Chapter 9: Security

What is Security?

- **Security** is a measure of the system's ability to protect data and information from unauthorized access while still providing access to people and systems that are authorized
- An action taken against a computer system with the intention of doing harm is called an ***attack***
- Attack can be in different forms
  - unauthorized attempt to access/modify data or service
  - intended to deny services to legitimate users

What is Security?

- Security has three main characteristics, called CIA:
- **Confidentiality** is the property that data or services are protected from unauthorized access.
  - For example, a hacker cannot access your income tax returns on a government computer.
- **Integrity** is the property that data or services are not subject to unauthorized manipulation.
  - For example, your grade has not been changed since your instructor assigned it.
- **Availability** is the property that the system will be available for legitimate use.
  - For example, a denial-of-service attack prevent you from ordering a book from an online bookstore.

What is Security?

- Other characteristics that support CIA are
  - **Authentication** verifies the identities of the parties to a transaction and checks if they are truly who they claim to be.
  - **Authorization** grants a user the privileges to perform a task.

Authentication

Who you are

Authorization

What you can do

Security General Scenario

Portion of Scenario	Possible Values
Source	Human or another system which may have been previously identified (either correctly or incorrectly) or may be currently unknown. A human attacker may be from outside the organization or from inside the organization.
Stimulus	**Unauthorized attempt** is made to display data, change or delete data, access system services, change the system's behavior, or reduce availability.
Artifact	System services; data within the system; a component or resources of the system; data produced or consumed by the system
Environment	The system is either online or offline, connected to or disconnected from a network, behind a firewall or open to a network, fully operational, partially operational, or not operational

Security General Scenario

Portion of Scenario	Possible Values
Response	Transactions are carried out in a fashion such that • data or services are protected from unauthorized access; • data or services are not being manipulated without authorization; • the data, resources, and system services will be available for legitimate use. The system tracks activities within it by • recording access or modification, • recording attempts to access data, resources or services, • notifying appropriate entities when an apparent attack is occurring.

