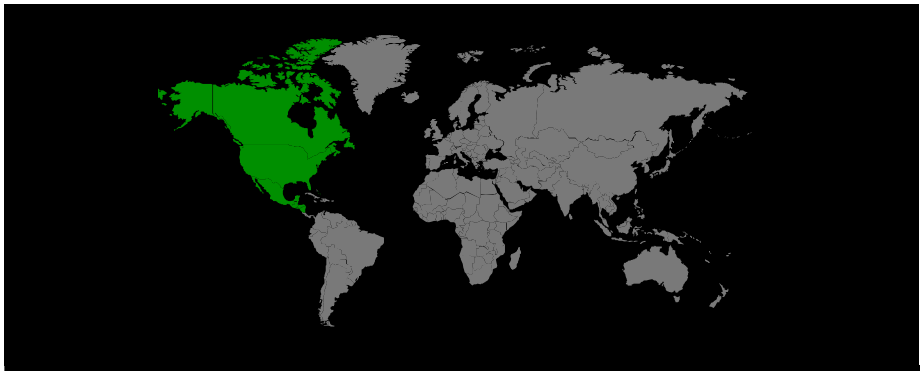


Aldo Dagnino, ABB Corporate Research / Industrial Software Systems

Improving the Definition of Quality Attribute Scenarios by Using Requirements Patterns

SATURN Conference 2009

The audience will…

- Become familiar with patterns that can be used to define non-functional requirements with clarity, precision, and required level of detail
- Learn how to use non-functional requirement patterns to better define quality scenarios that are the basis for the definition of the software architecture

Agenda

- Introduction to Requirements Engineering
- The Software Architecture Process
- Performance Requirements Patterns
- Architectural Scenario for Performance

ABB

Introduction:
Requirements Engineering

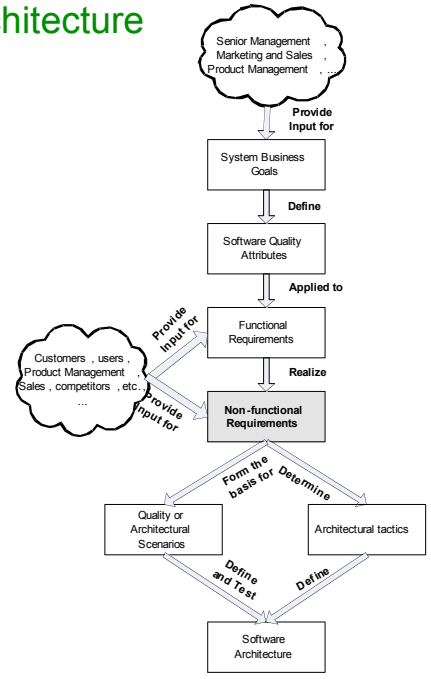
- Requirements Engineering (RE) is a set of activities devoted to understanding the needs of relevant stakeholders; comprehending the context in which the future software will operate
- Requirements analysts typically start with poorly defined and often conflicting ideas on what the system is supposed to do
- When stakeholders define a software-intensive system, it is more natural that they focus on specifying the functional requirements
- Users have also expectations on how well the product will work such as how easy it is to use, how quickly it runs, etc.
- These characteristics are called software quality attributes
- Quality attributes are difficult to define
- Excellent software products exhibit an exquisite balance among competing quality attributes
- Quality attributes drive architectural decisions for the software system

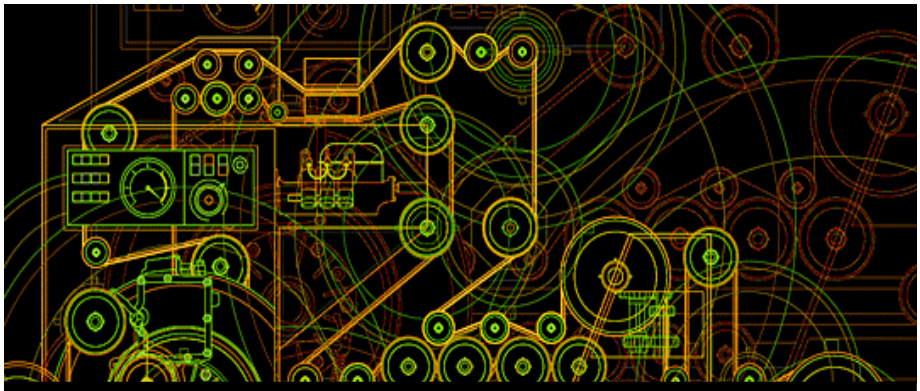
ABB

The Volere Shell ©
Elements

- Requirement number (unique identifier).
- Requirement type (functional, usability, etc.).
- Even/use case number (to cross-reference).
- Description (natural language statement).
- Fit Criterion/criteria (quantified goal that requirement has to meet).
- Rationale (reason behind requirement).
- Source (origin of requirement).
- Customer satisfaction (value on how satisfied a customer would be if this requirement were to be included in the software).
- Customer dissatisfaction (value on how dissatisfied the customer would be if requirement were not included in the software).
- Dependencies (associated requirement).
- Conflicts (conflict requirement ID).
- Supporting materials.
- History (relevant historical information on requirement).

