Outline for Today

- System quality attributes
- Quality attributes scenarios
- Achieving quality attributes through tactics
- Group work
The Problem

How to bridge the gap between requirements and code?

The Traditional Answer

- Ad hoc
- Requires gurus
- Unpredictable
- Costly

A Miracle Happens!
Software Development Methods

- Increased predictability
- Some design guidance
- BUT
- Limited applicability
- Still requires gurus
- Weak support for design analysis

The Role of Software Architecture

- Composition of large-scale components
- System-level abstractions
- Reuse of system-level design idioms
## Quality Attributes

- QA should be the deriving force behind arriving at a suitable architecture.

- Architecture is typically the first stage of development where QA are addressed.

- Reminder:
  - Architecture: mapping of functionality into structures such that the organization of structures is what helps achieve QA of interest.

## Functionality vs. QA

- Orthogonal?
  - Discuss (5 min)

- Functionality can exist with any structure (independent of structure).

- When there are other attributes that matter, then we restrict the allocation of functions to structures:
  - e.g. Allocation of components to team of developers.

- Architecture is critical to arriving at QA.

- QA should be designed in and the architecture must be evaluated to discover to which degree these QA might be achieved.

- Architecture simply provides the foundation for achieving QAs but is worthless if attention to details is absent in subsequent development practices.
Quality Attributes in This Class

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

Availability

- Concerned with system failures and duration of those failures.
- System failure means … when the system does not provide the service for which it was intended
Modifiability

- Cost of change, both in time and money.

Performance

- Timeliness; events occur and the system must respond in a timely fashion.
- Also about consumption of system resources
Security

- The ability of the system to prevent or resist unauthorized access while providing access to legitimate users.

Testability

- Refers to the ease with which the software can be made to demonstrate its faults or lack thereof.
- To be testable, the system must control inputs and be able to observe outputs.
Usability

- How easy it is for the user to accomplish tasks and what support the system provides for the user to accomplish these tasks.
  - Dimensions:
    - Learning system features
    - Using the system efficiently
    - Minimizing the impact of errors
    - Adapting the system to the user’s needs
    - Increasing confidence and satisfaction

We need a way to identify and specify verifiable/quantifiable QAs
Quality Attribute Scenarios - 1

- There’s a need for stating and documenting non-functional requirements
- QA-Scenario Consists of the following:
  - Source of stimulus
  - Stimulus
  - Environment
  - Artifact
  - Response
  - Response Measure

Quality Attribute Scenarios - 2

- Source of stimulus: what generates the stimulus (e.g., human, computer system, etc.)
- Stimulus: condition that needs to be considered when arrived at the system
- Environment: conditions of the system when the stimulus arrives (e.g., normal, overload, etc.)
- Artifact: element of the system that is stimulated (e.g., whole system, processor, etc.)
- Response: activity undertaken by the system/developers after arrival of stimulus
- Response measure: when the response occurs it should be measurable so that the requirement can be tested
## Availability Scenarios

<table>
<thead>
<tr>
<th>Availability</th>
<th>Possible values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source</td>
<td>Internal to system, external to system</td>
</tr>
<tr>
<td>Stimulus</td>
<td>Crash, omission, timing, no response, incorrect response</td>
</tr>
<tr>
<td>Artifact</td>
<td>System’s processors, communication channels, persistent storage</td>
</tr>
<tr>
<td>Environment</td>
<td>Normal operation; degraded (failsafe) mode</td>
</tr>
<tr>
<td>Response</td>
<td>Log the failure, notify users/operators, disable source of failure, continue (normal/degraded)</td>
</tr>
<tr>
<td>Response Measure</td>
<td>Time interval available, availability%, repair time, unavailability time interval</td>
</tr>
</tbody>
</table>

---

### Artifacts
- System’s processors
- Communication channels
- Persistent storage

### Sources
- Internal to system
- External to system

### Stimuli
- Crash
- Omission
- Timing
- No response
- Incorrect response

### Environments
- Normal operation
- Startup
- Shutdown
- Repair mode
- Degraded (failsafe) mode
- Overloaded operation

### Responses
- Prevent the failure
- Log the failure
- Notify users/operators
- Disable source of failure
- Temporarily unavailable
- Continue (normal / degraded)

### Measures
- Time interval available
- Availability %
- Detection time
- Repair time
- Degraded mode
- Unavailability time interval
### Modifiability Scenarios