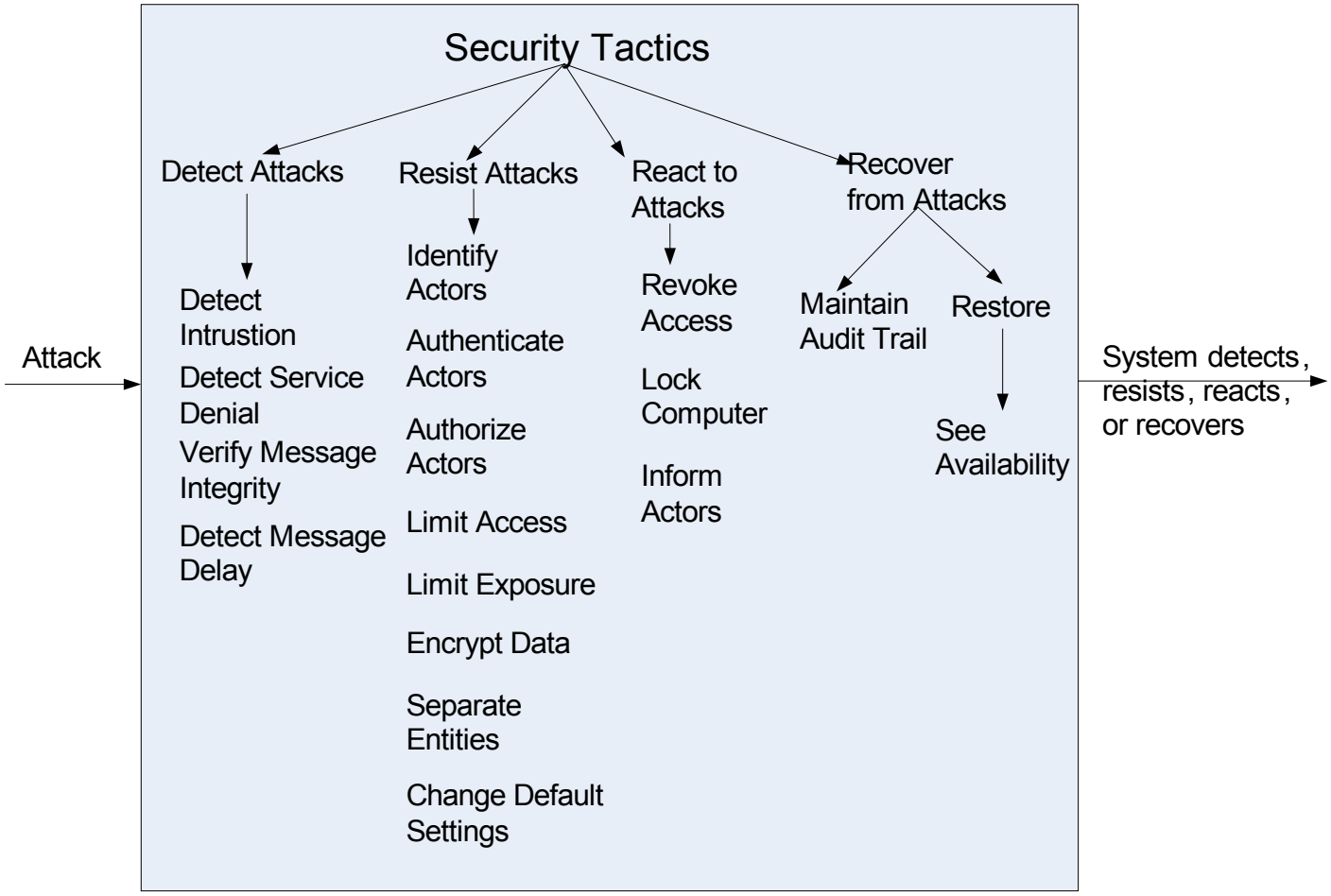
Security General Scenario

Portion of Scenario	Possible Values
Response Measure	One or more of the following • how much of a system is compromised when a particular component or data value is compromised, • how much time passed before an attack was detected, • how many attacks were resisted, • how long does it take to recover from a successful attack, • how much data is vulnerable to a particular attack

Sample Concrete Security Scenario

- A disgruntled employee from a remote location attempts to modify the pay rate table during normal operations. The system maintains an audit trail and the correct data is restored within a day.
  - Stimulus: unauthorized attempts to modify the pay rate table
  - Stimulus source: a disgruntled employee
  - Artifact: the system with pay rate table
  - Environment: during normal operation
  - Response: maintains an audit trail
  - Response measure: correct data is restored within a day

Security Tactics

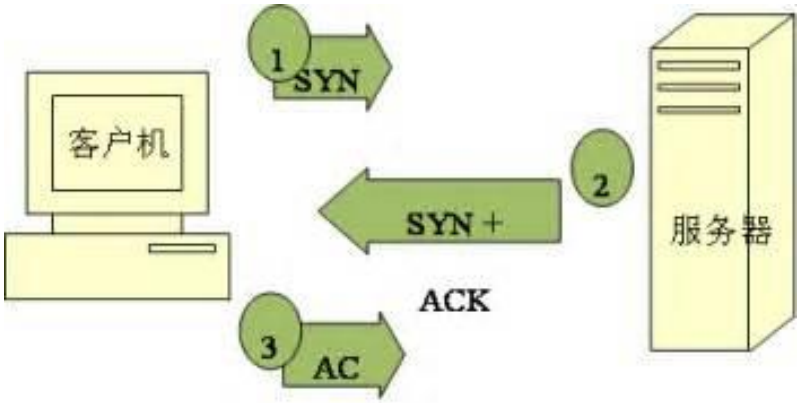


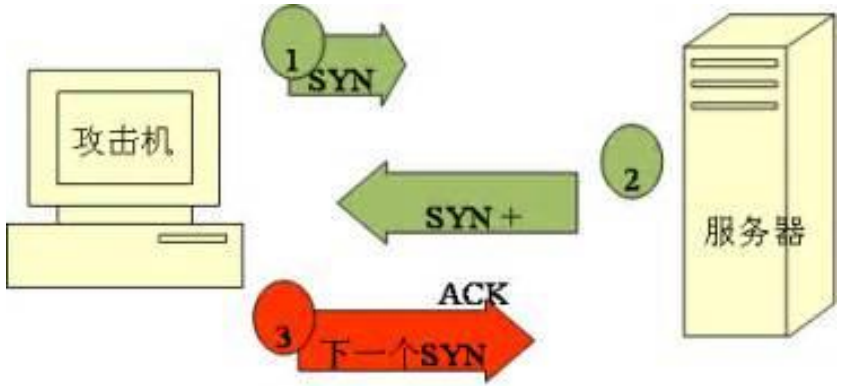
Detect Attacks

- **Detect Intrusion**: compare network traffic or service request patterns *within* a system to a set of signatures or known patterns of malicious behavior stored in a database.
- **Detect Service Denial**: comparison of the pattern or signature of network traffic *coming into* a system to historic profiles of known Denial of Service (DoS) attacks.

