Understanding Quality Attributes

Outline

- Quality attributes
- Architecture and quality attributes
- Quality attribute scenarios
- Achieving quality attributes

Quality Attributes

- **Performance**: the response time, utilization, and throughput behavior of the system.
- **Security**: a measure of system's ability to resist unauthorized attempts at usage or behavior modification, while still providing service to legitimate users.
- **Availability:** the measure of time that the system is up and running correctly; the length of time between failures and the length of time needed to resume operation after a failure.
- **Usability:** the ease of use and of training the end users of the system.
- **Interoperability:** the ability of two or more systems to cooperate at runtime

Quality Attributes

- **Modifiability**: the ease with which a software system can accommodate changes to its software
- **Reusability**: the degree to which existing applications can be reused in new applications
- **Testability**: the ease with which software can be made to demonstrate its faults

Quality Attributes

- **Cost and Schedule**: the cost of the system with respect to time to market, expected project lifetime
- **Marketability**: the use of the system with respect to market competition
- **Appropriateness for Organization**: availability of the human input, allocation of expertise, and alignment of team and software structure

Architecture and Quality Attributes

- Achieving quality attributes must be considered throughout design, implementation, and deployment.
- Satisfactory results are a matter of getting the big picture (architecture) as well as the details (implementation) correct
- For example:
  - Usability involves both architectural and nonarchitectural aspects. Making the user interface easy to use is nonarchitectural, but providing the user with undo operations is architectural.

Architecture and Quality Attributes

- Modifiability is determined by how functionality is divided (architectural) and by coding techniques within a module (nonarchitectural).
- Performance depends partially on how much communication is necessary among components and how shared resources are allocated (architectural) and partially on the choice of algorithms and how they are coded (nonarchitectural)

Architecture and Quality Attributes

- The message is:
  1. Architecture is critical to the realization of many qualities of interest in a system, and these qualities should be designed in and can be evaluated at the architectural level.
  2. Architecture, by itself, is unable to achieve qualities. It provides the foundation for achieving quality, but this foundation will be to no avail if attention is not paid to the details.

Architecture and Quality Attributes

- Within complex systems, quality attributes can never be achieved in isolation.
- The achievement of any one will have an effect on the achievement of others. For example, security and reliability exist is a state of mutual tension.

