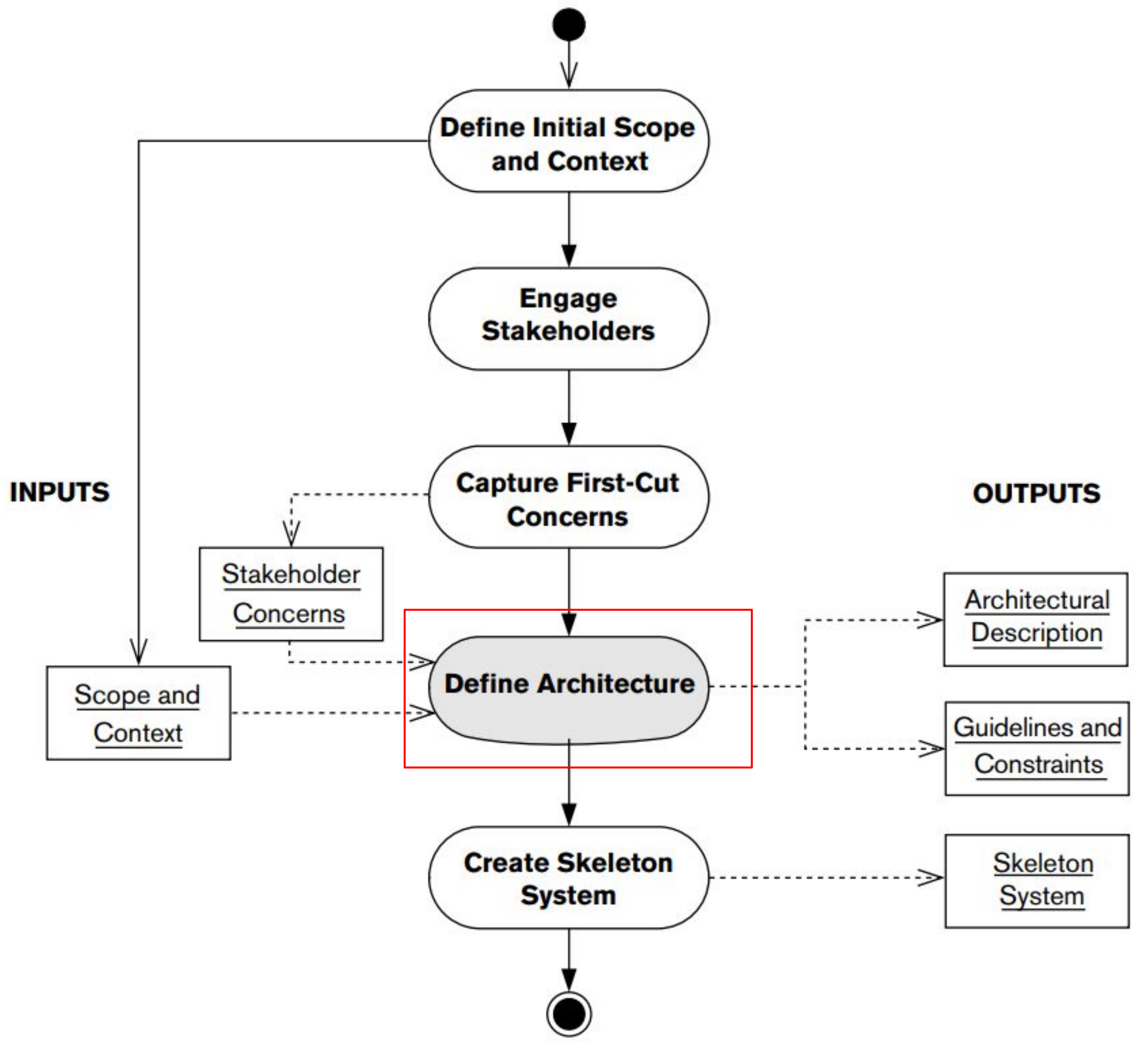
