Get Your Users “IN” the Doghouse and Keep Yourself Out: Eliciting Nonfunctional Requirements
Milwaukee SPIN 16 April 2009

The Quest for Excellent Requirements

Get Your Users “IN” the Doghouse and Keep Yourself Out: Eliciting Nonfunctional Requirements

Presented by: Roxanne E. Miller, CBAP
www.RequirementsQuest.com

Session Learning Objectives

Define nonfunctional requirements
Understand these vital, yet overlooked requirements
Classify common nonfunctional requirements
Get Your Users “IN” the Doghouse and Keep Yourself Out:
Eliciting Nonfunctional Requirements
Milwaukee SPIN        16 April 2009

Nonfunctional Definition

Learning Objective #1:
Define nonfunctional requirements

Levels of Requirements

Business Level
- Defines project scope.
- Identifies business benefits.
- Establishes high-level objectives.

User Level
- Focuses on business processes.
- Names user roles and user goals.
- Identifies flow of information and materials.

System Level
- Identifies processing of inputs and outputs.
- Defines functions and features.
- Describes quality attributes of the system environment.
Get Your Users “IN” the Doghouse and Keep Yourself Out:
Eliciting Nonfunctional Requirements
Milwaukee SPIN       16 April 2009

Nonfunctional in Context

High-level

Business Requirements

Sponsor point of view
Scope of the project
Business objectives

User Requirements

User point of view
User goals
User inputs and outputs

System Requirements

Functional—what the system does
Nonfunctional—how well the system does it

(Functional & Nonfunctional)

Detailed

Nonfunctional in Context

Oh No, Not Nonfunctional

Functional Requirements

Bucket 1

“A nonfunctional requirement is anything that is not functional.” NOT!

Nonfunctional Requirements

Bucket 2

Copyright©2009 Requirements Quest®. All rights reserved.
### A Handful of Definitions

**“Nonfunctional requirement”** – in software system engineering, a software requirement that describes not what the software will do, but how the software will do it, for example, software performance requirements, software external interface requirements, software design constraints, and software quality attributes. Nonfunctional requirements are difficult to test; therefore, they are usually evaluated subjectively. [Thayer2000]

**“Non-functional requirements”** – These are constraints on the services or functions offered by the system. They include timing constraints, constraints on the development process and standards. Non-functional requirements often apply to the system as a whole. They do not usually just apply to individual system features or services. [Sommerville2007]

**“Non-functional requirement”** – A property, or quality, that the product must have, such as an appearance, or a speed or accuracy property. [Robertson2003]

**“Nonfunctional requirement”** – A description of a property or characteristic that a software system must exhibit or a constraint that it must respect, other than an observable system behavior. [Wiegers2003]

**“Nonbehavioral requirements”** – Requirements that describe the required overall attributes of the system, including portability, reliability, efficiency, human engineering, testability, understandability, and modifiability. [Davis1993]

### What Does It Really Mean?

**Verbiage Removed:**

- General wordiness.
- Nonfunctional “examples.”
- “Descriptive” information that doesn’t add value.
- Inclusion or reference to “functional requirements.”

**...how the software will do it.**

- **Constraints** on the services or functions offered by the system.
- **Property**, or **quality**, that the product must have.
- **Property** that a software system must exhibit or a constraint that it must respect.
- **Required overall attributes** of the system.
A Simplified Definition

**Nonfunctional Requirement**

—a specification of *how well* a software *system* must function.

Functional versus Nonfunctional

**F** Product features.

**F** Describe the work that is done.

**F** Describe the actions with which the work is concerned.

**F** Characterized by verbs (active).

**N** Product properties.

**N** Describe the character of the work.

**N** Describe the experience of the user while doing the work.

**N** Characterized by adjectives.
Vital, Yet Why Overlooked?

Learning Objective #2:
Understand why nonfunctional requirements are vital, yet overlooked

Importance of Nonfunctional

All functional requirements may be satisfied, but if nonfunctional requirements are overlooked, the system will fail.
Nonfunctional Challenges

- No agreed upon formal **definition**.
- No complete **list**.
- No single universal **classification scheme**, accommodating all the needs of different application domains under different situations.
- Different people and organizations use **different terminologies**.