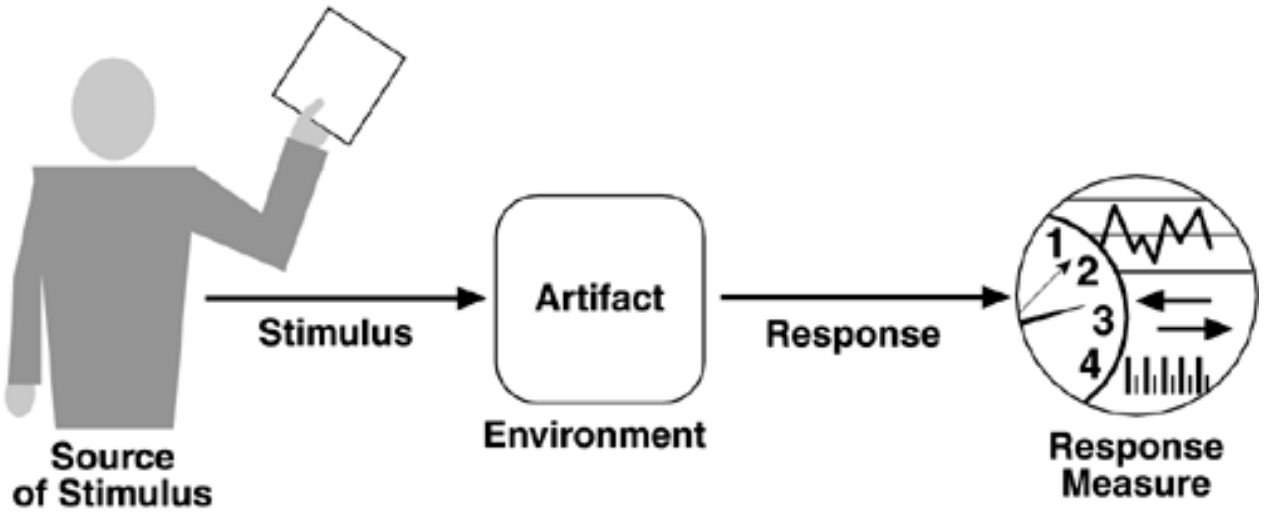
Quality Attribute Scenarios

- A quality attribute scenario is a quality-attribute-specific requirement. It consists of six parts.
  - 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

Quality Attribute Scenarios

- The six parts of a quality attribute scenario (cont'd).
  - 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 is occurs, it should be measurable in some fashion so that the requirement can be tested.

