Requirements:
how to write and read them

Valentina Presutti
Courtesy of Paolo Ciancarini
Agenda

- What are the requirements of a software product?
- Requirement engineering
- Classifying requirements
- Representing requirements
A Software Requirement is … (IEEE 610)

1. A condition or capability needed by a user to solve a problem or achieve an objective
2. A condition or capability that must be met or possessed by a system or component to satisfy a contract, standard, specification, or other formally imposed documents
3. A documented representation of a condition or capability as in (1) or (2)
Requirement (RUP)

A requirement describes a condition or capability to which a system must conform; either derived directly from user needs, or stated in a contract, standard, specification, or other formally imposed document.
Writing requirements

• To define **what** the customer requires to a software product
• (not to define **how** the product will be built)
Requirements engineering

• A process to define:
  – **Services** required by the users of a systems
  – **User Scenarios**
  – **Development constraints**
  – **Operational constraints**
Requirements engineering

- Elicitation
- Specification
- Management
- Analysis
- Verification
Techniques for requirements *elicitation*

- Interviews and forms
- Requirements workshop
- Brainstorming
- Storyboards
- Use cases
- Role playing games
- Prototyping
Main activities during the elicitation

- Identify the actors
- Identify the use cases
- Identify the scenarios
- Identify relationships among the use cases
- Refine the use cases
- Identify the non functional requirements
- Identify the participating entities
What is a requirement?

• An abstract and vague *description* of a system service or a constraint, or a detailed *specification* of a system function

• This because requirements have a double face:
  – They are the basis for *bidding* for a contract, so they must be open to different interpretations
  – They are the basis for *granting* a contract: then they must include sufficient details to exclude unwanted interpretations
The IEEE 830-1993 standard

- The Software Requirement Specification document (SRS) must correctly define all of the software requirements, but no more

- Eight qualities:
  1. Unambiguous
  2. Correct
  3. Complete
  4. Verifiable
  5. Consistent
  6. Modifiable
  7. Traceable
  8. Ranked (for importance and/or stability)
Goals of a SRS document

• Give feedback to the user
• Decompose a problem in independent sub-problems
• Input for the design phase
• Input for the testing, verification, and validation phases
Types of requirements

Properties of a system desired by the stakeholders

- Behavioral vs quality properties
  - Functional/non functional requirements
- Environment
  - System / user requirements
- Conflicts among requirements?
  - Ranking / priority
- Conflicts among stakeholders?
  - Stakeholders / views
  - Trade-off / negotiation
Categories of requirements

• **Functional requirements**: describe the behavior and interactions between the system and its environment
  – Eg.: the e-learning platform tracks student’s activities
  – Eg.: the platform is used for both reading texts and participating to exams

• **Non functional requirements**: describe the system properties visible to the users and not directly related to its functional behavior
  – Eg.: the platform has to manage up to 100 students concurrently
  – Eg.: If a user posts a question he must have an answer in max 48h
  – Eg.: The platform has to be available 24h per day 7 days per week (24/7)

• **Constraints** (“Pseudo requirements”): defined by the stakeholders
  – Eg.: The programming language will be Java
  – Eg.: The platform must deal with documents written in Office Word on Vista
What is NOT a requirement?

• System structure (or “architecture”)
• Implementation technologies
• Development method
• Development environment
• Programming language
• Reuse and portability concerns

• Instead the real world domain descriptions are included in the requirement document
Useful questions when writing a requirement document

• What is the goal of the system?
• What is the goal of its users?
• How it will interface with users, hardware, and other software systems?,
• Which performance do we expect from the system?
• Which constraints should be satisfied?
Describing and specifying requirements

• **Definition of requirements**
  – Description in natural language of a system, including its services and operational constraints
  – Written for customers and stakeholders

• **Specification of requirements**
  – Structured document detailing all services the system should offer
  – Written as a contract between stakeholders and developers
  – Written for contract bidders and developers
Readers of requirements

- Definition
  - Client Managers
  - System End-users
  - Client Engineers
  - Contractor Managers
  - System Architects