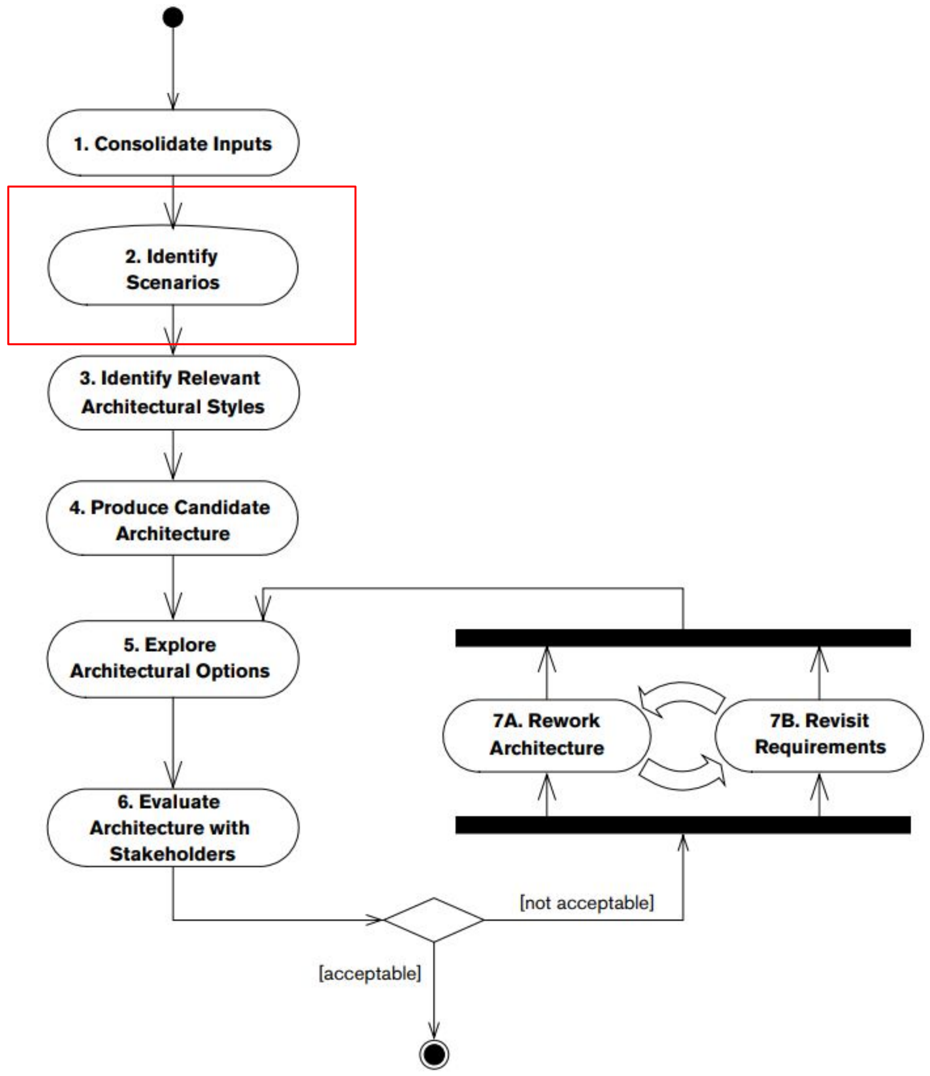
Identifying and Using Scenarios