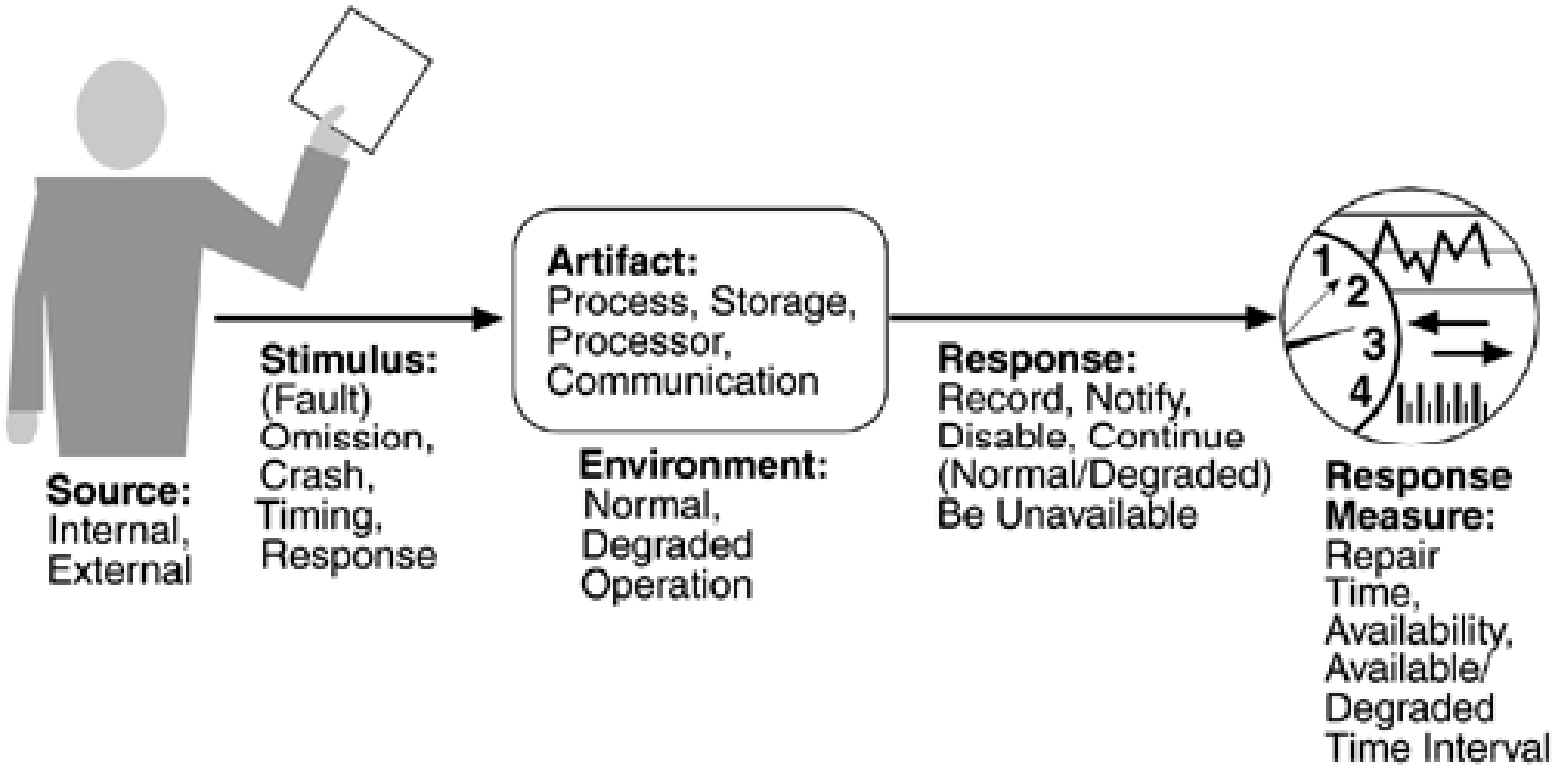
Quality Attribute Scenarios



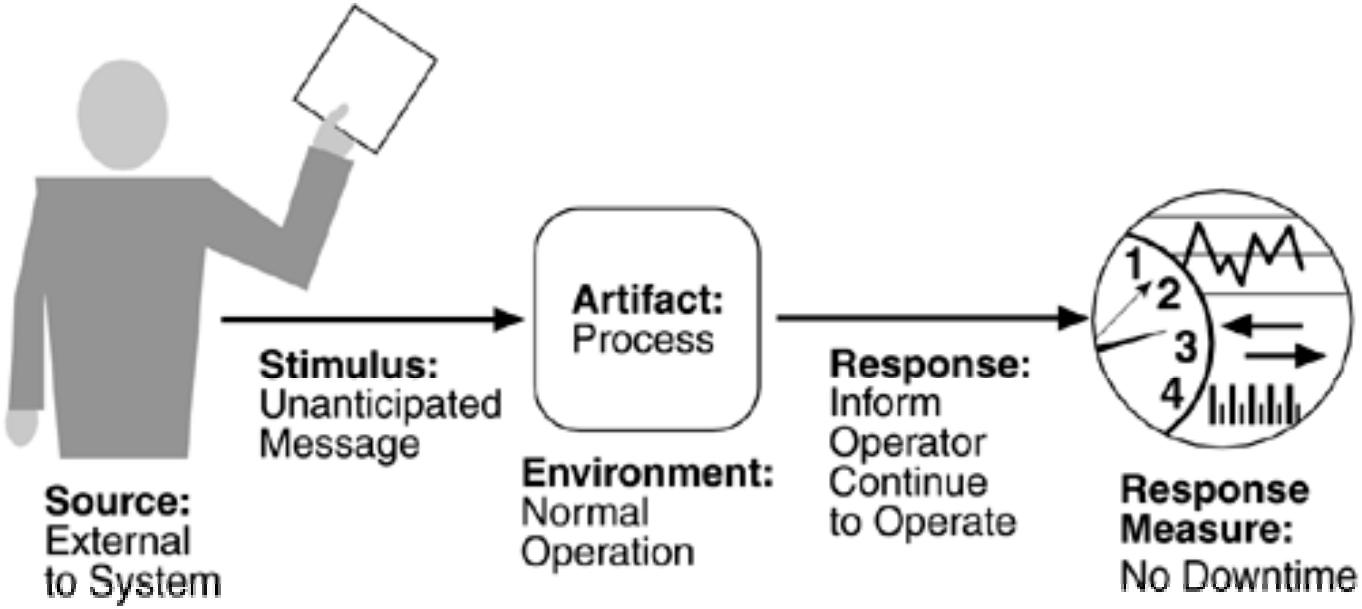
General vs. Concrete Quality Attribute Scenarios

- 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.

