Quality Attributes of Software Architecture

Ergude Bao

Beijing Jiaotong University

Contents

- Introduction to Quality Attribute
- Quality Attribute Scenario
- Typical Quality Attributes
  - Performance Attribute
  - Scalability Attribute
  - Modifiability Attribute
  - Security Attribute
  - Availability Attribute
  - Usability Attribute
- Design Trade-Off

Introduction to Quality attribute

Quality Attributes

- Performance
- Security
- Availability
- Scalability
- Usability
- Reliability
- Portability
- Modifiability
- Maintainability

Function of Quality Attributes

- Achieving quality attributes must be considered throughout design, implementation, and deployment
  - Usability
    - Providing the user with undo operations (architectural)
    - Making the user interface easy to use (non-architectural)
  - Modifiability
    - How functionality is divided (architectural)
    - By coding techniques within a module (non-architectural)
  - Performance
    - How much communication is necessary among components and how shared resources are allocated (architectural)
    - The choice of algorithms and how they are coded (non-architectural)

Specification of Quality Attributes

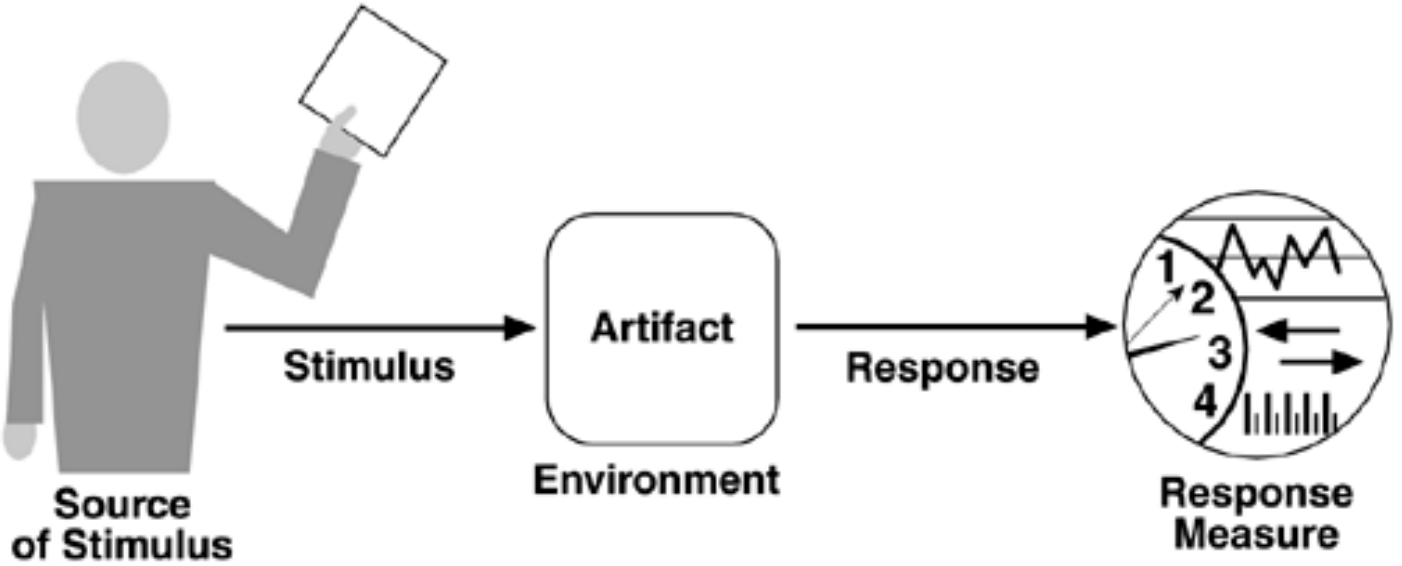
- Architects are often told in a far too imprecise manner
  - “My application must be fast/secure/scale”
- Quality attributes (QAs) must be made precise/measurable for a given system design
  - “It must be possible to scale the deployment from an initial 100 geographically dispersed user desktops to 10,000 without an increase in effort/cost for installation and configuration”