CSCE 742 - Lecture 7 - 09/25/2018

The Architecture Definition Process



The Architecture Definition Process



1. Consolidate Inputs
2. Identify Scenarios
3. Identify Relevant Architectural Styles
4. Produce Candidate Architecture
5. Explore Architectural Options
6. Evaluate Architecture with Stakeholders
7A. Rework Architecture
7B. Revisit Requirements
[not acceptable]
[acceptable]

Scenarios

- To ensure you meet stakeholder demands, continually consider how your ideas will work in practice.
- An **architectural scenario** is a well-defined description of an interaction between an external entity and the system.
    - It defines the event that triggers the scenario, the interaction initiated by the external entity, and the response required of the system.
    - Similar to use cases or user stories, but examines both quality and functionality.

Scenarios

Capture a range of architectural requirements:

- A set of interactions with users to which a system must respond.
- Processing in response to timed events.
- Peak load situations that could occur.
- Regulator demands.
- Failure response.
- A change that a maintainer might make.
- Any situation that the design must handle.

Scenario Types

- **Functional Scenarios** define how the system responds to external stimuli.
  - Users initiate transactions, AIR call or data sent through an interface, timed events.
- **System Quality Scenarios** define how the system should react in order to exhibit quality properties.
  - Ability to be modified to provide new functionality, to cope with peak load, to protect critical information.

Scenario Usage

- Provide input to architecture definition.
  - Help flesh out and find missing requirements.
- Evaluate the architecture
  - Force description of execution paths through system
  - Find missing/incompatible interfaces.
- Communicate with stakeholders
  - Concrete, easy to understand.
- Drive the testing process
  - Help prioritize testing efforts.

Identifying Scenarios

- Another format for requirements.
- Useful for “task-focused” requirements.
- Less useful for all-purpose invariants.
    - Shall statement: “System shall tolerate any single component failure without loss of service”
    - Scenario: “What if component X fails? What if component Y fails?”
- Require some effort to think about and write.
- Start from overview and fill in most important scenarios.

Functional Scenarios

Functional Scenario Format

- Overview
  - Brief description of what the scenario illustrates.
- System State
  - State of system before the scenario starts.
- System Environment
  - Significant observations about the environment that the system is running in.
- External Stimulus
  - The event that sets off the scenario.
- Required System Response
  - How should the system respond?