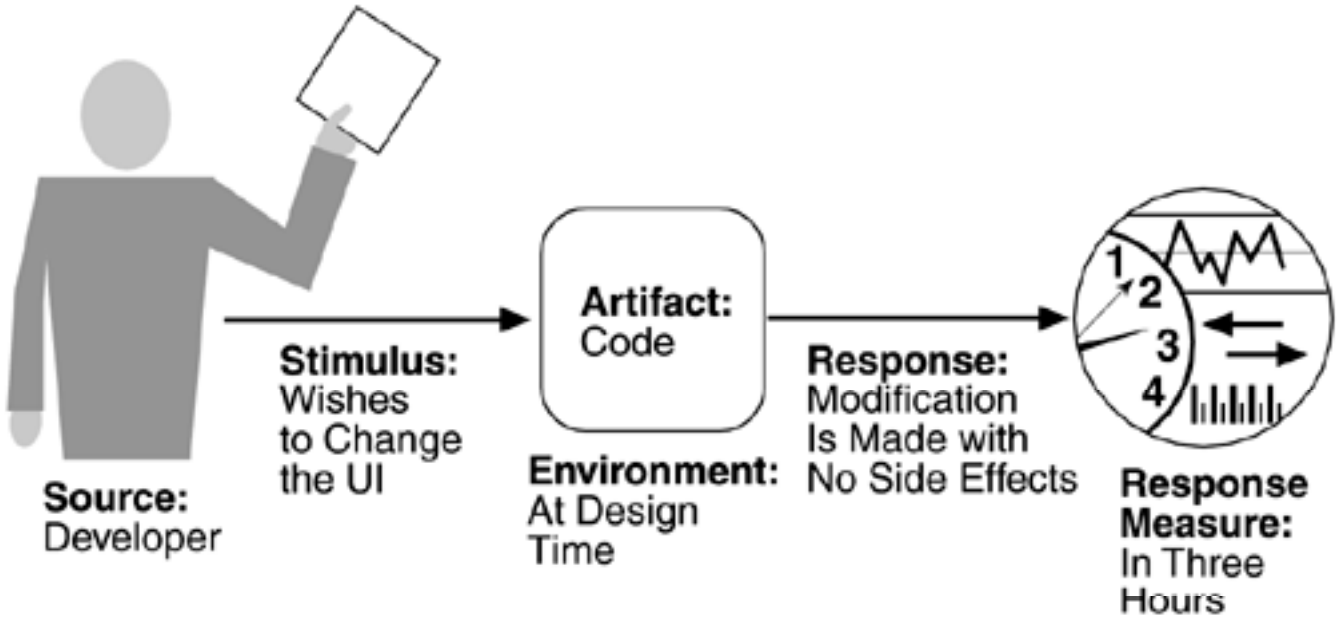
General Scenario for Availability



Sample Concrete Availability Scenario



Sample Modifiability Scenario

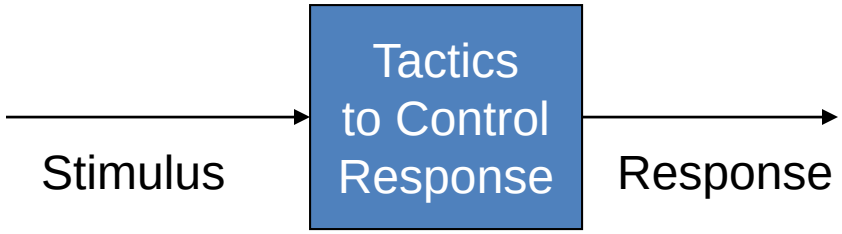


Achieving Qualities

Architectural Tactics

- A tactic is a design decision that influences the control of a quality attribute response.
- A collection of tactics is called an architectural strategy.
- A system design is a collection of decisions
  - Some ensure achievement of the system functionality
  - Others help control the quality attribute responses (which we call the tactics)

Architectural Tactics



Availability Tactics

- All approaches to maintaining availability involve:
  - Some type of redundancy
  - Some type of health monitoring to detect a failure
  - Some type of recovery when a failure is detected (either automatic or manual).