Quality Attribute Scenario

Quality Attribute Scenario

- A quality attribute scenario is a quality attribute specific requirement
  - Source of stimulus – the entity that generated the stimulus
  - Stimulus – a condition that needs to be considered when it arrives at a system
  - Environment – the particular conditions in which the stimulus occurs
  - Artifact – the system or the pieces of it that are stimulated
  - Response – the activity undertaken after the arrival of the stimulus
  - Response measure – when the response occurs, it should be measurable in some fashion so that the requirement can be tested

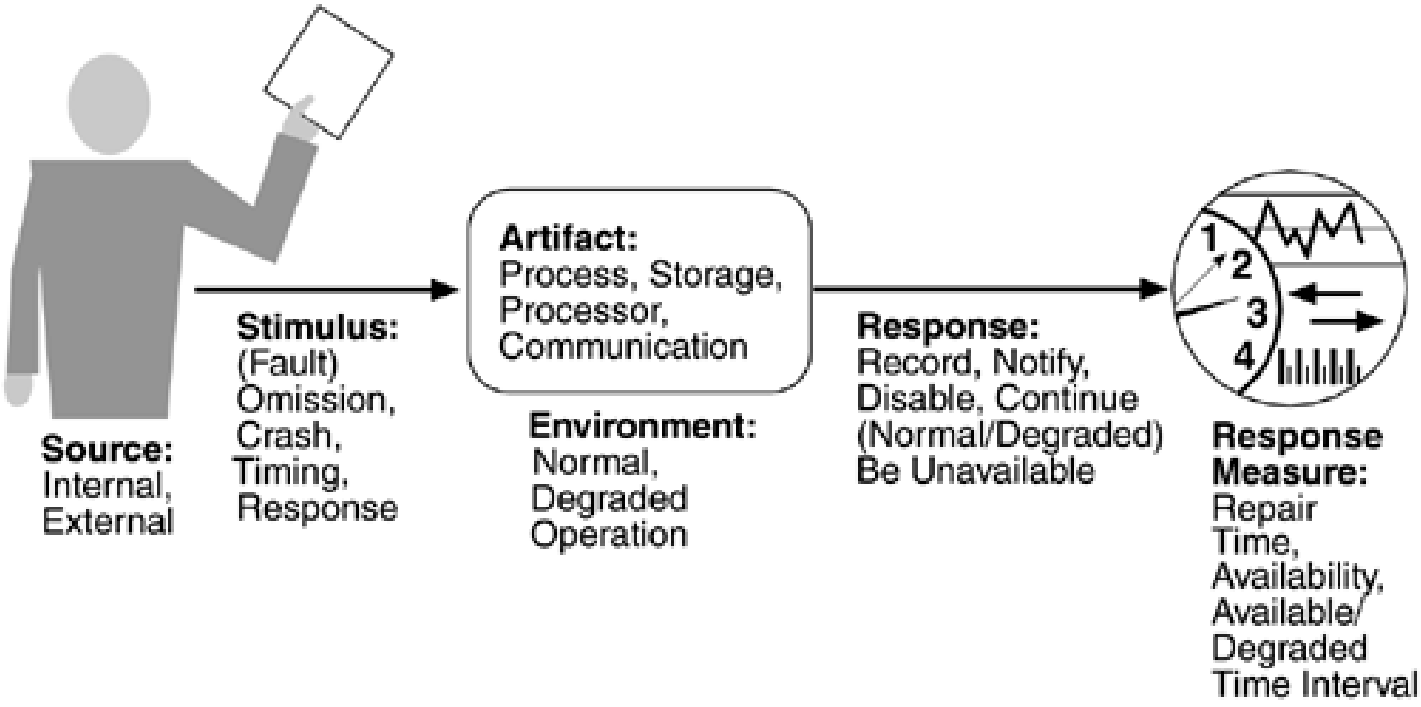
Quality Attribute Scenario



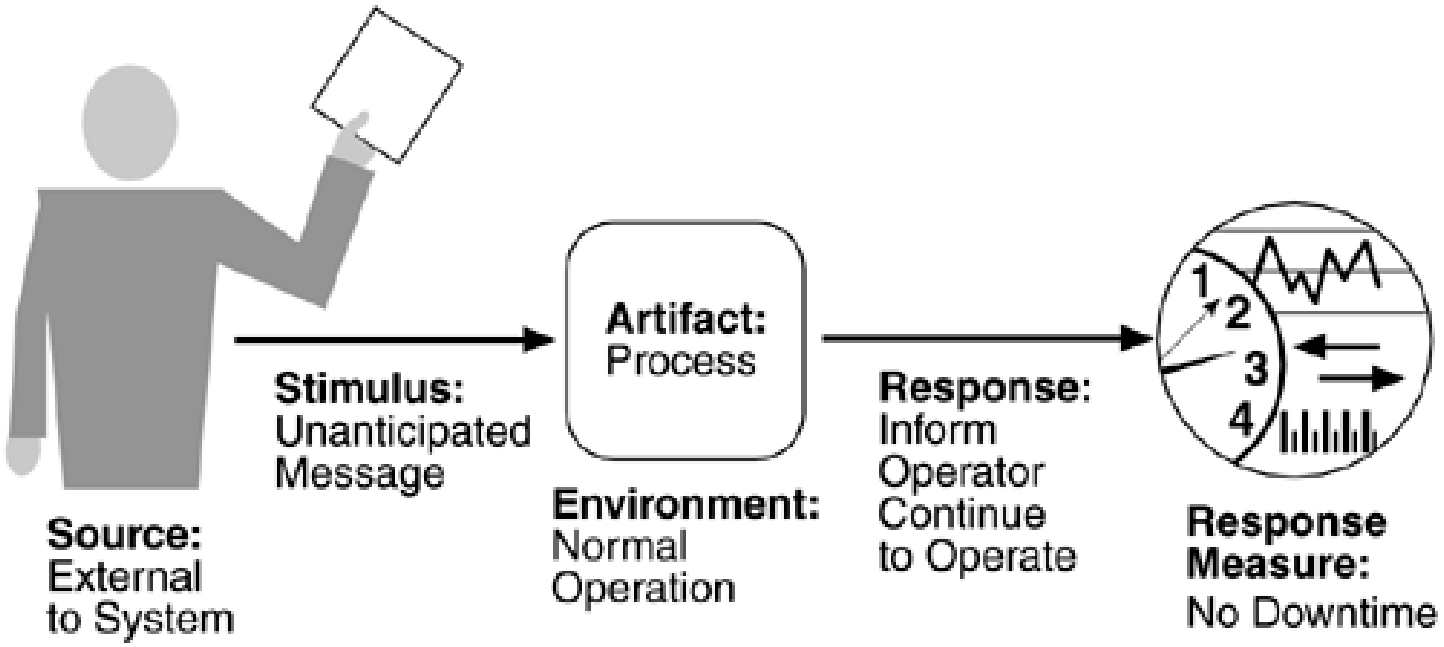
General vs. Concrete Quality Attribute Scenarios

- A general scenario is system independent and can, potentially, pertain to any system
- A concrete scenario is specific to the particular system under consideration
  - Concrete scenarios are needed to make the quality requirements operational
- A concrete scenario is a subset of a general scenario

General Availability Scenario



Concrete Availability Scenario



Function of Concrete Scenarios

- A collection of concrete scenarios can be used as the quality attribute requirements of a system
  - Concrete scenarios play the same role in the specification of quality attributes that use cases play in the specification of functional requirements
- Details of the responses are meaningful enough so that it is possible to test whether the system has achieved the response

Generation of Concrete Scenarios

- Theoretically quality attribute requirements should be obtained during requirement analysis, but in practice is seldom done
- It is the architect's task to ensure that this is accomplished by generating concrete quality attribute scenarios

Generation of Concrete Scenarios

- Quality-attribute-specific tables are used to create general scenarios and from them concrete scenarios are specified
  - The tables serve as checklist to ensure that all possibilities have been considered
- It does not matter to generate the same or similar scenarios from different quality attributes as the redundancies can easily be removed
  - The important is that no important requirements are omitted