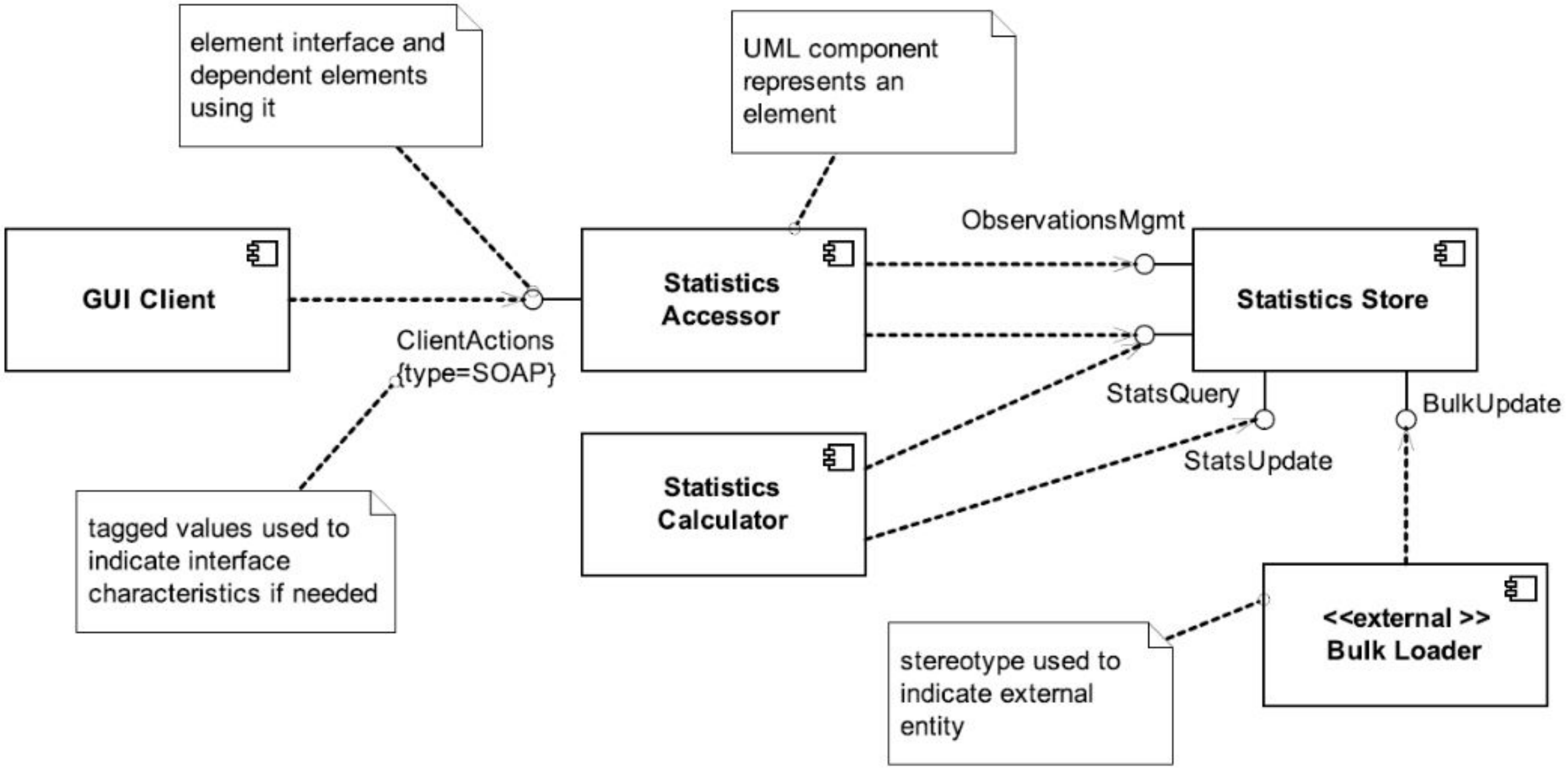
Functional Scenario Format

- External stimulus should describe both the **actor** making a request and **action**.
    - Actor: the user, environmental stimulus such as a failure or timer, or external system.
    - Action: the request, event, or activity.
- Required system response should describe both how the system responds and a **response measure**.
    - Success or failure criterion for the response.

Example: Statistics Processing

- Raw data is loaded into a database.
- Derived statistics are calculated automatically based on the data.
- Statisticians view the data and make reports.
- Clients access statistics and make deductions that are checked manually.

Functional View


element interface and dependent elements using it
UML component represents an element
GUI Client
Statistics Accessor
ObservationsMgmt
Statistics Store
ClientActions {type=SOAP}
StatsQuery
BulkUpdate
Statistics Calculator
StatsUpdate
tagged values used to indicate interface characteristics if needed
<<external >> Bulk Loader
stereotype used to indicate external entity

Functional Scenario

Incremental Statistics Update

- **Overview:** How the system deals with a change to the existing base data.
- **System State:** Summary statistics already exist for the sales quarter that the incremental statistics refer to. The system's databases have enough space to cope with the processing required for this update.
- **System Environment:** The deployment environment is operating normally, without problems.
- **External Stimulus:** Update to sales transactions for the previous quarter arrives via the Bulk Data Load external interface.
- **Required System Response:** The incoming data should automatically trigger background statistical processing to update the summary statistics for the affected quarter to reflect the updated sales transaction data. The old summary statistics should stay available until the new ones are ready.

Quality Scenarios

Quality Scenario Format

- Overview
  - Brief description of what the scenario illustrates.
- System State
  - Aspects of the state that affect quality
  - (i.e., information stored in the system)
- System Environment
  - Significant observations about the environment that the system is running in.
- External Stimulus
  - Environmental factors that initiate the scenario.
  - (i.e., infrastructure changes or failures, security attacks, etc.)
- Required System Response
  - How should it respond (from a quantifiable point of view)?
  - (i.e., how should it handle a defined increase in requests)?

Examples - Statistics Processing

Failure in Summary Database Instance

- **Overview:** How system behaves when database writes fail.
- **System state:** N/A
- **System environment:** The deployment environment is working correctly.
- **External Stimulus:** While writing summary statistics to the database, the system receives an exception indicating that the write failed (e.g., the database is full).
- **Required system behavior:** The system should immediately stop processing the statistics set it is working on and leave any work in progress behind. The system should log a fatal message to the operational console monitoring system and shut down.