Goal of Availability Tactics

Fault Detection Tactics

- *Ping/echo* – one component issues a ping and expects to receive back an echo within a predefined time.
- *Heartbeat* – one component emits a heartbeat periodically and another component listens for it.
- *Exceptions* – one method for recognizing faults is to encounter an exception raised when a fault is discovered.

Fault Recovery Tactics

- *Voting* – Processes running on redundant processors each take equivalent input and compute an output value that is sent to a voter that makes a decision on what to do using "majority rules" or "preferred component" or other basis.
- *Active redundancy (hot restart)* – All redundant components respond to events in parallel and the response from only one component is used (usually the first to respond).

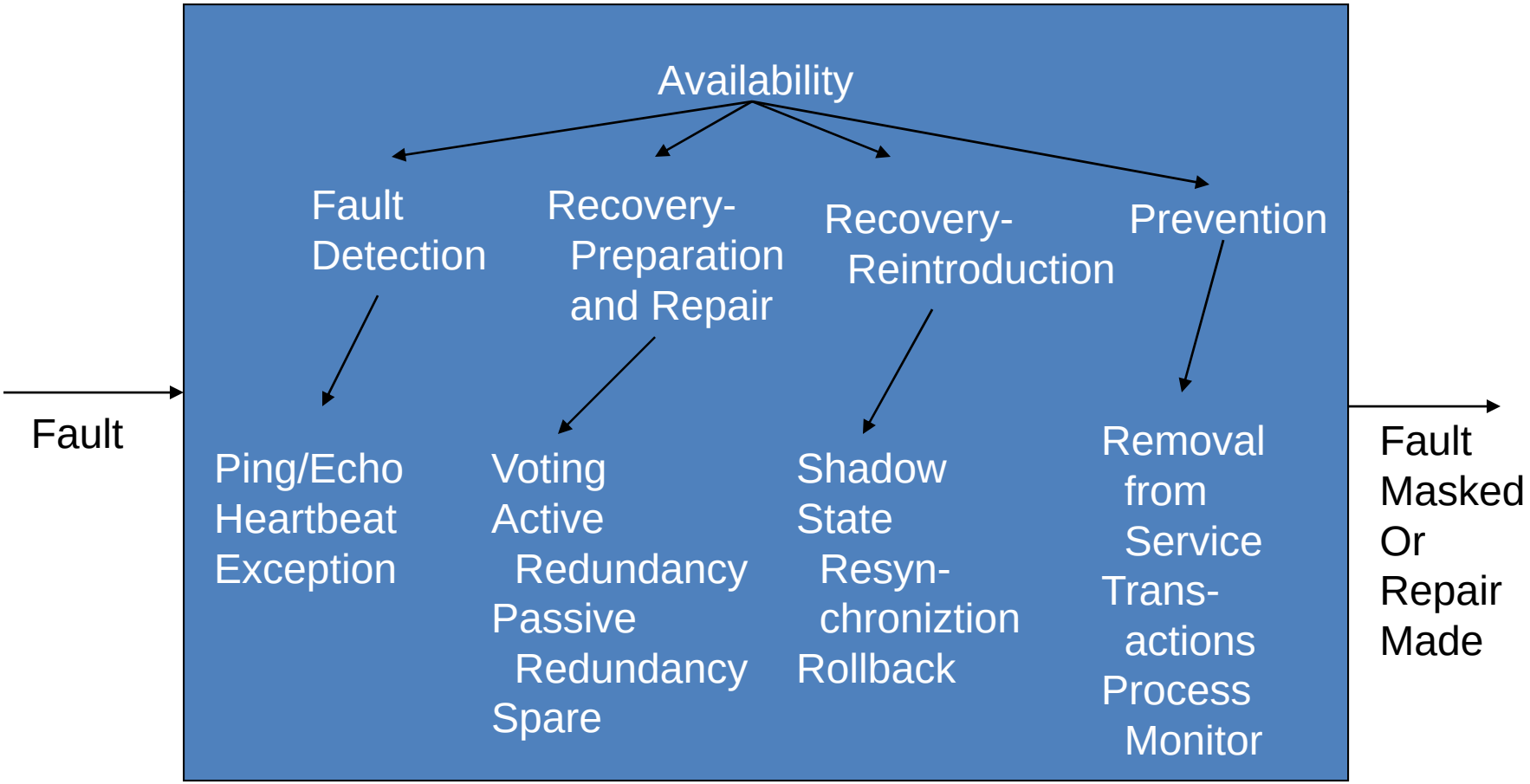
Fault Recovery Tactics

- *Passive redundancy* (warm restart/dual redundancy/triple redundancy) – One component (the primary) responds to events and informs the other components (the standbys) of state updates they must make. When a fault occurs the system must first make sure that the backup state is sufficiently fresh before resuming services.
- *Spare* – A standby spare computing platform is configured to replace many different failed components. It must be rebooted to the proper software configuration and have its state initialized when a failure occurs.

Fault Prevention Tactics

- *Removal from service* – The removal of a component from service to undergo activities to prevent failures.
- *Transactions* – The bundling of several sequential steps in which the entire bundle can be undone at once.
- *Process monitor* – Monitoring for a fault in a process and deleting the nonperforming process and creating a new instance of it.