Deny of Service Attack

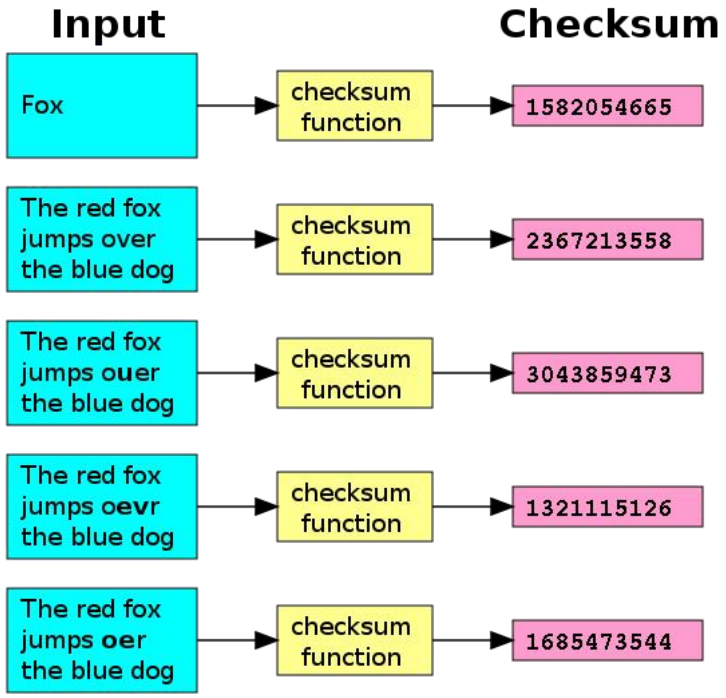
- Ping of Death
- UDP Flood
- TCP SYN





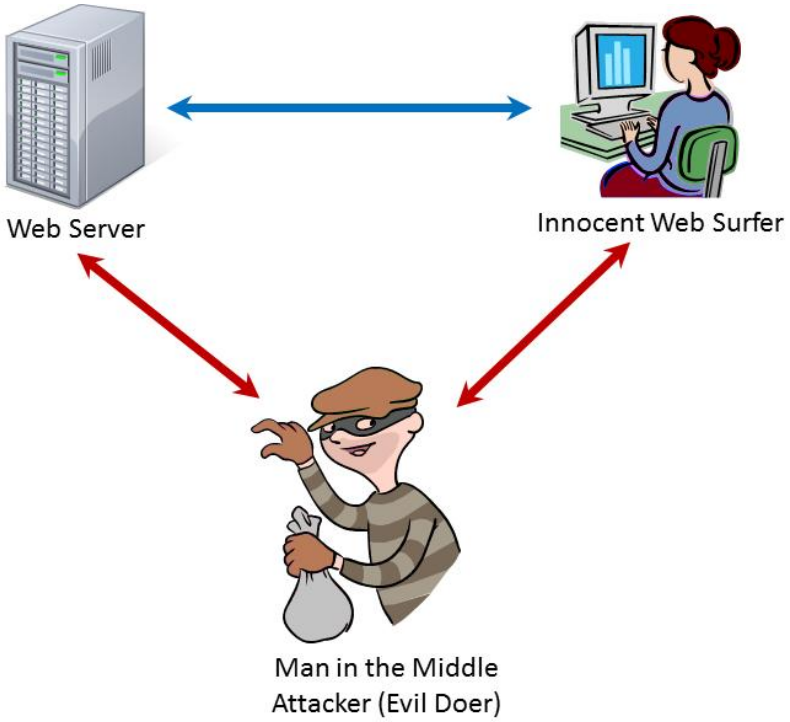
Detect Attacks

- **Verify Message Integrity**: use techniques such as *checksums* or *hash values* to verify the integrity of messages, resource files, deployment files, and configuration files.



Detect Attacks

- **Detect Message Delay**: checking the time that it takes to deliver a message, it is possible to detect suspicious timing behavior, i.e., man-in-the-middle attack