Examples - Statistics Processing

Daily Data Update Increases in Size

- **Overview:** How the system's end-of-day processing behaves when regular data volumes are suddenly greatly exceeded.
- **System state:** The system has summary statistics in its database for data that has been processed, and the system's processing elements are lightly loaded at the current rate of system load.
- **System environment:** The deployment environment is working correctly, and data is arriving at a steady rate of 1,000 to 1,500 items per hour.
- **External Stimulus:** The data update rate on a particular day suddenly increases to 4,000 items per hour.
- **Required system behavior:** When the end-of-day processing starts, the system should process that day's data set for a period until the processing time exceeds a system-configurable limit. At that point, the system should stop processing the data set, discard work in process, leave the previous set of summary statistics in place, and log a diagnostic message (including cause and action taken) to the operational console monitoring system.

Examples - Statistics Processing

Additional Summary Dimension Required

- **Overview:** How the system can cope with the need to extend the statistical processing provided.
- **System state:** N/A
- **System environment:** The deployment environment is operating normally, as initially delivered.
- **External Stimulus:** The need arises to support a new statistical dimension in the summary statistics to summarize sales by type of payment option used.
- **Required system behavior:** The development team should be able to add the required processing to provide the new statistical dimension without making any changes to the overall system structure (i.e., changing any interelement interfaces or interactions) and with a total effort of fewer than 4 person-weeks.

Performance Quality Scenarios

- Measure system performance (as opposed to user performance).
- Begins with an event arriving at the system.
  - Responding requires resources (including time) to be consumed.
- Arrival pattern for events can be:
  - Periodic (at regular time intervals)
  - Stochastic (events arrive according to a distribution)
  - Sporadic (unknown timing, but known properties)
    - "No more than 600 per minute"
    - "At least 200 ms between arrival of two events"

Performance Quality Scenarios

- Response measurements include:
    - **Latency:** The time between the arrival of the stimulus and the system's response to it.
    - **Deadlines in processing:** Points where processing must have reached a particular stage.
    - **Throughput:** Usually number of transactions the system can process in a unit of time.
    - **Response Jitter:** The allowable variation in latency.
    - **Number of events not processed** because the system was too busy to respond.

Performance Quality Scenarios

- For real-time systems (i.e., embedded devices), measurements are absolute.
  - Look at worst-case scenario.
- For non-real-time systems, measurements should be probabilistic.
  - 95% of the time, the response should be N.
  - 99% of the time, the response should be M.

Generic Performance Scenario

- **Overview:** Description of the scenario.
- **System/environment state:** The system can be in various operational modes, such as normal, emergency, peak load, or overload.
- **External Stimulus:** Stimuli arrive from external or internal sources. The stimuli are event arrivals. The arrival pattern can be periodic, stochastic, or sporadic, characterized by numeric parameters.
- **Required system behavior:** The system must process the arriving events. This may cause a change in the system environment (e.g., from normal to overload mode).
- **Response measure:** The response measures are the time it takes to process the arriving events (latency or a deadline), the variation in this time (jitter), the number of events that can be processed within a particular time interval (throughput), or a characterization of the events that cannot be processed (miss rate).

Bad Performance Scenario

- **Overview:** How the server handles concurrent requests with graceful response times.
- **System/environment state:** Application is packaged and deployed on the server. The server process is up and ready to serve requests.
- **External Stimulus:** Concurrent requests arrive in high volume.
- **Required system behavior:** Server spawns new threads and handle each request concurrently based on resources configured (like available memory, CPU speed etc.)
- **Response measure:** Server successfully handles all requests.

Good Performance Scenario

- **Overview:** Check system responsiveness for adding items to shopping cart under normal operating conditions.
- **System/environment state:** Normal load is defined as deployment environment with no failures and less than 20 customer requests per second. System is communicating over good internet connection to acceptable client (see glossary for expected internet / client specifications).
- **External Stimulus:** Customer adds product to shopping cart.
- **Required system behavior:** Web page refreshes. Icon on right side of web page displays last item added to cart. If item is out of stock, cart icon has exclamation point overlay on top of cart icon.
- **Response measure:** In 95% of requests, web page is loaded and displayed to user within 1 second. In 99.9% of requests, web page is loaded and displayed to user within 5 seconds.