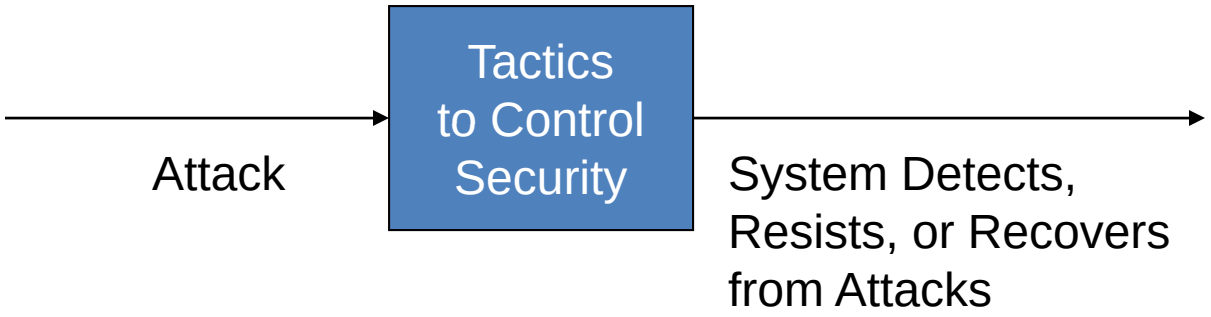
Summary of Availability Tactics



Security Tactics

- Three categories of security tactics
  - Resisting attacks
  - Detecting attacks
  - Recovering from attacks

Goal of Security Tactics



Resisting Attacks

- *Authenticate users* – ensuring that a user or remote computer is actually who it purports to be (e.g., via passwords).
- *Authorize users* – ensuring that an authenticated user has the rights to access and modify either data or services (e.g., via access control by user or user class within the system).
- *Maintain data confidentiality* – data should be protected from unauthorized access (e.g., via encryption of persistent data or use of VPN or SSL for a Web-based link).

Resisting Attacks

- *Maintain integrity* – data should be delivered as intended (e.g., via use of redundant encoded information like checksums or hash results).
- *Limit exposure* – allocate services to hosts so that limited services are available on each host.
- *Limit access* – restrict access based on message source or destination port if possible (e.g., via firewalls)

Detecting Attacks

- The detection of an attack is usually done through an intrusion detection system.
- These systems compare network traffic patterns to a database of patterns.

Detecting Attacks

- Intrusion detectors must have:
  - Some sort of sensor to detect attacks
  - Managers to do sensor fusion
  - Databases for storing events for later analysis
  - Tools for offline reporting and analysis
  - A control console so that the analyst can modify intrusion detection actions.

Recovering from Attacks

- Tactics concerned with restoring state: these overlap with availability tactics
- Concerned with attacker identification (for either preventive or punitive purposes)

Summary of Security Tactics