- Specification
  - System End-users
  - Client Engineers
  - System Architects
  - Software Developers
Conflicts and priorities

• Different stakeholders have different priorities
• Final users have expectations different from funding people
• Difficult to anticipate all problems and conflicts
• Prototyping is a useful technique to mitigate conflicts and anticipate problems
Problems with Natural Language

• Specification writers and readers should use the same NL wording with the same meaning, but this is rarely the case
• A specification written in NL has always several alternative interpretations
• The requirements should be partitioned and ranked but NL is not suitable for this
Alternatives to NL

• Structured Natural Language
• Visual notations, eg. UML
• Formal languages, eg. Z
The requirement process
The requirements document

• The requirement document defines “what” is required from the developers
• It is useful for both defining and analyzing the requirements
• *It is not a design document*
• It says *what*, not *how* the system will do
Requirements of a requirement document

- It must specify the functional behavior seen from outside the system
- Must specify implementation constraints
- Must be easily modifiable
- It is a reference document for maintainers
- Must anticipate some likely modifications of the system
- Must define how the system reacts to unexpected events
The requirements document: structure

- Introduction (requirements definition)
  - Why the system is being built
- Functional requirements
  - Detailed description of the services offered by the system
- Non-functional requirements
  - Constraints on product and process
- Evolution
  - Critical assumptions and likely future changes
- Glossary
  - Definition of technical terms
- Table of contents and index
The first page

- Project or product name
- Date
- Version number
- Authors
- Responsibilities of each author
- Main changes since last version
Definition of requirements

- Specify the external behavior
- The requirements should not indicate a computing model or a software architecture
Classifying the requirements

- Functional requirements describe the services a system offers or the effects of its usage (eg. “This procedure prints a document”)
- Non-functional requirements describe some system qualities (eg. “This procedure prints a document three pages per second”) or some constraints on its development
- Some requirements can be both functional and non-functional (eg. security requirements, like “The ability to securely transmit data to remote sites”)
Types of Requirements

The User & System

Functional

- Business Requirements
  - Scope and Vision Document
  - User Requirements
  - Use Case Document
  - System Requirements

Nonfunctional

- Business Rules
- Quality Attributes
- External Interfaces
- Constraints

The Environment

Requirements Specification
Functional requirements: examples

• A user can borrow a book
• The user can search in all library catalogues or in a subset of them
• The system has to offer all browsers necessary to display any stored document
• Each document has a unique ID that a user can exploit to check-out the document itself
Writing requirements

• Most requirement documents use the natural language (NL)

• Problems when using a NL for requirements:
  – **Clarity**. Precision is inversely proportional to readability
  – **Confusion**. Functional and non-functional requirements are normally mixed
  – **Ambiguity**. NL is easily misinterpreted
Using NL: the library example

A library is managed by some staff and has some users who can search and borrow books. A user can keep a book for one month, then has to give it back. The library contains one million books, and the staff buy about 10,000 books each year. A new book cannot be borrowed before six weeks. If a book is not available, the request is put in a waiting list. When a book is lost it is deleted from the catalog and its waiting list is canceled...
Exercise

• How many and which kinds of requirements can you find in the library example?
The library example

- “The system shall maintain records of all library materials including books, serials, newspapers and magazines, video and audio tapes, reports, collections of transparencies, computer disks and CD-ROMS.”
- “The system shall allow the users to search for an item by title, author, or ISBN.”
- “The system’s user interface shall be displayed using Firefox 2 and above”
- “The system shall support at least 20 transactions per second.”
- “There should be no more than 3 clicks from homepage to reach search results.”
- “The access permissions for system data may only be changed by the system’s data administrator”
Different types of requirements