Usability Quality Scenarios

- Measure user performance (as opposed to system). Scenarios center around:
  - Learning system features: What can the system do to make learning easier?
  - Using a system effectively: What can the system do to make the user more efficient?
  - Minimizing impact of errors: What can the system do so a user error has minimal impact?
  - Adapting to user needs: Can the system adapt to make a task easier?
  - Increasing confidence and satisfaction: What does the system do to provide feedback?

Generic Usability Scenario

- **Overview:** Description of the scenario.
- **System/environment state:** The user actions with which usability is concerned occur at runtime or at system configuration time.
- **External Stimulus:** The end user is the source of the stimulus for usability. The stimulus is that the end user wishes to use a system efficiently, learn to use the system, minimize the impact of errors, adapt the system, or configure the system.
- **Required system behavior:** The system should either provide the user with the features needed or anticipate the user’s needs.
- **Response measure:** The response is measured by 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.

Bad Usability Scenario

- **Overview:** The developer creates a working application (using Rails) that meets simple data storage and retrieval requirements.
- **System/environment state:** Rails is installed and working properly. No such application already exists to meet the data storage requirements.
- **External Stimulus:** The user identifies the requirements of the application and begins development.
- **Required system behavior:** Rails automatically generates and configures the majority of files necessary for the application to run and encapsulates commonly performed web development tasks.
- **Response measure:** Files are created in standard location.

Good Usability Scenario

- **Overview:** Developer adds RESTful access to domain object that previously did not support it.
- **System/environment state:** Ruby and Ruby on Rails installed. Developer has 2 weeks experience with RoR.
- **External Stimulus:** Developer adds RESTful access to Ruby domain object.
- **Required system behavior:** Developer successfully adds URIs for object using both RoR and vanilla Ruby, performing RoR task first. GET on URI displays current value of domain object and PUT on URI correctly modifies object.
- **Response measure:** In 95% of all instances, RoR solution requires less effort (across all of the following measures: SLOC manually written / time to write / count of files manually modified) than "vanilla" Ruby solution.

Availability Quality Scenarios

- The ability of the system to mask or repair faults such that the outage period does not exceed a required value over a time period.
- Measure how the system responds to failure.
    - When the system breaks, how long does it take to resume normal operation?
- Stimuli should always be a failure.
- Response measures should always include a measure of availability:
    - availability percentage, time to detect or repair fault, time system in degraded mode, no down time, etc.

Availability Quality Scenarios

- Scenarios must distinguish physical failures in the system and the software's perception of the failure.
  - Do not assume software is omniscient.
- Scenarios tend to deal with:
  - Failure of a physical component or external system.
  - Reconfiguration of the physical system.
  - Maintenance or reconfiguration of the software.

Generic Availability Scenario

- **Overview:** Description of the scenario.
- **System/environment state:** The state of the system when the fault or failure occurs may also affect the desired system response. If the system has already failed and is not in normal mode, it may be desirable to shut it down. If this is the first failure, degradation of response time or functions may be preferred.
- **External Stimulus:** Differentiate between internal and external origins of failure because desired system response may be different. Stimuli is an *omission* (a component fails to respond to an input), a *crash* (component repeatedly suffers omission faults), *timing* (a component responds but the response is early or late) or *response* (a component responds with an incorrect value).

Generic Availability Scenario

- **Required system behavior:** There are a number of possible reactions to a failure. Fault must be detected and isolated before any other response is possible. After the fault is detected, the system must recover from it. Actions include logging the failure, notifying selected users or other systems, taking actions to limit the damage caused by the fault, switching to a degraded mode with either less capacity or less function, shutting down external systems, or becoming unavailable during repair.
- **Response measure:** Can specify an availability percentage, or it can specify a time to detect the fault, time to repair the fault, times or time intervals where system must be available, or duration for which the system must be available.

Bad Availability Scenario

Handling Isolation

- **Overview:** How the server manages multiple applications with desired isolation.
- **System/environment state:** Multiple applications (unrelated) are deployed on the server. The server is up and running.
- **External Stimulus:** User(s) establishes session with each of these applications.
- **Required system behavior:** Server deploys each application in its own context which can be configured to share or not share any application specific data between them.
- **Response measure:** Applications are isolated.