Table of General Availability Scenario

Portion of Scenario	Possible Values
Source	Internal to the system; external to the system
Stimulus	Fault; omission, crash, timing, response
Artifact	System's processors, communication channels, persistent storage processes
Environment	Normal operation; degraded mode (i.e., fewer features, a fall-back solution.)
Response	System should detect event and do one or more of the following: record it notify appropriate parties, including the user and other systems disable sources of events that cause fault or failure be unavailable for a prespecified interval continue to operate in normal or degraded mode
Response Measure	Time interval when the system must be available; availability time; time interval in which system can be in degraded mode; repair time

Typical Quality Attributes Performance

Performance

- Performance requires
  - Metric of amount of work performed in unit time
  - Deadline that must be met
  - And so on
- Solutions
  - Localize critical operations and minimize communications
  - Use coarse rather than fine-grained components
  - And so on

Performance

- Enterprise applications often have strict performance requirements, e.g.
  - 1000 transactions per second
  - 3 second average latency for a request
- Many examples of poor performance in enterprise applications

Throughput

- Measure of the amount of work an application must perform in unit time
  - Transactions per second
  - Messages per minute
- Is required throughput:
  - Average?
  - Peak?

Response Time

- Measure of the latency an application exhibits in processing a request
  - Usually measured in seconds (or mili-seconds)
- Is required response time:
  - Guaranteed?
  - Average?

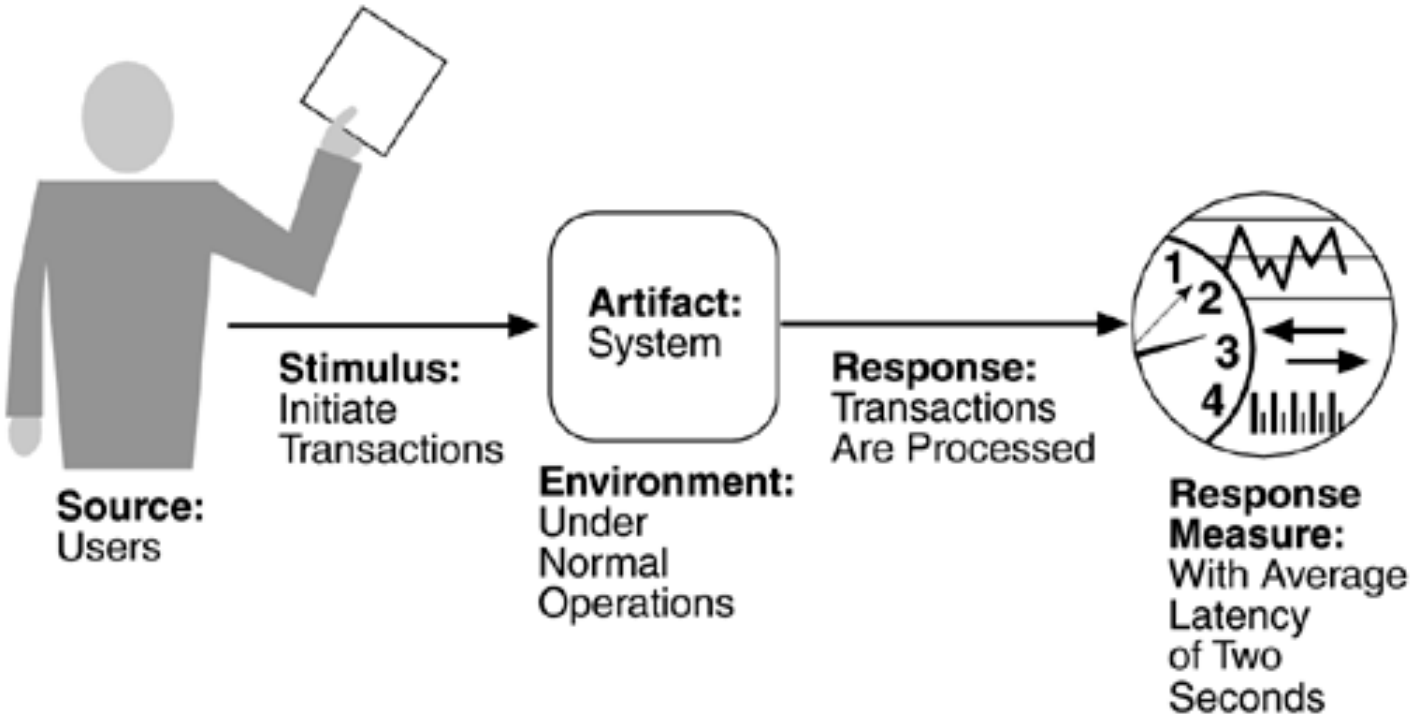
Deadline

- Something must be completed before some specified time
  - Payroll system must complete by 2 am so that electronic transfers can be sent to bank
  - Weekly accounting run must complete by 6 am Monday so that figures are available to management
- Deadlines are often associated with batch jobs in IT systems