Resist Attacks

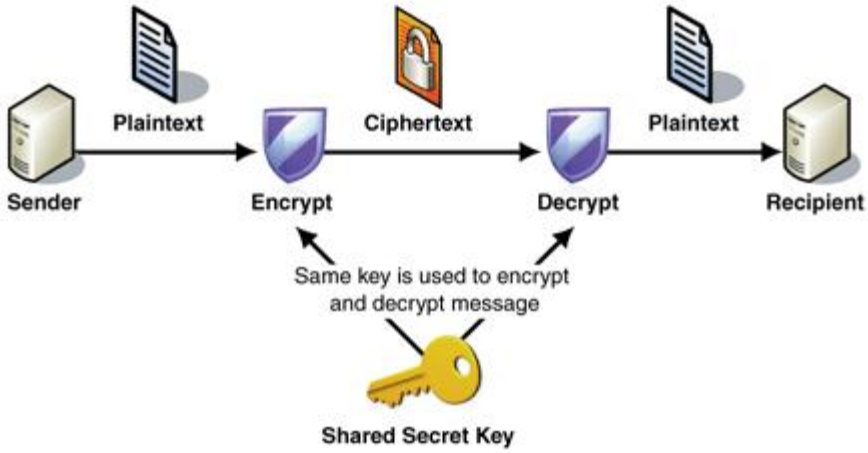
- **Identify Actors**: identify the source of any external input to the system.
- **Authenticate Actors:** ensure that a user or remote computer is actually who or what it purports to be.
- **Authorize Actors:** ensuring that an authenticated actor has the rights to access and modify either data or services.
- **Limit Access**: limiting access to resources such as memory, network connections, or access points.

Resist Attacks

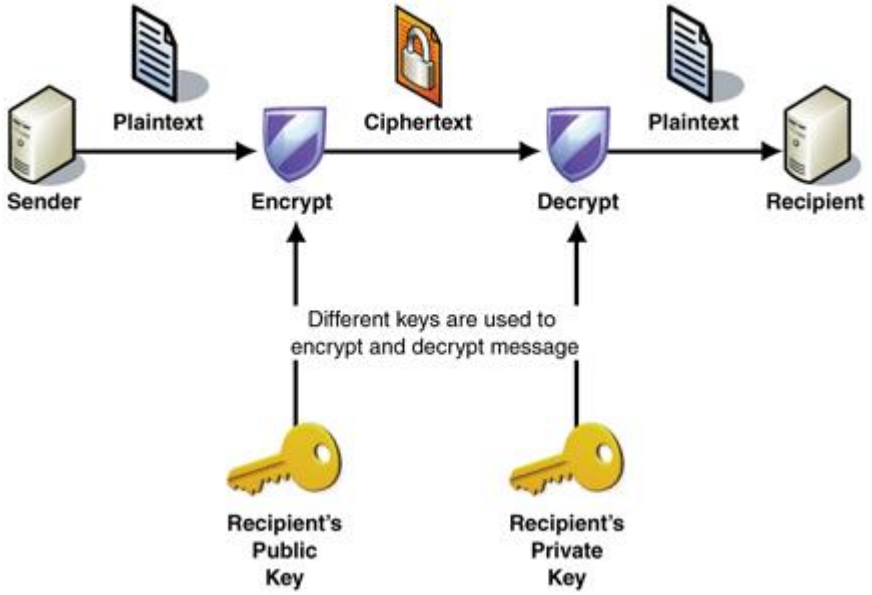
- **Limit Exposure**: minimize the attack surface of a system by having the fewest possible number of access points.
- E.g., firewall is a single point of access to the intranet
- E.g., closing a port
- Passive defense

Resist Attacks

- **Encrypt Data**: apply some form of encryption to data and to communication.



Symmetric encryption



asymmetric

Resist Attacks

- **Separate Entities:** can be done through physical separation on different servers, the use of virtual machines
- **Change Default Settings**: Force the user to change settings assigned by default.

React to Attacks

- **Revoke Access**: limit access to sensitive resources, even for normally legitimate users and uses, if an attack is suspected.
- **Lock Computer**: limit access to a resource if there are repeated failed attempts to access it.
- **Inform Actors**: notify operators, other personnel, or cooperating systems when an attack is suspected or detected.

Summary

- Attacks against a system can be characterized as attacks against the confidentiality, integrity, or availability of a system.
- Identifying, authenticating, and authorizing actors are tactics intended to determine which users or systems are entitled to what kind of access to a system.
- Tactics exist to detect an attack, limit the spread of any attack, and to react and recover from an attack.

Chapter 10: Testability

What is Testability?

- **Software testability** refers to the ease with which software can be made to demonstrate its faults through (typically execution-based) testing.
- **Testability** refers to the probability that it will fail on its *next* test execution.
- If a fault is present in a system, then we want it to fail during testing as quickly as possible.

What is Testability?

- For a system to be testable, it must be possible to *control* each component's inputs (and possibly manipulate its internal state) and then to *observe* its outputs (and possibly its internal state).

