Defining Composite Critical Scenarios for the Development of Large Scale System Architecture Using an SEI's ADD-based Framework

Aldo Dagnino
ABB Inc.
US Corporate Research Center
Raleigh, NC

SATURN 2008
Fourth SEI Software Architecture Technology User Network Workshop
April 28-May 1 of 2008
Pittsburgh, Pennsylvania
ADD is a methodology used to define a system architecture that bases the decomposition process on the quality attributes the system (software) has to fulfill.

The architectural design using the ADD methodology can begin when the architectural drivers are known with some level of confidence.

In ADD Tactics and Architectural patterns are selected to satisfy a set of quality attributes within a critical scenario that provides context for those quality attributes.
Steps for Creating a Software Architecture

1. Define Business Case for the System
   This step allows to define the main business drivers and hence set priorities

2. Understand the Requirements
   The functionality of the system and its quality attributes are the source of functional and non-functional requirements, which are needed to define the architecture

3. Leveraging Quality Attribute Scenarios
   Encapsulate functional, non-functional requirements, and constraints into scenarios used to evaluate architecture options

4. Creating the Architecture Option(s)
   The functionality of the system and its quality attributes are the source of functional and non-functional requirements, which are needed to define the architecture

5. Documenting and Communicating Architecture Options

6. Analyzing or Evaluating the Architecture Options
   Evaluating architecture options for the qualities it supports

7. Implementing the Systems Based on the Architecture

8. Ensuring that the Implementation Conforms to the Architecture
Prioritizing Business Goals

Stages

Business Considerations

Define Business Goals

Quality Attributes

Prioritize Business Goals

Use Cases

Non Functional Requirements

Prioritized Non Functional Requirements

Functional Requirements

Quality Attribute or Critical Scenarios with all NFRs from prioritized Quality Attributes

Other products, customers, market, legacy systems, product managers, etc....
Define Business Case for the System
This step allows to define the main business drivers and hence set priorities

Understand the Requirements
The functionality of the system and its quality attributes are the source of functional and non-functional requirements, which are needed to define the architecture

Leveraging Quality Attribute Scenarios
Encapsulate functional, non-functional requirements, and constraints into scenarios used to evaluate architecture options

Creating the Architecture Option(s)
The functionality of the system and its quality attributes are the source of functional and non-functional requirements, which are needed to define the architecture

Documenting and Communicating Architecture Options

Analyzing or Evaluating the Architecture Options
Evaluating architecture options for the qualities it supports

Implementing the Systems Based on the Architecture

Ensuring that the Implementation Conforms to the Architecture
Business Goals

- Prioritized Business Goals
  - Business goals associated with the project are elicited from selected project stakeholders
  - Business goals are prioritized for stakeholders to guide architectural tradeoffs

- Example of prioritized business goals:
  - Lower commissioning costs by 30%
  - Ensure system is available 99.99%
  - Maintain current system performance
Mapping Business Goals and Quality Attributes

**Business Goal**
- Lower commissioning costs by xx%
- Ensure system is available 99.9%
- Maintain current system performance

**Quality Attributes**
- Commissionability
- Availability
- Performance
## Business Goals and Quality Attributes

<table>
<thead>
<tr>
<th>Business Goals</th>
<th>Software Qualities</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lower commissioning (customization, installation, and configuration) and recurring system setup costs by xx% in new System</td>
<td><strong>Commissionability</strong></td>
</tr>
<tr>
<td>Ensure data integrity in system</td>
<td><strong>Data Integrity</strong></td>
</tr>
<tr>
<td>Ensure System is available 99.99%</td>
<td><strong>Availability</strong></td>
</tr>
<tr>
<td>Maintain current system performance</td>
<td><strong>Performance</strong></td>
</tr>
</tbody>
</table>

---

The diagram on the right side of the page illustrates the relationships between different objects and classes, which are not described in the text. The object class diagram includes labels and arrows indicating connectivity and direction, but the specifics of the diagram are not detailed in the provided excerpt.
Understand Requirements

Define Business Case for the System
This step allows to define the main business drivers and hence set priorities

Understand the Requirements
The functionality of the system and its quality attributes are the source of functional and non-functional requirements, which are needed to define the architecture