Table of General Performance Scenario

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, miss rate, data loss

Concrete Performance Scenario



Typical Quality Attributes
Scalability

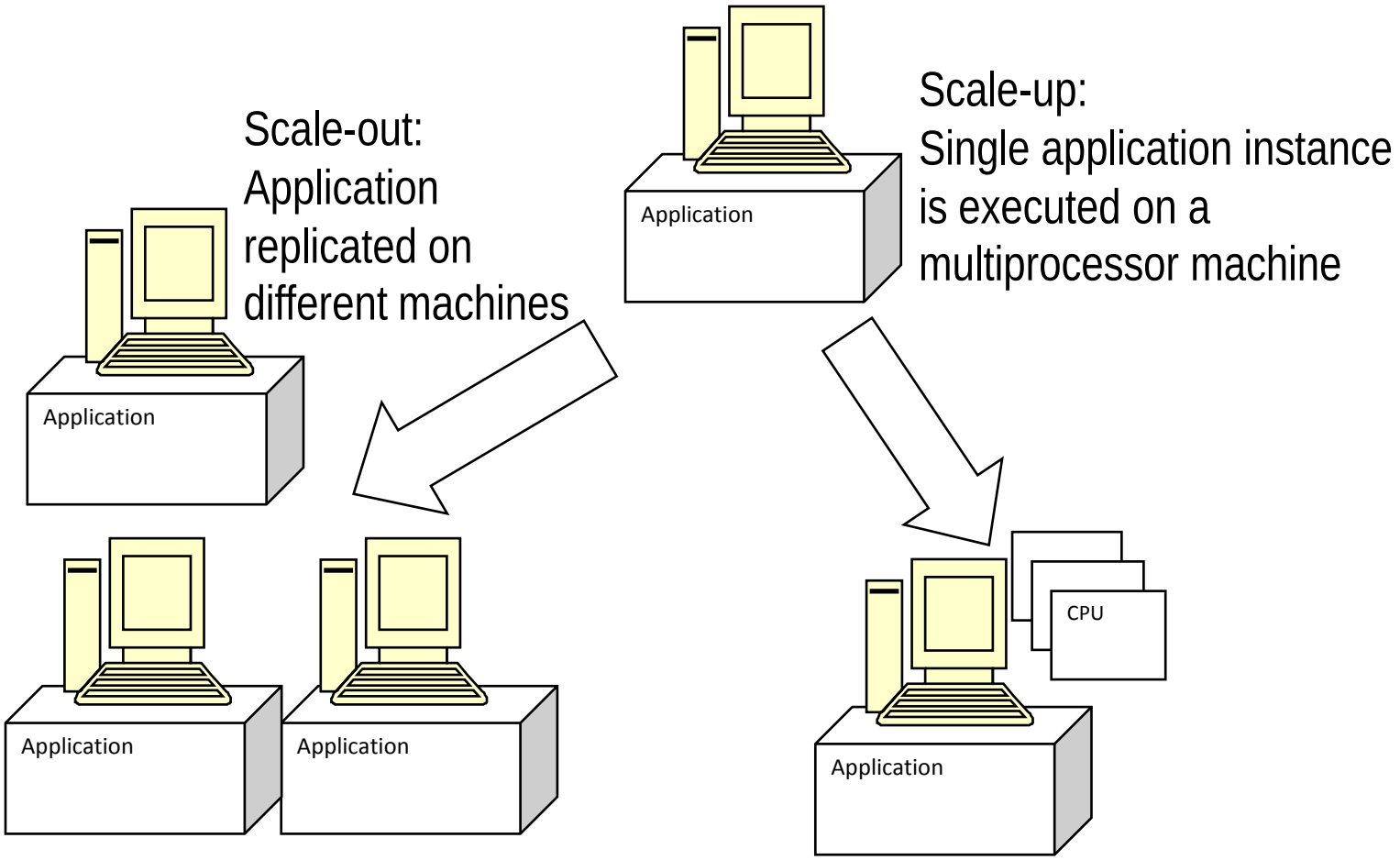
Scalability

- Scalability is a desirable property of a system, a network, or a process, which indicates its ability to either handle growing amounts of work in a graceful manner or to be readily enlarged
- 4 common scalability issues in IT systems
  - Request load
  - Connections
  - Data size
  - Deployments

Request Load

- How does a 100 rps application behave when simultaneous request load grows?
  - From 100 to 1000 requests per second?
- Ideal solution is as the load increases, throughput remains constant (i.e. 100 rps), and response time per request increases only linearly (i.e. 10 seconds)

Another solution



Connections

- What happens if number of simultaneous connections to an application increases
  - If each connection consumes a resource?
  - If the number of connections exceeds the maximum number?

Data Size

- How does an application behave as the data it processes increases in size? Can application/algorithms scale to handle increased data requirements?
  - Chat application sees average message size double?
  - Database table size grows from 1 million to 20 million rows?
  - Image analysis algorithm processes images of 100MB instead of 1MB?

Deployment

- How does effort to install/deploy an application increase as installation base grows?
  - Install new users?
  - Install new servers?
- Solutions typically revolve around automatic download/installation
  - E.g. downloading applications from the Internet

Scalability

- Often overlooked, since not major cause of application failure
- Hard to predict and hard to test/validate
- Reliance on proven designs and technologies is essential

Typical Quality Attributes Modifiability

Modifiability

- Modifiability measures how easy it may be to change an application to cater for new (non-) functional requirements
  - 'May' – nearly always impossible to be certain
- Modifiability is about the cost of change
  - Must estimate cost/effort
- Modifiability should be assessed in context of how a system is likely to change
  - No need to consider changes that are highly unlikely to occur

Strategies

- Strategy 1: Think beforehand
  - Convincing possibility and impact analysis of changes
  - A demonstration of how the corresponding solution to accommodate the modification
- Strategy 2: Minimize dependencies
  - Changes isolated to single components likely to be less expensive than those that cause ripple effects across the architecture

Typical Quality Attributes
Security

Security

- Security is a measure of the system's ability to resist unauthorized usage while providing its services to legitimate users
- An attempt to breach security is an attack – it could be to gain access to data or services or to deny service to others

Strategies

- Authentication: applications can verify the identity of their users and other applications with which they communicate
- Authorization: authenticated users and applications have defined access rights to the resources of the system
- Encryption: messages sent to/from the application are encrypted