<table>
<thead>
<tr>
<th>Availability</th>
<th>Possible values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source</td>
<td>End-user, developer, system-administrator</td>
</tr>
<tr>
<td>Stimulus</td>
<td>Add/delete/modify functionality or quality attribute</td>
</tr>
<tr>
<td>Artifact</td>
<td>GUI, platform, environment, design document, code…</td>
</tr>
<tr>
<td>Environment</td>
<td>At runtime, compile time, build time, design-time</td>
</tr>
<tr>
<td>Response</td>
<td>Locate places in architecture for modifying, modify, test modification, deploy modification</td>
</tr>
<tr>
<td>Response Measure</td>
<td>Cost in effort, money, time, extent affects other system functions or qualities</td>
</tr>
</tbody>
</table>

![Modifiability Scenarios Diagram](image)
Performance Scenarios

<table>
<thead>
<tr>
<th>Performance</th>
<th>Possible values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source</td>
<td>A number of sources both external and internal</td>
</tr>
<tr>
<td>Stimulus</td>
<td>Periodic events, sporadic events, stochastic events</td>
</tr>
<tr>
<td>Artifact</td>
<td>System, component</td>
</tr>
<tr>
<td>Environment</td>
<td>Normal mode; overload mode; reduced capacity mode</td>
</tr>
<tr>
<td>Response</td>
<td>Process stimulus; change level of service</td>
</tr>
<tr>
<td>Response Measure</td>
<td>Latency, deadline, throughput, capacity jitter, miss rate, data loss</td>
</tr>
</tbody>
</table>
### Security Scenarios

<table>
<thead>
<tr>
<th>Security</th>
<th>Possible values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source</td>
<td>User/system who is legitimate/imposter/unknown with full/limited access</td>
</tr>
<tr>
<td>Stimulus</td>
<td>Attempt to display/modify data; access services</td>
</tr>
<tr>
<td>Artifact</td>
<td>System services, data</td>
</tr>
<tr>
<td>Environment</td>
<td>Normal operation; degraded (failsafe) mode</td>
</tr>
<tr>
<td>Response</td>
<td>Authenticate user; hide identity of user; grant/block access; encrypt data; detect excessive demand…</td>
</tr>
<tr>
<td>Response Measure</td>
<td>Time /effort/resources to circumvent security measures with probability of success</td>
</tr>
</tbody>
</table>

**Artifact**
- System services
- Data within the system
- Component/resource of the system
- Data produced/consumed by the system

**Source**
- Identified user
- Unknown user
- Hacker from outside the organization
- Hacker from inside the organization

**Stimulus**
- Attempt to display data
- Attempt to modify data
- Attempt to delete data
- Access system services
- Change system’s behavior
- Reduce availability

**Environment**
- Normal mode
- Overload mode
- Reduced capacity mode
- Emergency mode
- Peak mode

**Response**
- Process events
- Change level of service

**Measure**
- Latency
- Deadline
- Throughput
- Jitter
- Miss rate
- Data loss
### Testability Scenarios

<table>
<thead>
<tr>
<th>Testability</th>
<th>Possible values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source</td>
<td>Unit developer, increment integrator, system verifier, client acceptance tester, system user</td>
</tr>
<tr>
<td>Stimulus</td>
<td>Analysis, architecture, design, class, subsystem integration, system delivered</td>
</tr>
<tr>
<td>Artifact</td>
<td>Piece of design, piece of code, complete system</td>
</tr>
<tr>
<td>Environment</td>
<td>At design time, at development time, at compile time, at deployment time</td>
</tr>
<tr>
<td>Response</td>
<td>Provide access to state data values, observes results, compares expected to actual results</td>
</tr>
<tr>
<td>Response Measure</td>
<td>% coverage; prob. of failure; time to perform tests; length of time to prepare test environment</td>
</tr>
</tbody>
</table>

### Testability Scenarios Diagram

- **Artifact**: Portion of the system being tested
- **Source**
  - Unit tester
  - Integration tester
  - System tester
  - Acceptance tester
  - End user
  - Automated testing tools
- **Stimulus**: Execution of tests due to completion of code increment
- **Environment**
  - Design time
  - Development time
  - Compile time
  - Integration time
  - Deployment time
  - Run time
- **Response**
  - Execute test suite & capture results
  - Capture cause of fault
  - Control & monitor state of the system
- **Measure**
  - Effort to find fault
  - Effort to achieve coverage %
  - Probability of fault being revealed by next test
  - Time to perform tests
  - Effort to detect faults
  - Length of longest dependency chain
  - Time to prepare test environment
Usability Scenarios

