Quality Attributes and Tactics

Some material in these slides is adapted from Software Architecture in Practice, 2nd edition by Bass, Clements and Kazman.

Requirements

- Are Functional and Quality requirements orthogonal?
- Architectural Vs Non-Architectural requirements

Use Case Template

Use Case Number:	EU-xxxx : Indicates an essential use case, i.e., a use case that describes activity in system independent terms	
Use Case Name:	*Enter name of Use Case.*	
Overview:	*Describe the purpose of the Use Case and give a brief description.*	
Type:	*Enter Use Case priority* (primary, secondary, optional)	
Actors:	*List all actors that participate in this Use Case. Indicate the actor that initiates the use case by placing "initiator" in brackets after the actor name. Also, indicate primary actors by placing "primary" in brackets after actor name.*	
Properties:	***Performance:***	
	Security:	
	Other:	

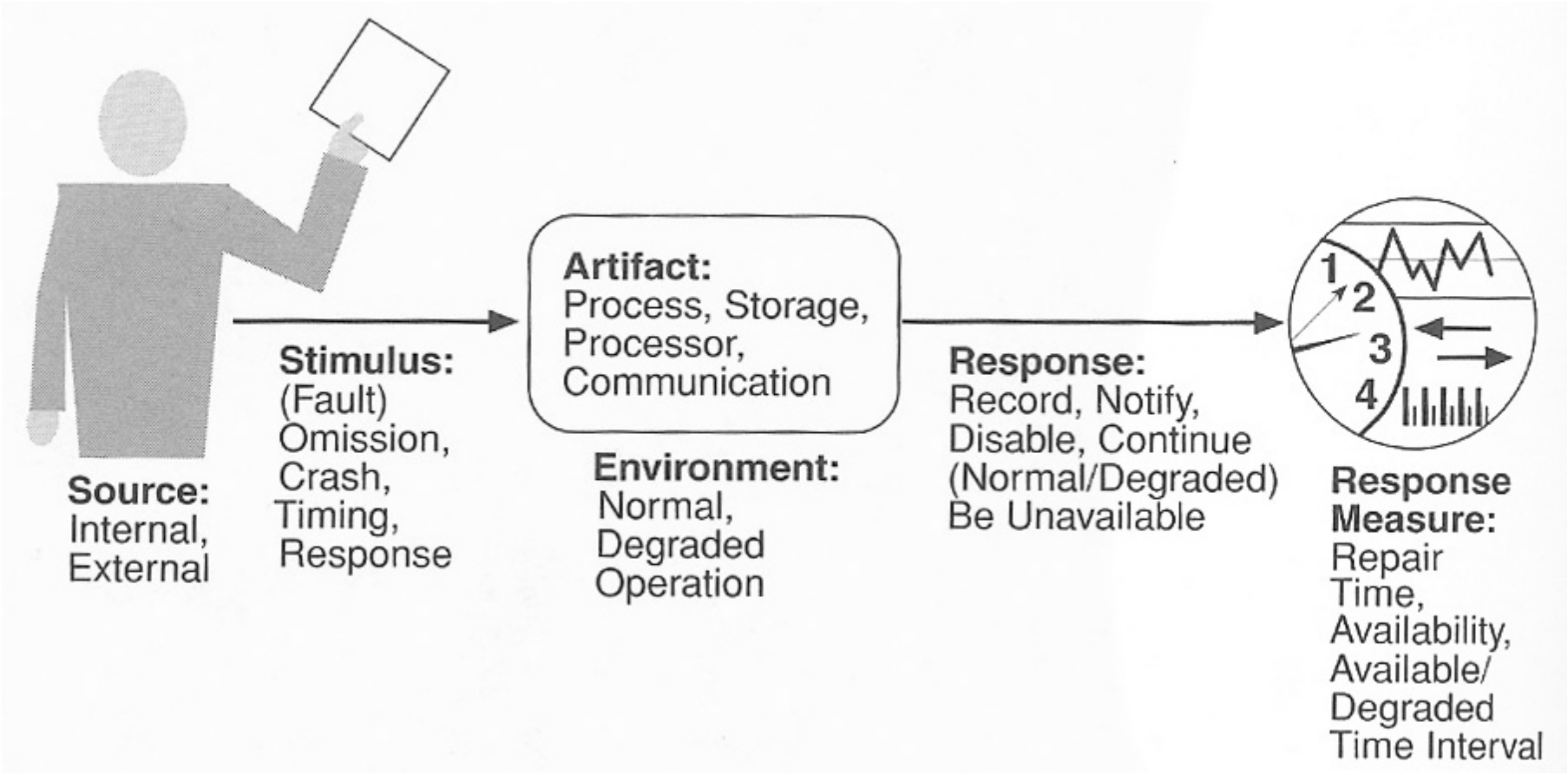
Use Case Template – cont'd

Pre condition:	*Enter the condition that must be true when the main flow is initiated. This should reference the conceptual model.*	
Flow:	***Main Flow:*** *Steps should be numbered.*	
		Subflows: *Break down of main flow steps*
		Alternate Flows: *Include the post condition for each alternate flow if different from the main flow.*
Post Condition:	*Enter the condition that must be true when the main flow is completed. This should reference the conceptual model. Include the following information in this section:*	
Cross References:	*References to other Use Cases or textual requirements that relate to this Use Case.*	

System Qualities

- Availability
- Performance
- Security
- Modifiability
- Testability
- Usability

Quality Scenarios - General



Availability

- Failures and faults
- Mean time to failure, repair
- Downtime

Availability Table

- Source: internal, external
- Stimulus: type of fault
- Artifact: processors, channels, storage
- Environment: normal, degraded
- Response: logging, notification, switching to backup, restart, shutdown
- Measure: availability, repair time, required uptime

Availability Scenario example

Availability of the crossing gate controller:

Scenario: Main processor fails to receive an acknowledgement from gate processor.

- Source: external to system
- Stimulus: timing
- Artifact: communication channel
- Environment: normal operation
- Response: log failure and notify operator via alarm
- Measure: no downtime

Availability Tactics

- Fault detection
  - Ping/echo
  - Heartbeat
  - Exception
- Fault recovery
