow
State Resynchronization
Rollback
Removal from Service
Transactions
Process Monitor
Fault Masked or Repair Made
Modifiability
Localize Changes
Prevention of Ripple Effect
Defer Binding Time
Changes Arrive
Semantic Coherence
Anticipate Expected Changes
Generalize Module
Limit Possible Options
Abstract Common Services
Hide Information
Maintain Existing Interface
Restrict Communication Paths
Use an Intermediary
Runtime Registration
Configuration Files
Polymorphism
Component Replacement
Adherence to Defined Protocols
Changes Made, Tested, and Deployed Within Time and Budget


Performance
Resource Demand
Resource Management
Resource Arbitration
Events Arrive
Increase Computation Efficiency
Reduce Computational Overhead
Manage Event Rate
Control Frequency of Sampling
Introduce Concurrency
Maintain Multiple Copies
Increase Available Resources
Scheduling Policy
Response Generated Within Time Constraints
Security
Resisting Attacks
Dectecting Attacks
Recovering from an Attack
Attack
Authenticate Users
Authorize Users
Maintain Data Confidentiality
Maintain Integrity
Limit Exposure
Limit Access
Intrusion Detection
Restoration
Identification
See Availability
Audit Trail
System Detects, Resists, or Recovers from Attacks


Testability
Manage Input/Output
Internal Monitoring
Completion of an Increment
Record/Playback
Separate Interface from Implementation
Specialized Access Routines/Interfaces
Built-in Monitors
aults Detected
Usability
Separate User Interface
Support User Initiative
Support System Initiative
User Request
Cancel
Undo
Aggregate
User Model
System Model
Task Model
User Given Appropriate Feedback and Assistance