Good Availability Scenario

Availability while adding new taps

- **Overview:** How the system handles additional taps being added to the system.
- **System/environment state:** The system is operating normally, without problems.
- **External Stimulus:** A user powers up a new Kegboard on the network with six additional taps.
- **Required system behavior:** The kegboards send init messages to the central Kegbot server. The server interrogates the kegboards and adds the additional taps to the inventory of taps. The system continues to service the existing taps without interruption.
- **Response measure:** There is no interruption of service to existing taps. Within 1 second, the new kegboard is added to the administrative interface on the KegBot web server for administraton configuration.

Good Availability Scenario 2

Web server failure at e-commerce site

- **Overview:** One of the client-facing web servers fails during transmission of client page update.
- **System/environment state:** System is working correctly under normal load. Customer has generated a "add item to shopping cart" post, which was routed to web server <X> in transaction pool.
- **External Stimulus:** Web server <X> crashes during response generation.
- **Required system behavior:** Response page may be corrupted on client browser. Load balancer component no longer receives heartbeat message from web server and so removes it from the pool of available servers after 2s of missed messages, or upon next request sent to the server. Load balancer will remove the server from the pool of available servers. From client's perspective, a page reload will be automatically routed to alternate server by load balancer and page will be correctly displayed.
- **Response measure:** Upon client-side page refresh, client state and display contains state after last transaction. Time for re-routed refresh is equivalent to "standard" refresh (<1 second 95% of the time).

Security Quality Scenarios

- Measure of the system’s ability to protect data from unauthorized access while still providing service to authorized users.
- Scenarios measure response to attack.
  - Stimuli are attacks from external systems/users or demonstrations of policies (log-in, authorization).
- Responses include auditing, logging, reporting, analyzing.
  - Response measures include amount of data loss/compromise, time to detect/mitigate, % of attacks resisted, etc.

Generic Security Scenario

- **Overview:** Description of the scenario.
- **System/environment state:** The attack can come when the system is either online or offline, either connected to or disconnected from a network, either behind a firewall or open to a network, fully operational, partially operational, or not operational.
- **External Stimulus:** The source of the attack may be either a human or another system. It may have been previously identified or may be currently unknown. A human attacker may be from outside the organization or from inside the organization. The stimulus is an attack (unauthorized attempt to display data, change or delete data, access services, change the system’s behavior, or reduce availability).

Generic Security Scenario

- **Required system behavior:** The system should ensure that transactions are such that:
  - Data/services are protected from unauthorized access
  - Data/services are not manipulated without authorization
  - Parties to a transaction are identified and cannot repudiate their involvement
  - Data, resources, and system services will be available for legitimate use.

  The system should also track activities by

  - Recording access or modification and attempts to access data, resources, or services
  - Notifying appropriate entities (people or systems) when an apparent attack is occurring.

Generic Security Scenario

- **Response Measure:** Measures of a system's response include:
  - 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 it took to recover from a successful attack
  - How much data was vulnerable to a particular attack.

Bad Security Scenario

Malicious Control

- **Overview:** How the server restricts damage when someone maliciously gains control over it.
- **System/environment state:** Multiple applications are deployed on the server. The servers are running and serving requests
- **External Stimulus:** One of the applications deployed is breached.
- **Required system behavior:** Server can be configured with different privileges, providing more granular control over their access to system resources and potentially preventing one breached application from allowing access to others.
- **Response measure:** Remaining applications are not breached.

Good Security Scenario

Unsuccessful Authentication

- **Overview:** A user attempts to authenticate but the authentication fails due to unrecognized auth token or due to system unavailability.
- **System/environment state:** There is a valve installed on the tap. There is a flow meter installed on the tap. There is a piezo buzzer installed on the Kegboard. Authentication hardware (RFID or one-wire) is installed on the Kegboard. There is no pour in progress. The system is operating normally, without problems.
- **External Stimulus:** A user presents an auth token to the authentication hardware on the Kegboard.
- **Required system behavior:** The auth token is unrecognized, and the valve is not opened. An audible sound is played from the piezo buzzer, indicating authentication failure.
- **Response measure:** No beer is dispensed.