  - Preparation
    - Voting
    - Active redundancy (hot backup)
    - Passive redundancy (warm backup)
    - Spare
  - Re-Introduction
    - Shadow operation
    - State resynchronization
    - Checkpoint/rollback
- Fault prevention
  - Removal from service
  - Transactions
  - Process monitor

System Qualities

- Availability
- **Performance**
- Security
- Modifiability
- Testability
- Usability

Performance

- Event arrival patterns
  - Periodic
  - Stochastic
  - Sporadic

- Event servicing
  - Latency – Time between the arrival of stimulus and the system's response to it
  - Jitter – Variation in latency
  - Throughput – Number of transactions the system can process in a second

Performance Table

- Source: external, internal
- Stimulus: event arrival pattern
- Artifact: system services
- Environment: normal, overload
- Response: change in mode?
- Measure: latency, deadline, throughput, jitter, miss rate, data loss

Performance Scenario example

Performance of the crossing gate controller:

Scenario: Main processor commands gate to lower when train approaches.

- Source: external - arriving train
- Stimulus: sporadic
- Artifact: system
- Environment: normal mode
- Response: remain in normal mode
- Measure: send signal to lower gate within 1 millisecond

Performance Tactics

- Resource demand
  - Increase computational efficiency
  - Reduce computational overhead
  - Manage event rate
  - Control frequency of sampling
  - Bound execution times
  - Bound queue sizes
- Resource management
  - Introduce concurrency
  - Maintain multiple copies of data or computations
  - Increase available resources
- Resource arbitration
  - Scheduling policy
    - FIFO
    - Fixed-priority
      - semantic importance – Based on domain characteristic
      - deadline monotonic – Higher priority to shorter deadlines
      - rate monotonic – Higher priority to shorter periods
    - Dynamic priority

System Qualities

- Availability
- Performance
- **Security**
- Modifiability
- Testability
- Usability

Security

- Nonrepudiation
- Confidentiality
- Integrity
- Assurance
- Availability
- Auditing

Security Table

- Source: user/system, identified?
- Stimulus: display info, change info, access services, deny services
- Artifact: services, data
- Environment: online/offline, connected?
- Response: logging, block access, notification
- Measure: time, probability of detection, recovery

Security Scenario example

Security of the crossing gate controller:

Scenario: Hackers are prevented from disabling system.