Leveraging Quality Attribute Scenarios
Encapsulate functional, non-functional requirements, and constraints into scenarios used to evaluate architecture options

Creating the Architecture Option(s)
The functionality of the system and its quality attributes are the source of functional and non-functional requirements, which are needed to define the architecture

Documenting and Communicating Architecture Options

Analyzing or Evaluating the Architecture Options
Evaluating architecture options for the qualities it supports

Implementing the Systems Based on the Architecture

Ensuring that the Implementation Conforms to the Architecture
Develop Non-Functional Requirements

**Quality Attribute**
System Quality

**Commissionability**

**Customer-related Non Functional Requirements**
Associated/derived from Quality Attribute

- NFR0250 - No source code change for a specific customer deployment
- NFR0260 - Complete software build (compilation of source code) within x hours
- NFR0270 - Complete system installation (including OS) within y hours (human interaction)
<table>
<thead>
<tr>
<th>Business Goals</th>
<th>Driving Non-Functional Requirements</th>
<th>Software Qualities</th>
</tr>
</thead>
</table>
| Lower commissioning (customization, installation, and configuration) and recurring system setup costs by xx% for new system | NFR0250 - No source code change for a specific customer deployment  
NFR0260 - Complete software build (compilation of source code) within x hours  
NFR0270 - Complete system installation (including OS) within y hours (human interaction) | Commissionability |
| Ensure 100% data integrity in system | NFR0110 - Ensure data integrity across integrated systems  
NFR0100 - Provide controlled access to its data stores  
NFR0030 - Prevent data corruption and data loss | Data Integrity |
| Ensure system is available 99.9% | NFR0750 - Be available 24*7 (RFPs: 99.9%)  
NFR0720 - Failover for XXX and YYY functions shall be completed within xx seconds  
NFR0630 - The architecture must support hot failover  
NFR0740 - Loose coupling of components  
NFR0700 - Complete cold startup (from power-off condition) shall be completed with all functions scheduled for execution within xx time | Availability |
| Maintain current system performance | NFR0820 - Events processing speed  
NFR0870 - Display update upon change max exec time is xx sec  
NFR0030 - Prevent data corruption and data loss  
NFR0890 - Process events  
NFR0910 - Execution time of applications  
NFR0900 - Time to run sequence of applications from Topology Processing to Contingency Analysis  
NFR0860 - Supervisory control max execution time is 1 sec  
NFR0920 - CPU and memory consumption of applications | Performance |
Leveraging Quality Attribute Scenarios

1. Define Business Case for the System
   This step allows to define the main business drivers and hence set priorities

2. Understand the Requirements
   The functionality of the system and its quality attributes are the source of functional and non-functional requirements, which are needed to define the architecture

3. Leveraging Quality Attribute Scenarios
   Encapsulate functional, non-functional requirements, and constraints into scenarios used to evaluate architecture options

4. Creating the Architecture Option(s)
   The functionality of the system and its quality attributes are the source of functional and non-functional requirements, which are needed to define the architecture

5. Documenting and Communicating Architecture Options

6. Analyzing or Evaluating the Architecture Options
   Evaluating architecture options for the qualities it supports

7. Implementing the Systems Based on the Architecture

8. Ensuring that the Implementation Conforms to the Architecture
Architectural drivers (quality attribute scenarios) include the combination of functional and quality requirements that shape the architecture:

- Define unique functions (as architectural Functional Requirements) of modules in the system
- Select associated Non-functional Requirements
- Quality Attributes or Critical System Scenarios provide the functional context under which Non Functional Requirements are defined
- Architectural patterns that satisfy the critical scenarios are then selected
Quality Attribute Scenarios

- Encapsulate a set of architectural functional and non-functional requirements that uniquely define the system being architected

- Are described by a set of detailed architectural product requirements

- Can incorporate one or more Use Cases
Requirements Development

Quality Attribute Scenario Elements

Source of stimulus → Stimulus → Artifact → Environment → Response → Measure

© ABB, 2008 - 15
Quality Attribute Scenario: Run a Sequence of Applications

Operator → Run an application sequence → System sub-module → Algorithmic convergence → Less than xx minutes

Source of stimulus | Stimulus | Environment | Response

Response Measure
Develop Critical System Scenarios

Critical Scenario
The operator runs a sequence of complex applications

- Operator
- Topology Processor
- State Estimator
- Contingency Analysis

Critical Scenario Diagram:
- If State Estimator converges
  - Start State Estimator
  - Start Contingency Analysis
- If State Estimator does not converge
  - Change input parameters
  - Rerun State Estimator

Customer (Architectural) Requirements
Includes Functional and Non-functional requirements (and any constraints)

- The system shall allow the operator to run the state estimator application
- The system shall allow the operator to run sensitivity analyses
- The system shall allow the operator to run the PS model
- The system shall allow the operator to run a sequence of applications in an “industry acceptable” time etc . . .