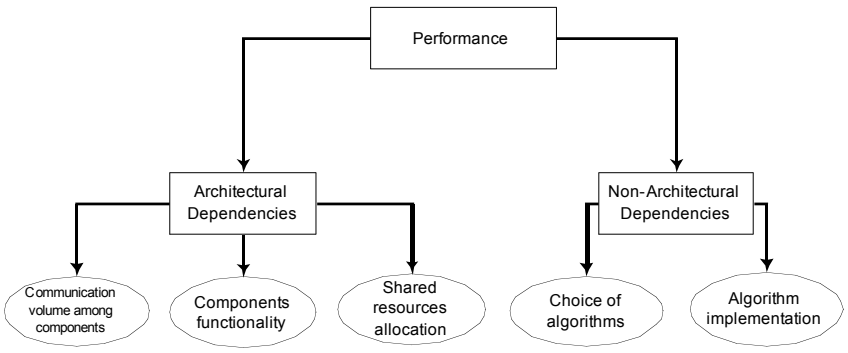
Software Architecture
The Process





Performance Requirements Patterns

Software Performance Quality Dependencies



Performance Requirements Patterns

Response Time Requirements Pattern	Throughput Requirements Pattern	Dynamic Capacity Requirements Pattern	Static Capacity Requirements Pattern
Common Performance Requirements Principles			

Common Performance Principles

- **Performance requirements are easy to write and hard to implement**
- **Software and hardware work together**
- **Identify where in the system the performance target belongs**
- **Avoid arbitrary performance requirements**
- **Define criticality of meeting performance requirement**
- **Measure the performance of the current system**
- **Define the timing of performance target.**
- **Define one performance target per requirement.**
- **Think options on how to meet a performance requirement**
- **Build a sizing model to conduct sensitivity analyses of performance**

(Withall, 2007)

Response Time
Requirement Pattern

- This requirement pattern is typically used when the user is interested in the time that takes an operation to be performed
- This requirement is stated as response time.
- Response time is the length of time between a request being submitted at a particular location of a system and a response being perceived at the same location.
- It is typically applied to user response time, which is the length of time between a user submitting the request and a response being displayed on the screen.

Response Time
Requirement Pattern

Requirement # 35 | ***Requirement Type*** Performance | ***Event /Use Case #*** 10

Description
- ***Operation Type*** . Any inquiry shall complete the display of its results

Fit Criteria
- ***(Tolerable length of time)*** In no longer than 4 seconds
- ***(Timing boundary start)*** From the time the user submits the request
- ***(Timing boundary finish)*** Until system displays results
- ***(Indicative hardware set -up)*** When using 56 k bits per second modem connection

Rationale
This figure is based on acceptance tests of previous version of system indicating that users begin to lose patience soon after this time .
- ***Exceptional Case (High load caveat).*** This requirement does not apply to inquiries across large volumes of data where arbitrary selection criteria are allowed
- ***Motivation*** . To ensure customers do not loose patience waiting for the system's response

Source
Historical acceptance tests of previous version of system

Customer Satisfaction 4 | ***Customer Dissatisfaction*** 4

Dependencies
None

Supporting Materials
Acceptance tests results of previous version of system

History
This requirement was first raised by Product Manager on 12/02/2008

Throughput
Requirement Pattern

- This requirement determines how fast the system can deliver its output. It is essential to have a sound basis for determining a target throughput. Consider the following steps to define a target throughput:
  - Decide what to measure in the system based on highest importance. Distinguish between incoming and outgoing throughput.
  - Identify other relative volumes of secondary activities if needed.
  - Specify required hardware set-up
  - Determine average throughput over a relatively long period of time.
  - Determine peak throughput the system must cope with the peak load.

Throughput
Requirement Pattern

Requirement # 78

Requirement Type Performance

Event/Use Case # 19

Description
- ***Component*** . The processor module of the system shall be able to process
- ***Throughput object type*** . The entry of orders by customers at a rate of at least

Fit Criteria
- ***Target Throughput quantity per unit time period*** . 10 orders per second
- ***Indicative hardware set up*** . No special consideration

Rationale
- ***Justification*** . Refer to the system sizing model for details on how the fit criterion was determined
- ***Contingency Statement*** . No contingency has been added
- ***Target Achievement Timeframe Statement*** . This rate represents the actual demand expected

Source

Motivation

Customer Satisfaction ***Customer Dissatisfaction***

Dependencies

Supporting Materials

History

Dynamic Capacity
Requirement Pattern

- This requirement pattern specifies the quantity of a particular type of entity for which the system must be able to perform processing at the same time.
- This pattern helps specifying the quantity of a particular type of entity for which the system must perform processing at the same time.
- It is intended primarily for the number of simultaneous users a system must be capable of handling. It also suggests what to do when too many users come along all at once.
- Specifying dynamic capacity is difficult unless there is an existing system for which figures can be obtained

Dynamic Capacity
Requirement Pattern

Requirement # 120 | ***Requirement Type*** Performance | ***Event / Use Case #*** 16