- Source: unauthorized user
- Stimulus: tries to disable system
- Artifact: system service
- Environment: online
- Response: blocks access
- Measure: service is available within 1 minute

Security Tactics

- Resisting attacks
  - Authenticate users
  - Authorize users
  - Maintain data confidentiality
  - Maintain integrity
  - Limit exposure
  - Limit access
- Detecting and Recovering from attacks
  - Intrusion detection system - anomalous patterns
  - Restoration - see availability tactics
  - Identification - maintain audit trail

System Qualities

- Availability
- Performance
- Security
- **Modifiability**
- Testability
- Usability

Modifiability

- What can change?
- When is it changed?
- Who changes it?

Modifiability Table

- Source: developer, administrator, user
- Stimulus: add/delete/modify function or quality
- Artifact: UI, platform, environment
- Environment: design, compile, build, run
- Response: make change and test it
- Measure: effort, time, cost

Modifiability Scenario example

Modifiability of the restaurant locator:

Scenario: User may change behavior of system.

- Source: end user
- Stimulus: wishes to change radius of search
- Artifact: list of available restaurants and their locations
- Environment: runtime
- Response: finds option and uses it successfully
- Measure: no more than 5 clicks required

Modifiability Tactics

- Localize changes
  - Maintain semantic coherence
    - abstract common services
  - Anticipate expected changes
  - Generalize module
  - Limit possible options
- Prevent ripple effects
  - Hide information
  - Maintain existing interfaces
  - Restrict communication paths
  - Use an intermediary
- Defer binding time

System Qualities

- Availability
- Performance
- Security
- Modifiability
- **Testability**
- Usability

Testability

- Probability of fault discovery
- Need to control components
- Need to observe component failure

TestabilityTable

- Source: developer, tester, user
- Stimulus: milestone completed
- Artifact: design, code component, system
- Environment: design, development, compile, deployment, run
- Response: can be controlled and observed
- Measure: coverage, probability, time

Testability Scenario example

Testability of the restaurant locator:

Scenario: New versions of system can be completely tested relatively quickly.

- Source: system tester
- Stimulus: integration completed
- Artifact: whole system
- Environment: development time
- Response: all functionality can be controlled and observed
- Measure: entire regression test suite completed in less than 24 hours

Testability Tactics

- Manage Input/Output
  - Record/playback
  - Separate interface from implementation
  - Specialize access
- Internal monitoring
  - Built-in monitors

System Qualities

- Availability
- Performance
- Security
- Modifiability
- Testability
- **Usability**

Usability

- Learning System features.
- Using a system efficiently.
- Minimizing the impact of errors.
- Adapting the system to user needs.
- Increasing confidence and satisfaction.

UsabilityTable

- Source: end user
- Stimulus: wish to learn/use/minimize errors/ adapt/feel comfortable
- Artifact: system
- Environment: configuration or runtime
- Response: provide ability or anticipate
- Measure: task time, number of errors, user satisfaction, efficiency

Usability Scenario example

Usability of the restaurant locator:

Scenario: User may undo actions easily.

- Source: end user
- Stimulus: minimize impact of errors
- Artifact: system
- Environment: runtime
- Response: wishes to undo a filter
- Measure: previous state restored within one second

Usability Tactics

- Design time
  - Separate UI from rest of system
- Runtime
  - Support user initiative
    - Cancel, Undo
  - Support system initiative
    - User/System/Task models

Architecture Activity (Individual)

- Describe concrete scenarios related to your project 3 case study for 2-3 system qualities using the **table format.**