Alternative Names & Spelling

- **Quality Attributes**.
- **Quality Requirements**.
- **Nonbehavioral Requirements**.
- “ilities” or “ities.”
- Spelled with or without a hyphen. Non-functional or **Nonfunctional**.
Nonfunctional Complexity

**Subjective**
- Viewed, interpreted and evaluated differently by different people. Statements are often too brief and vague.

**Relative**
- Interpretation and importance may vary depending on the particular system and organizational needs.

**Integrated**
- Attempts to achieve one requirement can hurt or help the achievement of other requirements due to the nature of their global impact.

Nonfunctional Classification

**Learning Objective #3:**
Making sense of nonfunctional requirements with a user-focused classification
Get Your Users “IN” the Doghouse and Keep Yourself Out: Eliciting Nonfunctional Requirements
Milwaukee SPIN 16 April 2009

**Quality Attributes Tree**

- **General Utility**
  - As-Is Utility
  - Maintainability

- **Utility**
  - Portability
  - Reliability
  - Efficiency
  - Human Engineering
  - Testability
  - Understandability
  - Modifiability

- **Device-independence**
- **Self-containedness**
- **Accuracy**
- **Completeness**
- **Robustness / Integrity**
- **Consistency**
- **Accountability**
- **Device Efficiency**
- **Accessibility**
- **Communicativeness**
- **Self-descriptiveness**
- **Structuredness**
- **Conciseness**
- **Legibility**
- **Augmentability**

**Quality Attributes Tree**

- **Access Control**
- **Accountability**
- **Accuracy**
- **Adaptability**
- **Additivity**
- **Adjustability**
- **Agility**
- **Auditability**
- **Autonomy**
- **Availability**
- **Capability**
- **Capacity**
- **Clarity**
- **Code Space**
- **Commonality**
- **Compatibility**
- **Comprehensibility**
- **Conceptuality**
- **Conciseness**
- **Confidentiality**
- **Configurability**
- **Connectivity**
- **Consistency**
- **Controllability**
- **Data Space**
- **Decomposability**
- **Degradation of Service**
- **Dependability**
- **Development Cost**
- **Development Time**
- **Distributivity**
- **Diversity**
- **Ease of use**
- **Efficiency**
- **Elasticity**
- **Enhanceability**
- **Evolutivity**
- **Execution Cost**
- **Extensibility**
- **External Consistency**
- **Fault Tolerance**
- **Feasibility**
- **Flexibility**
- **Formality**
- **Generality**
- **Human Engineering**
- **Independence**
- **Informativeness**
- **Integrity**
- **Internal Consistency**
- **Interoperability**
- **Intuitiveness**
- **Learnability**
- **Leveragability**
- **Main Memory**
- **Maintainability**
- **Maintenance Cost**
- **Maintenance Time**
- **Maturity**
- **Mean Performance**
- **Measurability**
- **Mobility**
- **Modifiability**
- **Modularity**
- **Naturalness**
- **Nomadity**
- **Observability**
- **Off-Peak Period**
- **Operability**
- **Operating Cost**
- **Peak Period**
- **Performance**
- **Planning Cost**
- **Planning Time**
- **Portability**
- **Prediction**
- **Process Mgmt. Time**
- **Productivity**
- **Project Stability**
- **Project Tracking Cost**
- **Promptness**
- **Quality**
- **Reconfigurability**
- **Recoverability**
- **Recovery**
- **Reliability**
- **Repeatability**
- **Replaceability**
- **Replicability**
- **Response Time**
- **Responsiveness**
- **Retirement Cost**
- **Robustness**
- **Safeguards**
- **Safety**
- **Sensitivity**
- **Similarity**
- **Simplicity**
- **Space Boundness**
- **Space Performance**
- **Specificity**
- **Stability**
- **Subjectivity**
- **Supportability**
- **Surety**
- **Survivability**
- **Suscptibility**
- **Sustainability**
- **Tankness**
- **Testability**
- **Throughput**
- **Timeliness**
- **Traceability**
- **Trainability**
- **Transferability**
- **Transparency**
- **Understandability**
- **Uniformity**
- **Usability**
- **User-Friendlyness**
- **Validity**
- **Variability**
- **Verifiability**
- **Versatility**
- **Visibility**
- **Visibility**
- **Visibility**
### Nonfunctional Types