Testability General Scenario

Portion of Scenario	Possible Values
Source	Unit testers, integration testers, system testers, acceptance testers, end users, either running tests manually or using automated testing tools
Stimulus	A set of tests are executed due to the completion of a coding increment; the complete implementation of the system; or the delivery of the system to the customer.
Environment	Design time, development time, compile time, integration time, deployment time, run time
Artifacts	The portion of the system being tested
Response	One of the following: execute tests and capture results; capture activity that resulted in the fault; control and monitor the state of the system

Testability General Scenario

Portion of Scenario	Possible Values
Response Measure	One or more of the following: effort to find a fault; effort to achieve a given percentage of state space coverage; probability of fault being revealed by the next test; time to perform tests; effort to detect faults; length of time to prepare test environment

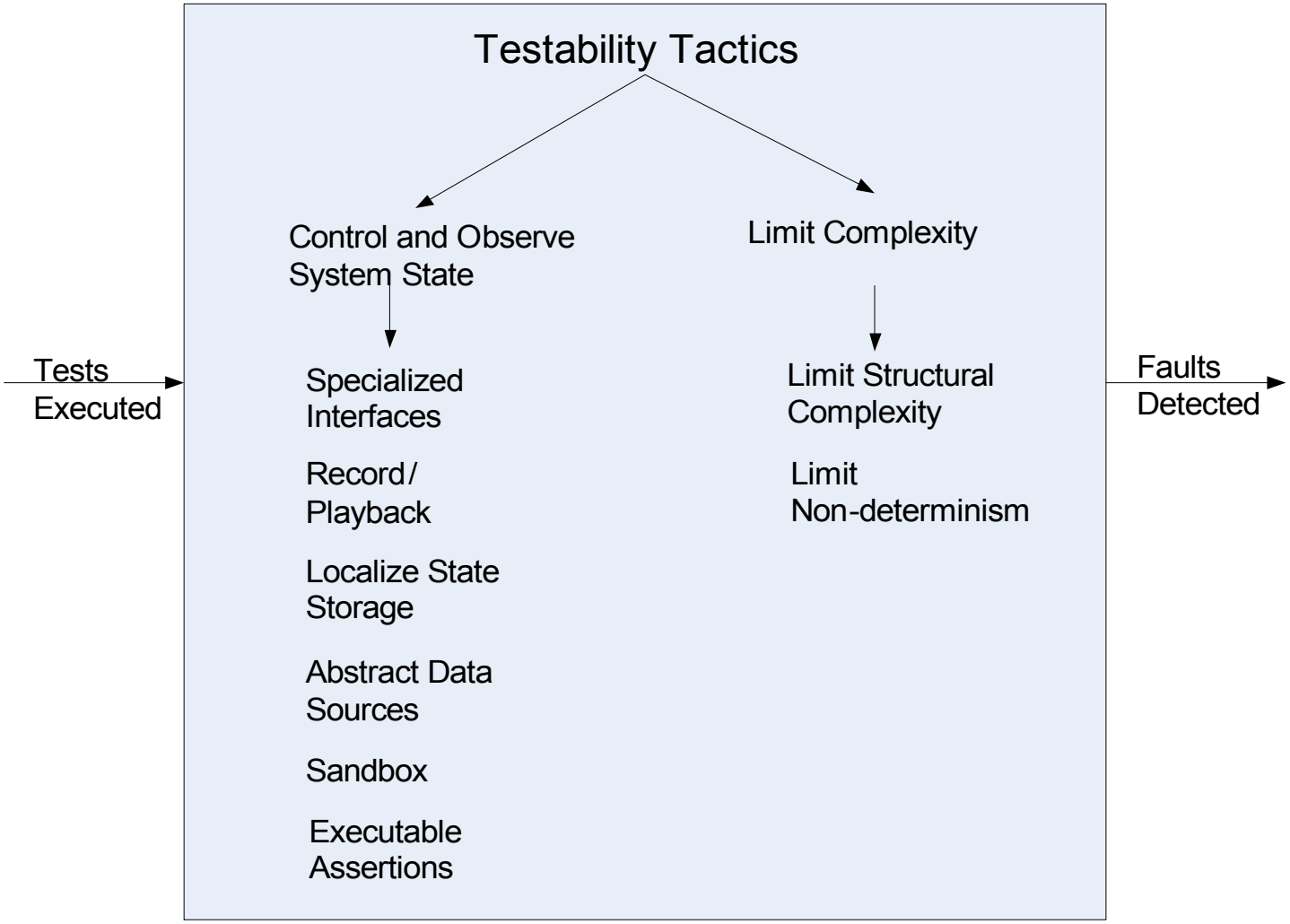
Sample Concrete Testability Scenario

- The unit tester completes a code unit during development and performs a test sequence whose results are captured and that gives 85% path coverage within 3 hours of testing.
  - Source: unit tester
  - Stimulus: code unit completed
  - Artifact: code unit
  - Environment: development
  - Response: capture results
  - Response measure: 85% path coverage in three hours

Goal of Testability Tactics

- **The goal of tactics** for testability is to allow for easier testing when an increment of software development has completed.
- There are two categories of tactics for testability:
  - The first category deals with *adding controllability and observability* to the system.
  - The second deals with *limiting complexity* in the system's design.