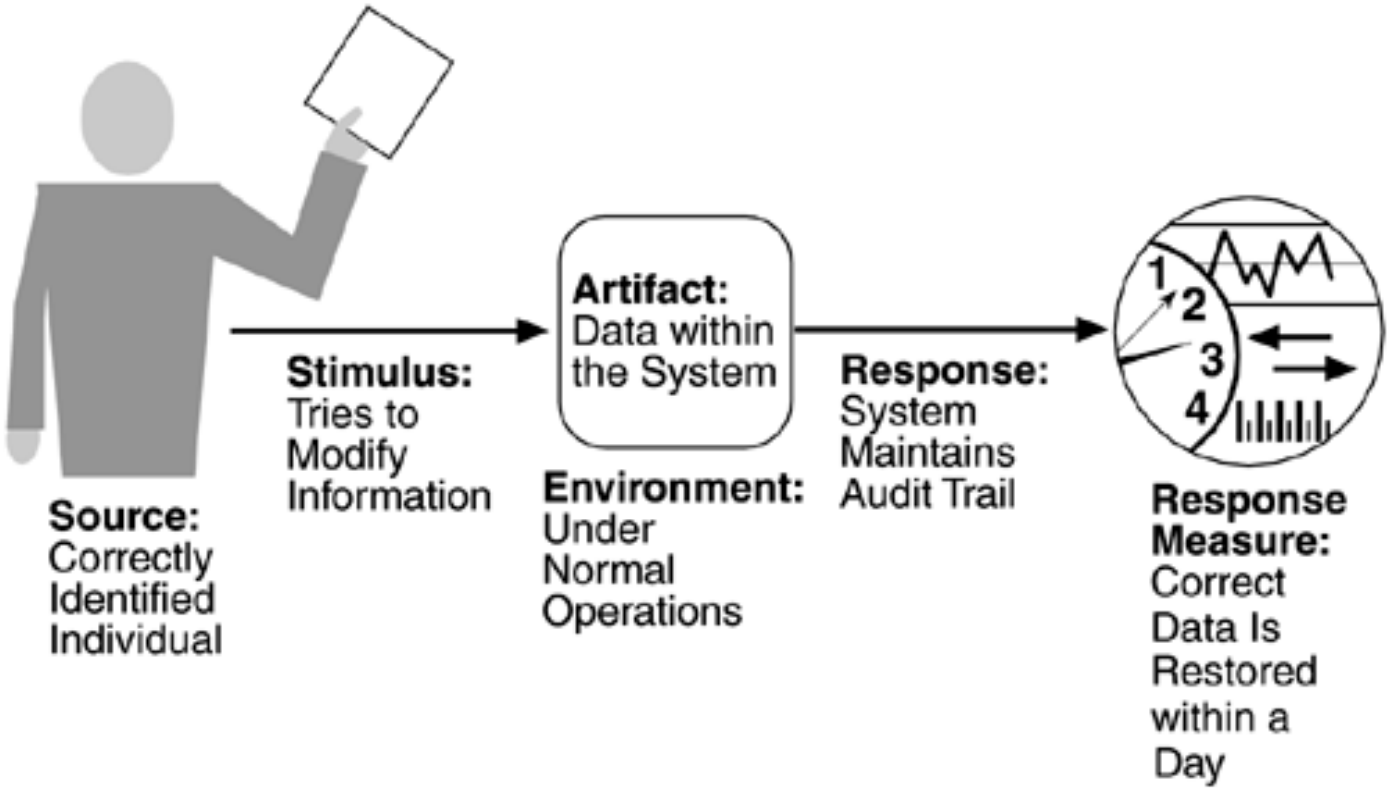
Strategies

- Integrity: contents of a message are not altered in transit
- Non-repudiation: sender of a message has proof of delivery and the receiver is assured of the sender's identity
  - Neither can subsequently refute their participation in the message exchange, i.e. a transaction cannot be denied by any of the parties to it
- Auditing: system tracks activities within it at levels sufficient to reconstruct them

Table of General Security Scenario

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 access to data and/or services; allows 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

Concrete Security Scenario



Typical Quality Attributes
Availability

Availability

- Availability is concerned with system failure and its consequences
  - Key requirement for most IT applications
  - How the system failure is detected
  - How frequently system failures occur
  - What happens when a failure occurs
  - How long a system is allowed to be out of operation
  - When failures may occur safely
  - How failures can be prevented
  - What kinds of notifications are required when a failure occurs

Availability

- Measured by the proportion of the required time it is useable
  - 100% available during business hours
  - No more than 2 hours scheduled downtime per week
  - 24 x 7 x 52 (100% availability)
- Period of loss of availability determined by
  - Time to detect failure
  - Time to correct failure
  - Time to restart application
- Related to an application's reliability and recoverability
  - Unreliable applications suffer poor availability
  - Capability to recover affects data after a system failure

Strategies

- Eliminate single points of failure
- Replication and failover
- Automatic detection and restart

Table of General Availability Scenario

Portion of Scenario	Possible Values
Source	Internal to the system; external to the system
Stimulus	Fault; omission, crash, timing, response
Artifact	System's processors, communication channels, persistent storage processes
Environment	Normal operation; degraded mode (i.e., fewer features, a fall-back solution.)
Response	System should detect event and do one or more of the following: record it notify appropriate parties, including the user and other systems disable sources of events that cause fault or failure be unavailable for a pre-specified interval continue to operate in normal or degraded mode
Response Measure	Time interval when the system must be available; availability time; time interval in which system can be in degraded mode; repair time