- **Nonfunctional**
  - **Organizational**
  - **Product**
  - **External**
    - **Delivery**
    - **Implementation**
    - **Standards**
      - **Usability**
      - **Efficiency**
      - **Reliability**
      - **Portability**
    - **Performance**
    - **Space**
    - **Legislative**
    - **Ethical**
    - **Interoperability**

[Ian Sommerville 2007]

### What are the User’s Needs?

- **Operation** How well does the system operate for daily use by the user?
- **Revision** How easy is it to correct errors and add functions?
- **Transition** How easy is it to adapt to changes in the technical environment?
## A User-Focused Classification

<table>
<thead>
<tr>
<th>User Concern</th>
<th>Nonfunctional</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Operation</strong></td>
<td></td>
</tr>
<tr>
<td>How well is it safeguarded against unauthorized access?</td>
<td>Access Security</td>
</tr>
<tr>
<td>How dependable is it during normal operating times?</td>
<td>Availability</td>
</tr>
<tr>
<td>How fast, how many, and how well does it respond?</td>
<td>Efficiency</td>
</tr>
<tr>
<td>How accurate and authentic is the data?</td>
<td>Integrity</td>
</tr>
<tr>
<td>How immune is the system to failure?</td>
<td>Reliability</td>
</tr>
<tr>
<td>How resilient is the system from failure?</td>
<td>Survivability</td>
</tr>
<tr>
<td>How easy is it to learn and operate the system?</td>
<td>Usability</td>
</tr>
<tr>
<td><strong>Revision</strong></td>
<td></td>
</tr>
<tr>
<td>How easy is it to modify to work in different environments?</td>
<td>Flexibility</td>
</tr>
<tr>
<td>How easy is it to upkeep and repair?</td>
<td>Maintainability</td>
</tr>
<tr>
<td>How easy is it to expand or upgrade its capabilities?</td>
<td>Scalability</td>
</tr>
<tr>
<td>How easy is it to show it performs its functions?</td>
<td>Verifiability</td>
</tr>
<tr>
<td><strong>Transition</strong></td>
<td></td>
</tr>
<tr>
<td>How easy is it to interface with another system?</td>
<td>Interoperability</td>
</tr>
<tr>
<td>How easy is it to transport?</td>
<td>Portability</td>
</tr>
<tr>
<td>How easy is it to convert for use in another system?</td>
<td>Reusability</td>
</tr>
</tbody>
</table>

### Operation: Access Security

- **Definition** — the extent to which the system is safeguarded against deliberate and intrusive faults from internal and external sources.

- **User concern**
  - How well is the system safeguarded against unauthorized access?

- **Example:**

  Employees shall be forced to change their password the next time they log in if they have not changed it within the length of time established as “password expiration duration.”
Operation: Availability

**Definition** — the degree to which users can depend on the system to be up (able to function) during “normal operating times.”

**User concern**

- How dependable is the system during normal operating times?

**Example:**

The Automated Teller Machine shall be at least 99.5 percent available on weekdays between 6:00 a.m. and midnight local time. The machine shall be at least 99.95 percent available on weekdays between 3:00 p.m. and 7:00 p.m. local time.

Operation: Efficiency

**Definition** — the extent to which the software system handles capacity, throughput, and response time.

**User concern**

- How fast does it process (capacity)?
- How many at a time (throughput)?
- Response time?