<table>
<thead>
<tr>
<th>Security</th>
<th>Possible values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source</td>
<td>End user</td>
</tr>
<tr>
<td>Stimulus</td>
<td>Wants to: learn system, use system, recover from errors, adapt system, feel</td>
</tr>
<tr>
<td></td>
<td>comfortable</td>
</tr>
<tr>
<td>Artifact</td>
<td>System</td>
</tr>
<tr>
<td>Environment</td>
<td>At runtime, or configure time, install-time</td>
</tr>
<tr>
<td>Response</td>
<td>Provides requested features, anticipate user's needs</td>
</tr>
<tr>
<td>Response</td>
<td>Task time, number of errors, number of tasks accomplished, user satisfaction,</td>
</tr>
<tr>
<td>Measure</td>
<td>gain of user knowledge, amount of time/data lost</td>
</tr>
</tbody>
</table>

Usability Scenarios

Source
- End user (possibly special role)

Stimulus
- Use the system efficiently
- Learn to use the system
- Minimize impact of errors
- Adapt the system
- Configure the system

Environment
- Runtime
- Configuration time

Response
- Provide features needed
- Anticipate the user's needs

Measure
- Task time
- Number of errors
- Number of tasks accomplished
- User satisfaction
- Gain of user knowledge
- Ratio of successful operations to total operations
- Amount of time / data lost when error occurs
### Example Availability Scenario

<table>
<thead>
<tr>
<th>Availability</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Source</td>
<td>Heartbeat monitor</td>
</tr>
<tr>
<td>Stimulus</td>
<td>Server unresponsive</td>
</tr>
<tr>
<td>Artifact</td>
<td>Process</td>
</tr>
<tr>
<td>Environment</td>
<td>Normal operation</td>
</tr>
<tr>
<td>Response</td>
<td>Inform operator, switch to backup server and continue to operate</td>
</tr>
<tr>
<td>Response Measure</td>
<td>Zero system downtime</td>
</tr>
</tbody>
</table>

### Sample Availability Scenario

<table>
<thead>
<tr>
<th>Availability</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Source</td>
<td>Heartbeat Monitor</td>
</tr>
<tr>
<td>Stimulus</td>
<td>Server Unresponsive</td>
</tr>
<tr>
<td>Artifact</td>
<td>Process</td>
</tr>
<tr>
<td>Environment</td>
<td>Normal Operation</td>
</tr>
<tr>
<td>Response</td>
<td>Inform Operator Continue to Operate</td>
</tr>
<tr>
<td>Response Measure</td>
<td>No Downtime</td>
</tr>
</tbody>
</table>
Example – Web-based Financial System

- Web-based financial system

Users initiate 1000 transactions per minute stochastically under normal operations, and these transactions are processed with an average latency of two seconds.

Question: What type of QA is of concern here?

Performance

Example – continue

Users initiate 1000 transactions per minute stochastically under normal operations, and these transactions are processed with an average latency of two seconds.

- Source of stimulus:
  - Users
- stimulus:
  - Initiate transactions
- Environment:
  - Normal operation
- artifact:
  - System
- Response:
  - All transactions are processed
- Response measure:
  - Average latency of 2 sec
Example – Modifiability Scenario

- Web-based financial system

A developer wishes to change the user interface to make a screen’s background color blue. This change will be made to the code at design time. It will take less than three hours to make and test the change and no side effect changes will occur in the behavior.
A developer wishes to change the user interface to make a screen’s background color blue. This change will be made to the code at design time. It will take less than three hours to make and test the change and no side effect changes will occur in the behavior.

- Source of stimulus:
  - Developer
- stimulus:
  - Changes to GUI
- Environment:
  - Design phase
- artifact:
  - Source code
- Response:
  - Modification is made with no side effect
- Response measure:
  - < 3 hrs

Semester Teams

- T1: Randy, Max, Kassie, Jose
- T2: Martin, Kristian, Kim, Michael (M)
- T3: Christian, Alejandro, Nadezda, Emmanuel
- T4: Damian, Adeel, Almaraz
- T5: Edgar, Mitchell, Michael (E)
Group Exercise (20 min)

- Come up with a sample scenario for the quality attribute assigned to your team
  - Team 1: Availability
  - Team 2: Performance
  - Team 3: Security
  - Team 4: Modifiability
  - Team 5: Testability