Activity - Airport Parking

- You will develop a new automated parking system at the CAE airport.
- In this new system:
    - A user can insert their card into the card reader at the ramp entrance.
    - This will record the time they entered airport parking.
    - They then can use the same credit or debit card to pay at an exit lane.
    - The system should be fully automated.
    - The system should also support ticketed parking
        - User receives a ticket and pays either by credit card or cash on exiting.

Applying Scenarios

Activity - Airport Parking

The system will be deployed within the physical architecture of the airport parking garage, incorporating:

- Entrance Kiosks
  - Card dispensers
  - Credit card reader for e-park
  - Card reader for contract parking
- Parking ramp levels
  - Signage {FULL / not full}
  - Entry gates
- Exit Kiosks
  - Signage: {OPEN / ePark ONLY / CLOSED}
  - Staffed Kiosks
  - Automated Kiosks
- Security Cameras
- Hardware for Parking System
  - Dual Server w/Failover (can switch in event of failure)
  - Clustered DB
  - Storage Area Network

Activity - Airport Parking

1. Describe two functional scenarios
2. Describe two quality scenarios

Activity - Airport Parking

- Quality Scenario for Performance
  - Time to exit ramp
- Quality Scenario for Availability
  - Exit Kiosk Malfunction
  - Loss of Server
  - Loss of Connection to Credit Card Processing
- Quality Scenario for Usability
  - Entry / Exit
  - Administrative Interface: Lockdown, Full open, Spaces remaining, etc.

“Good” Scenarios

- Credible
  - Describes a realistic scenario.
- Valuable
  - Can be directly used during architectural definition.
- Specific
  - Addresses a single, concrete situation.
- Precise
  - Intended user of scenario should be clear about the described situation and response.
- Comprehensible
  - Writing should be unambiguous and free of jargon.

Effective Scenario Use

- Identify a focused scenario set
  - Too many scenarios can be distracting.
  - Prioritize no more than 15-20.
- Use distinct scenarios
  - Avoid having multiple scenarios centered around near-identical events. They are redundant.
  - Consider demands placed on the system.
- Use scenarios early
  - Most impactful early in development to focus architecture and design activities on most important aspects of the system.

Effective Scenario Use

- Include system quality scenarios!
  - Great potential for investigating, validating, and understanding quality properties.
  - You will need to augment stakeholder-provided scenarios to consider quality.
- Include failure scenarios!
  - Consider the important failure cases and identify scenarios to address them.
- Involve stakeholders closely
  - Stakeholders may reveal scenarios you didn't consider and have differing priorities than you do.

Checklist (Food for Thought)

- Have you defined a wide enough range of system quality scenarios?
- Have you defined a wide enough range of failure and exception scenarios?
- Have you prioritized your scenarios by stakeholder importance and risk?
- Have you kept the number of scenarios down to a manageably small level?
- Have you reviewed the responses and behaviors with stakeholders?

Checklist (Food for Thought)

- Have you included scenarios that you think will be valuable (based on previous experience) as well as those nominated by your stakeholders?
- Are all scenarios cataloged and named?
- Have you made sure gaps or mistakes in the requirements are addressed?
- Have you revised mismatches between required and actual response or behavior in the architectural design appropriately?

Key Points

- Defining and applying scenarios ensures that your architecture will exhibit required functionality and behavior.
    - **Functional scenarios** define how the system responds to external stimuli.
    - **System quality scenarios** define how the system responds to environmental factors that affect quality properties.
    - The specification should include the initial system state and environment, external stimulus or environment changes, and the required system response (and how to measure it).

Next Time

- The Functional Viewpoint
  - Sources: Rozanski & Woods: ch. 17

- Homework:
  - Assignment 1
    - Due 10/02
  - Project Part 2
    - Due 10/11

Project - Part 2

- Setting the system scope and requirements
  - Explain the purpose, scope, and audience.
  - Identify groups of stakeholders.
  - Describe important functional and non-functional requirements of the system.
  - Define the scope using a context diagram + explanations.
  - Create five functional and five quality-based scenarios.
  - Rate the viewpoints and perspectives in terms of importance.
  - Define change cases.