Description
- ***Operation Type*** . The system shall accommodate

Fit Criteria
- ***Entity Count*** . 200
- **Entity Type** . Customers
- ***Entity Condition*** . Logged in and active simultaneously when tickets for a popular concert go on sale (the definition of active customer is given in requirement 222 ; the definition of popular concert is given in requirement # 333)
- ***Duration of Peak*** . From half an hour before the published sale time
- ***Achievement Timeframe*** . Until two hours afterwards .
- ***Peak Period Concession*** . During a popular concert initial sale peak , it is acceptable for secondary services offered by the website (including any involving large downloads of the streaming of audio or video) to be shut down .
- ***Indicative Hardware Set*** ***-up.*** When using 56k bits per second modem connection

Rationale

Source

Motivation

Customer Satisfaction | ***Customer Dissatisfaction***

Dependencies

Supporting Materials

History

Static Capacity
Requirement Pattern

- This requirement specifies the quantity of a particular type of entity that the system must be able to store permanently (typically in a database).
- It should not be used to specify how long the data must be stored or how much disk space the system requires.
- The importance of static capacity is indirect in that all aspects of the system must be designed and built so as to be practical and work well when the target number of entities is present.
- Most systems have one type of entity that determines the quantity of most or all other high volume entities; customer is often the best type of entity to use.

Static Capacity
Requirement Pattern

Requirement # 231 ***Requirement Type*** Performance ***Event/Use Case #*** 06

Description
- ***Operation type*** . The system shall be able to process a minimum of

Fit Criteria
- ***Entity Count*** . 1,000,000
- ***Entity Type*** . Customers .
- ***Entity Inclusion Criteria*** . This figure covers only those customers who have accessed the Web site in the past three months or placed an order within the past twelve months .
- ***Achievement Timeframe Statement*** . It is not expected that this level of business will be reached earlier than two years after initial implementation .

Rationale
Previous version of the software have shown this be an acceptable number of customers

Source

Motivation

Customer Satisfaction ***Customer Dissatisfaction***

Dependencies

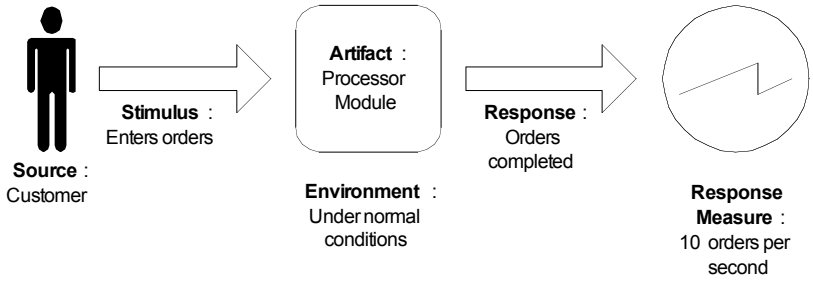
Supporting Materials

History

Architectural Scenario
Elements

- A quality attribute scenario is directly derived from a non-functional requirement and consists of the following parts:
  - ***source of stimulus***, which is the entity that generated the stimulus;
  - ***stimulus***, which is the condition that needs to be considered when it arrives at a system;
  - ***environment***, which determines under which conditions the stimulus occurs;
  - ***artifact***, is the element that receives the stimulus;
  - ***response***, which is the activity undertaken after the arrival of the stimulus;
  - ***response measure***, which represents the way a response is measured when it occurs.

Architectural Scenario
Throughput Requirement



Conclusions

- Defining functional requirements in software-intensive systems is always complex as required level of specificity is often “elusive”
- If defining functional requirements is difficult, defining non-functional requirements seem to be even more difficult because:
  - Clearly defining quality attributes of a system requires a lot of precision
  - Often testing non-functional requirements is not in the minds of people that define the non-functional requirements
- Architectural scenarios can be easily derived as all information needed is defined in the requirements patterns shown in this presentation
- Non-functional requirements patterns help to define test cases for the architectural scenarios

ABB

Summary
Contact Information

Author: Aldo Dagnino

ABB Corporate Research
Industrial Software Systems Program
940 Main Campus Drive
Raleigh, NC 27606 USA

Email: aldo.dagnino @ us.abb.com
Phone: (919) 807 - 5731

ABB

References

- [1] A. Bass, P. Clements, and R. Kazman, R., *Software Architecture in Practice*, SEI Series in Software Engineering, Addison Wesley, USA.
- [2] B. H. C. Cheng and J. M. Atlee, "Research Directions in Requirements Engineering", *Future of Software Engineering Conference*, IEEE, 2007.
- [3] I. Ozkaya, L. Bass, R. L. Nord, and R. Sangwan, "Making Practical Use of Quality Attribute Information", *IEEE Software*, March-April 2008, pp. 25-33.
- [4] S. Robertson and J. Robertswon, *Mastering the Requirements Process*, ACM Press, UK
- [5] K. E. Weigers, *Software Requirements*, Microsoft Press, United States, 2003.
- [6] S. Wthall, *Software Requirements Patterns*, Microsoft Press, Redmont, Washington, United States, 2007.

ABB

Power and productivity
for a better world™

ABB