Testability Tactics



Specialized Interfaces

- To control or capture variable values for a component through normal execution.
- A *set* and *get* method of variables
- A *report* method that returns state of the object
- A *reset* method to set the internal state

Control and Observe System State

- **Record/Playback**: capturing information crossing an interface and using it as input for further testing.
- **Localize State Storage**: to start a system in an arbitrary state for a test, it is most convenient if that state is stored in a single place.

Control and Observe System State

- **Sandbox**: isolate the system from the real world to enable experimentation
- **Executable Assertions**: assertions are hand coded and placed at desired locations to indicate when and where a program is in a faulty state

Limit Complexity

- **Limit Structural Complexity**: avoiding or resolving cyclic dependencies between components, and reducing dependencies between components in general
- E.g., limit the depth of inheritance tree
- High cohesion, and loose coupling can help testability

Limit Complexity

- **Limit Non-determinism**: finding all the sources of non-determinism, such as unconstrained parallelism, and weeding them out as far as possible.

Testability v.s. Fault-tolerance

- Testability is to make the fault easier to show up
- Fault-tolerance attempts to hide the faults, and make it difficult to reveal the faults.
- Are the two design goals incompatible?

Testability and other Quality Qttributes

- What other quality attributes do you think testability is most in conflict with? What other quality attributes do you think testability is most compatible with?

Summary

- Controlling and observing the system state are a major class of testability tactics
- Complex systems are difficult to test because of
  - the large state space in which their computations take place,
  - and the larger number of interconnections among the elements of the system.
- Keeping the system simple is another class of tactics that supports testability.

Chapter 11: Usability

What is Usability?

- **Usability** is concerned with *how easy it is for the user to accomplish a desired task* and *the user support the system provides*
- **Usability** is one of the cheapest and easiest ways to improve a system's quality

What is Usability?

- Usability comprises the following areas:
  - Using a system efficiently.
  - Learning system features.
  - Minimizing the impact of errors.
  - Adapting the system to user needs.
  - Increasing confidence and satisfaction.