The time duration from Topology processing to Contingency Analysis must be less than three (xx) minutes under normal conditions (State Estimator converges)

An application executive subsystem defines dependences and relationships between applications

There are x (x) types of requests for running applications (sequence): from operator, from event, from continuous timer
Another Critical System Scenario

Failover for XXX and YYY functions shall be completed within xy seconds.
Add new element and sub-elements to the system
Integrate with external systems
Process and display events
Run advanced numerical applications
Support hot-failover
Run a sequence of complex applications
Prioritizing Non-Functional Requirements

Critical Scenarios

Other products, customers, market, legacy systems, product managers, etc.

Business Considerations

Define Business Goals

Quality Attributes

Non Functional Requirements

Prioritize Non Functional Requirements

Quality Attribute or Critical Scenarios

Use Cases

Functional Requirements
### Prioritized Non-functional Requirements

<table>
<thead>
<tr>
<th>ID</th>
<th>Requirement</th>
<th>Calculated Avg.</th>
<th>Impact on Architecture</th>
<th>Prototype Importance</th>
<th>10*CalcAvg +ImpactArch</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.2.1.1</td>
<td>The new system will have one code base to ensure the functionalities of XXX system by using common technology and programming language, as well as shared or reusable modules.</td>
<td>5.0</td>
<td>5</td>
<td>5</td>
<td>55</td>
</tr>
<tr>
<td>4.2.4.1</td>
<td>The new system will have consistent mechanisms for integration with other components of overall system.</td>
<td>5.0</td>
<td>5</td>
<td>3</td>
<td>55</td>
</tr>
<tr>
<td>4.6.1.1</td>
<td>The failover of a server component using a backup database completes in no more than xx seconds.</td>
<td>5.0</td>
<td>5</td>
<td>1</td>
<td>55</td>
</tr>
<tr>
<td>4.3.3.2</td>
<td>The system allows the user to perform only the authorized actions.</td>
<td>5.0</td>
<td>5</td>
<td>1</td>
<td>55</td>
</tr>
<tr>
<td>4.2.5.4</td>
<td>The implementation plan shall facilitate the transition of the XXX team to the new development environment.</td>
<td>5.0</td>
<td>5</td>
<td>0.1</td>
<td>55</td>
</tr>
<tr>
<td>4.2.1.2</td>
<td>The impact of changes will be minimized in the new system thanks to a modular design</td>
<td>5.0</td>
<td>3</td>
<td>3</td>
<td>53</td>
</tr>
<tr>
<td>4.1.3.2</td>
<td>Shortcuts to achieve multi-row operations are available (aggregated data). Mass-update.</td>
<td>5.0</td>
<td>3</td>
<td>3</td>
<td>53</td>
</tr>
<tr>
<td>4.2.4.2</td>
<td>The new system will allow for interactive exchange of data between new YYY module and ZZZ module (messaging layer).</td>
<td>4.7</td>
<td>5</td>
<td>5</td>
<td>52</td>
</tr>
<tr>
<td>4.4.1.3</td>
<td>Dialog response time is less than x seconds in average and never more than y seconds.</td>
<td>4.7</td>
<td>5</td>
<td>5</td>
<td>52</td>
</tr>
</tbody>
</table>

---

**Develop Quality Scenarios Using Prioritized NFRs**
Observations

- It is easier for Senior Management to prioritize Business Drivers than non functional requirements.
- Creation of Composite Scenarios provides a richer approach to prototype architectural options.
- Prioritization of business goals automatically prioritizes qualities and non functional requirements.
- Potential for missing important non functional requirements that fall outside the prioritization scheme.
- Analysis may become more complex using Composite Scenarios.