Typical Quality Attributes
Usability

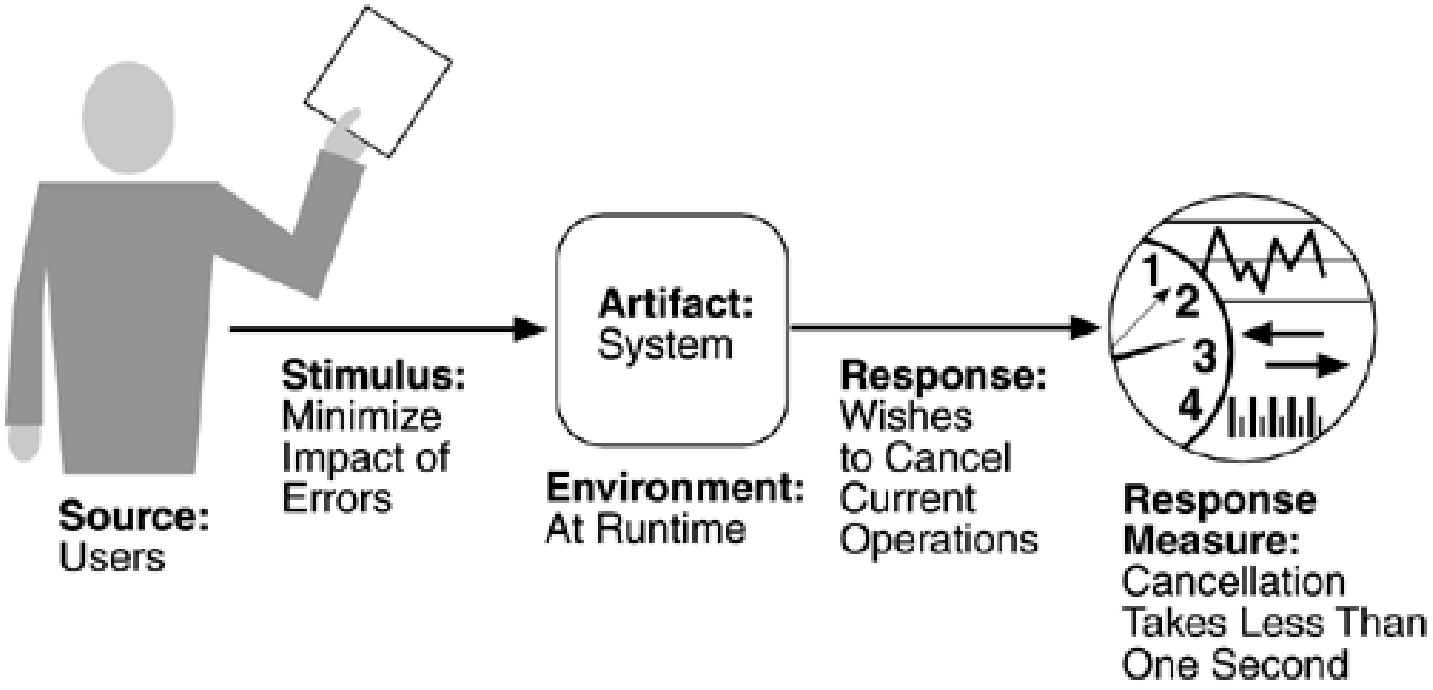
Usability

- Usability is concerned with how easy it is for the user to accomplish a desired task and the kind of user support provided
  - Learning system features
  - Using a system efficiently
  - Minimizing the impact of errors
  - Adapting the system to user needs
  - Increasing confidence and satisfaction

Table of General Usability Scenario

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 configuration time
Response	System provides one or more of the following responses: 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; re-use 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

Concrete Usability Scenario



Design Trade-Off

Design Trade-Off

- Quality attributes are rarely orthogonal, they interact and affect each other
  - Using coarse-grained components improves performance but reduces maintainability
  - Introducing redundant data improves availability but makes security more difficult
  - Localizing safety-related features usually means more communication so degrading performance

Design Trade-Off

- Architects must create solutions that makes sensible design compromises
  - Not possible to fully satisfy all competing requirements
  - Must satisfy all stakeholder needs