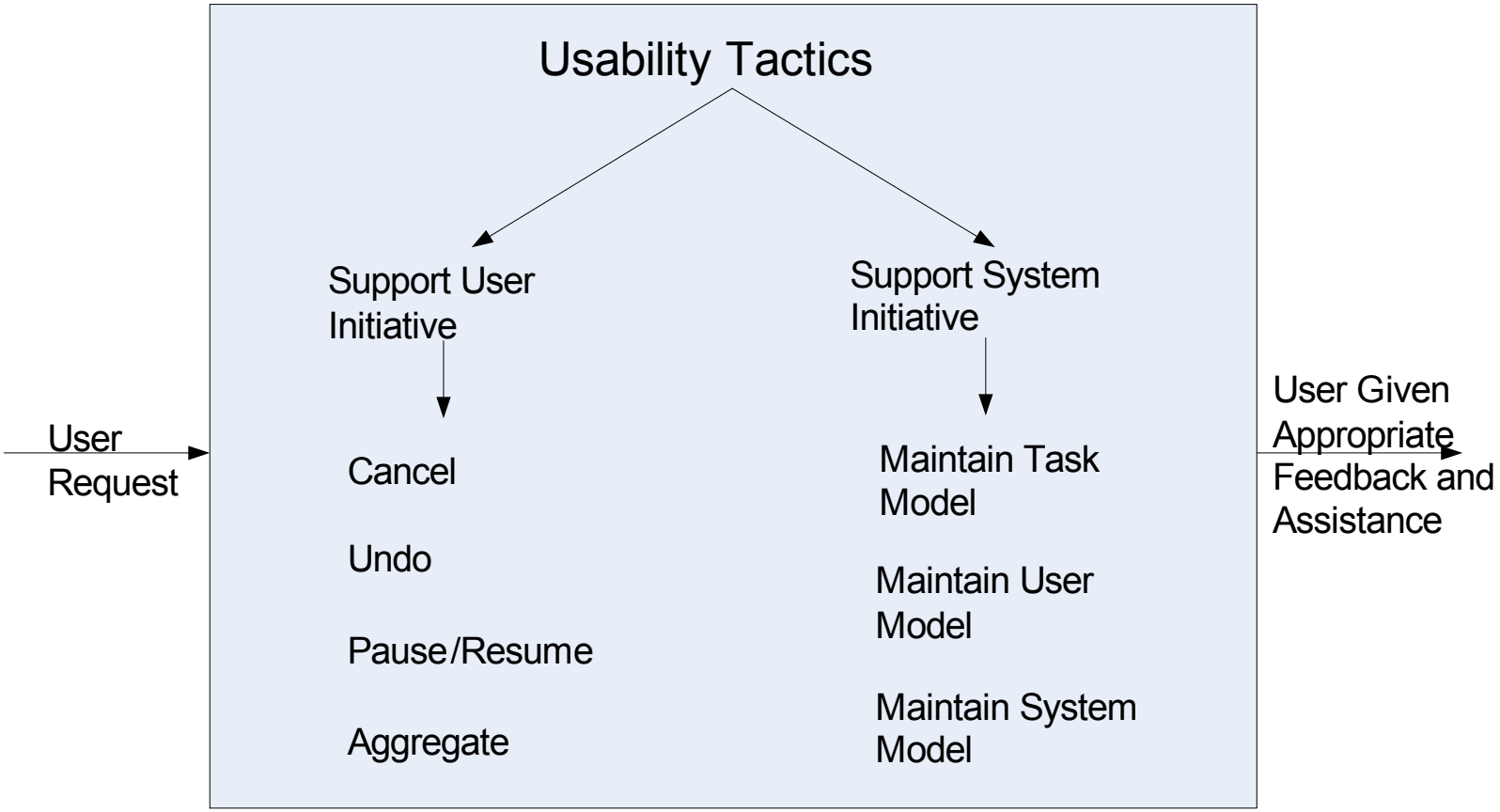
Usability General Scenario

Portion of Scenario	Possible Values
Source	End user
Stimulus	End user tries to use a system efficiently, learn to use the system, minimize the impact of errors, adapt the system, or configure the system
Environment	Runtime or configuration time
Artifacts	System or the specific portion of the system with which the user is interacting.
Response	The system should either provide the user with the features needed or anticipate the user's needs.
Response Measure	One or more of the following: task time, number of errors, number of tasks accomplished, user satisfaction, gain of user knowledge, ratio of successful operations to total operations, or amount of time or data lost when an error occurs.

Sample Concrete Usability Scenario

- The user downloads a new application and is using it productively after two minutes of experimentation.
  - Source: user
  - Stimulus: download a new application
  - Artifact: system
  - Environment: runtime
  - Response: user uses application productively
  - Response measure: within two minutes of experimentation

Usability Tactics



Support User Initiative

- **Cancel**: the system must listen for the cancel request;
  - the command being canceled must be terminated; resources used must be freed; and collaborating components must be informed.
- **Pause/Resume**: temporarily free resources so that they may be re-allocated to other tasks.
  - Used for long-running operation

Support User Initiative

- **Restore**: maintain a sufficient amount of information about system state so that an earlier state may be restored
- **Aggregate**: ability to aggregate lower-level objects into a group, so that a user operation may be applied to the group, freeing the user from the repetitive operations.

Support System Initiative

- **Maintain Task Model**: determines context so the system can have some idea of what the user is attempting and provide assistance.
- **Maintain System Model**: system maintains an explicit model of itself. This is used to determine expected system behavior so that appropriate feedback can be given to the user.

Summary

- Architectural support for usability involves allowing the user to take the initiatives such as cancelling a long running command, undoing a completed command, and aggregating data and commands.
- To predict user or system response, the system must keep a model of the system, and the task.

Chapter 12:
Other Quality Attributes

Portability

- **Portability** is also a special form of modifiability.
- Portability refers to the ease with which software that built to run on one platform can be changed to run on a different platform.

Development Distributability

- **Development Distributability** is the quality of designing the software to support distributed software development.
- The system are developed using globally distributed teams.
- The purpose is to minimize the coordination among the teams.

Scalability

- **Horizontal scalability** (scaling out) refers to adding more resources to logical units such as adding another server to a cluster.
- **Vertical scalability** (scaling up) refers to adding more resources to a physical unit such as adding more memory to a computer.
- Results in a measurable improvement without disruptive operations
- Centralized v.s. decentralized?