**Example:**

The system must download the new rate parameters within ten minutes of a non-scheduled rate change.
Operation: Integrity

**Definition** — the degree to which the data maintained by the software system is accurate, authentic, and without corruption.

**User concern**

-how accurate and authentic is the data?

**Example:**

The integrity of the system data area must be checked by the internal audit system twice per second; if inconsistencies in the data are detected, the system operation should be disabled.

---

Operation: Reliability

**Definition** — the extent to which the software system consistently performs the specified functions without failure.

**User concern**

-how immune is the system to failure?

**Example:**

No more than 10 claim assignments out of 5000 can be “unassigned” because of system failures.
### Operation: Survivability

**Definition** — the extent to which the software system continues to function and recovers in the presence of a system failure.

**User concern**

- How resilient is the system from failure?

**Example:**

All policy statement parameters shall have default values specified, which the Report Writer system shall use if a parameter’s input data is missing or invalid.

### Operation: Usability

**Definition** — the ease in which the user is able to learn, operate, prepare inputs and interpret outputs through interaction with a system.

**User concern**

- How easy is it to learn and operate the system?

**Example:**

A trained order-entry clerk shall be able to submit a complete order for a product chosen from a supplier catalog in a maximum of 7 minutes, and an average order entry time of 4 minutes.
Revision: Flexibility

**Definition** — the ease in which the software can be modified to adapt to different environments.

**User concern**

How easy is it to change and add new features?

**Example:**

It shall be possible to add a new delivery option for customer mailing method by developing and “plugging in” the functionality necessary to support that delivery option.

Revision: Maintainability

**Definition** — the ease in finding and fixing faults in the software system.

**User concern**

How easy is it to upkeep and repair the system?

**Example:**

The application development process must have a regression test procedure that allows complete re-testing within 2 business days.
Revision: Scalability

Definition — the degree in which the software system is able to expand its processing capabilities upward and outward to support business growth.

User concern

How easy is it to expand or upgrade the capabilities of the system?

Example:

Any increase in the number of customers shall not degrade system availability to an extent noticeable by any users.

Revision: Verifiability

Definition — the extent to which tests, analysis, and demonstrations are needed to prove that the software system will function as intended.

User concern

How easy is it to show it performs its functions?

Example:

The maximum number of test cases to cover testing of any particular source code module shall be 20.
**Transition: Interoperability**

**Definition** — the extent to which the software system is able to couple or facilitate the interface with other systems.

**User concern**

- How easy is it to interface with another system?

**Example:**

The system must be able to interface with any HTML (HyperText Markup Language) browser.

**Transition: Portability**

**Definition** — the ease in which a software system from its current hardware or software environment can be transferred to another environment.

**User concern**

- How easy is it to transport?

**Example:**

The system is designed to run in business offices, but the intent is to have a version which will run in manufacturing assembly plants.
**Transition: Reusability**

**Definition** — the extent to which a portion of the software system can be converted for use in another.

**User concern**

用户提供：How easy is it to convert for use in another system?

**Example:**

The payment subsystem design is based on the payment module from the ALPHA product line. The ePAYZ system should not be modified unless absolutely necessary.

---

**Session Learning Objectives**

- Define nonfunctional requirements
- Understand these vital, yet overlooked requirements
- Classify common nonfunctional requirements
A Requirement “Pattern”

- Basic details
- Applicability
- Discussion
- Content
- Template(s)
- Example(s)
- Extra requirements
- Development considerations
- Testing considerations


Over 2,000 Suggested Questions

- By Focus
  - Data (what?)
  - Roles (who?)
  - Purpose (why?)
  - Timing (when?)
  - Logistics (where?)
  - Process (how?)

- By Perspective
  - Requirements Supplier
  - Requirements Receiver

Get Your Users “IN” the Doghouse and Keep Yourself Out:
Eliciting Nonfunctional Requirements
Milwaukee SPIN 16 April 2009

Questions

Presented by: Roxanne E. Miller, CBAP
www.RequirementsQuest.com

Copyright©2009 Requirements Quest®. All rights reserved.