Deployability

- Deployability is concerned with how an executable arrives at a host platform and how it is invoked.
- Examples:
  - Javascripts
  - Virus
  - Agent

Mobility

- Mobility deals with the problems of movement and affordances of a platform
- Issues in mobility include
  - Battery management
  - Disconnection and reconnection
  - Hand-off in cellular network
  - …

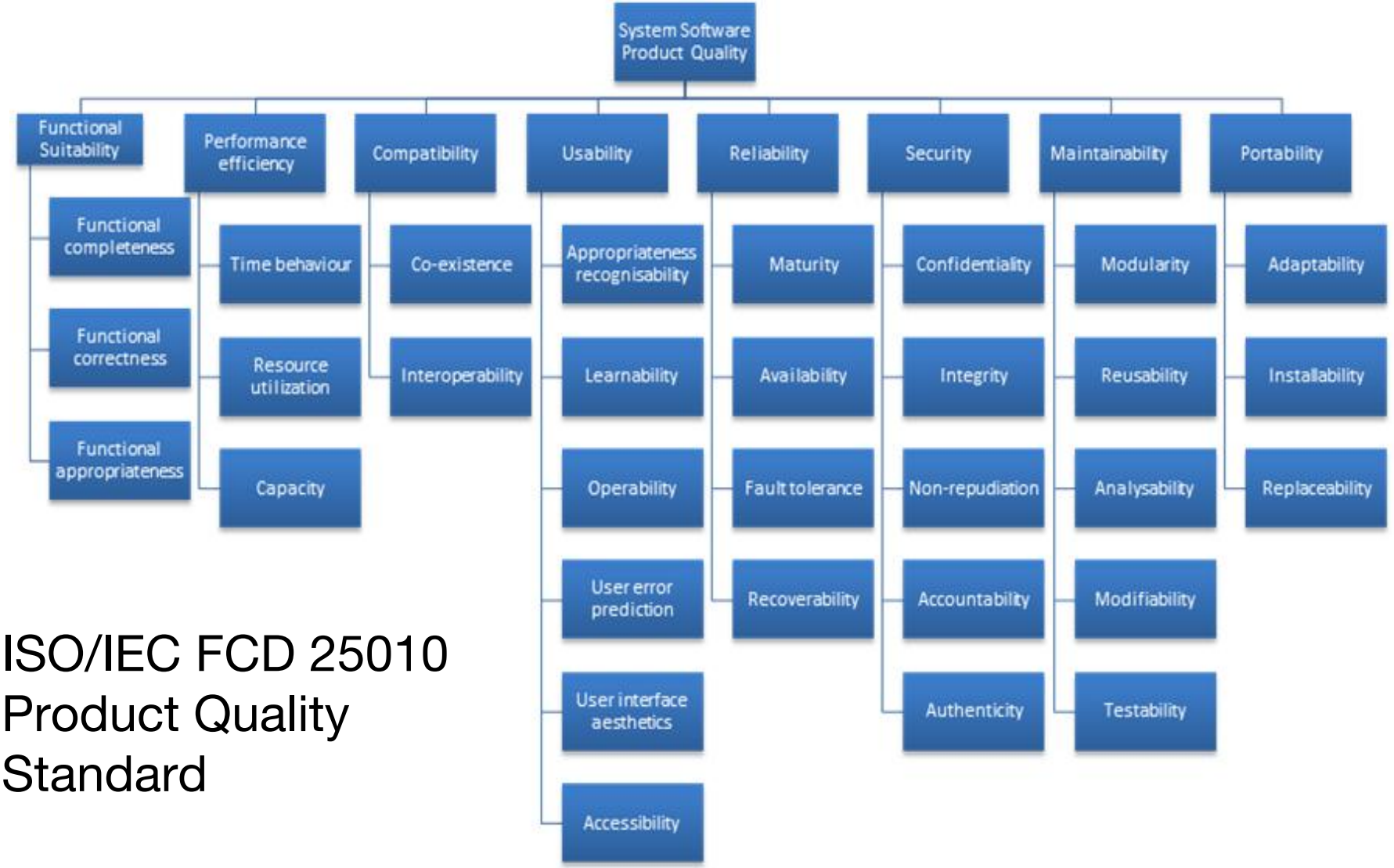
Monitorability

- **Monitorability** deals with the ability of the operations staff to monitor the system while it is executing.
- For example
  - the health of various component
  - Average transaction processing time
  - Queue length

Safety

- **Software safety** is about the software's ability to avoid entering states that cause or lead to damage, injury, or loss of life
- and to recover and limit the damage when it does enter into bad states.
- The architectural concerns with safety are almost identical with those for availability

Standard Lists of Quality Attributes



ISO/IEC FCD 25010 Product Quality Standard