1. “The system shall maintain records of all library materials…”  
   [System requirement]
2. “The system shall allow the users to search for an item by title, author, or ISBN.”  
   [Functional requirement]
3. “The system’s user interface shall be displayed using Firefox 2 and above”  
   [Implementation requirement]
4. “The system shall support at least 20 transactions per second.”  
   [Performance requirement to specify acceptable QoS]
5. “There should be no more than 3 clicks…”  
   [Usability requirement]
6. “The access permissions for system data may…”  
   [Security requirement]
Using structured NL

• The NL can be restricted in order to increase clarity and precision
• A structured NL can avoid some typical ambiguities by constraining the requirement documents to have some structure
• Usually a structured NL specification is based on a pre-defined form
Form for functional requirements

• Define the function or entity
• Describe inputs and their sources
• Describe output and their destinations
• Other entities possibly involved
• Pre and post conditions
### Example

<table>
<thead>
<tr>
<th>Function</th>
<th>Add node</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Adds a node to an existing design. The user selects the type of node, and its position. When added to the design, the node becomes the current selection. The user chooses the node position by moving the cursor to the area where the node is added.</td>
</tr>
<tr>
<td>Inputs</td>
<td>Node type, Node position, Design identifier.</td>
</tr>
<tr>
<td>Source</td>
<td>Node type and Node position are input by the user, Design identifier from the database.</td>
</tr>
<tr>
<td>Outputs</td>
<td>Design identifier.</td>
</tr>
<tr>
<td>Destination</td>
<td>The design database. The design is committed to the database on completion of the operation.</td>
</tr>
<tr>
<td>Requires</td>
<td>Design graph rooted at input design identifier.</td>
</tr>
<tr>
<td>Pre-condition</td>
<td>The design is open and displayed on the user's screen.</td>
</tr>
<tr>
<td>Post-condition</td>
<td>The design is unchanged apart from the addition of a node of the specified type at the given position.</td>
</tr>
<tr>
<td>Side-effects</td>
<td>None</td>
</tr>
</tbody>
</table>

*Definition: ECLIPSE/Workstation/Tools/DE/RD/3.5.1*
Rationale

• It is important to explain the reason (rationale) for each requirement
• The rationale helps the developer to understand the applicative domain and put in context the requirement
• The rationale is critical when a requirement has to be modified: its presence reduces the possibility that a change in the requirements document has unexpected effects
Non-functional requirements

- Non-functional requirements are criteria usable to judge the operation of a system, rather than its specific behaviors.
- They define the critical properties of a system. Examples: reliability, throughput, answer time, security, user-friendliness.
- Some non-functional requirements put constraints on the system development process or infrastructure.
- Non-functional requirements are often more important than functional ones, because if they are not satisfied the system will be refused.
Non-functional requirements: examples

• Product requirement (standard)
  – For each communication message between the users and the system the standard ASCII will be used

• Process requirement (standard)
  – The development process and related documentation will conform to the standard ISO-XYWZ-2007

• External requirement (privacy)
  – The system will not allow any operator to know any personal information on customers, except their name and reference ID
Classification of non-functional reqs
Requirements validation

• Prove that the requirements define the system that the customer really wants
• Requirements error costs are high so validation is very important
  – Fixing a requirements error after delivery may cost up to 100 times the cost of fixing an implementation error
Requirements checking

- **Validity.** Does the system provide the functions which best support the customer’s needs?
- **Consistency.** Are there any requirements conflicts?
- **Completeness.** Are all functions required by the customer included?
- **Realism.** Can the requirements be implemented given available budget and technology?
- **Verifiability.** Can the requirements be checked?
Reqs validation techniques

• Requirements reviews
  – Human analysis of the requirements, performed in specific meetings

• Prototyping
  – Using an executable model (prototype or animation) of the system to check requirements

• Test-case generation
  – Developing tests for requirements to check testability
Requirements reviews

- Regular reviews should be held while the requirements definition is being written
- Both client and contractor staff should be involved in reviews
- Reviews may be formal (with completed documents) or informal
Review checks

• **Verifiability.** Is the requirement realistically testable?
• **Comprehensibility.** Is the requirement properly understood?
• **Traceability.** Is the origin of the requirement clearly stated?
• **Adaptability.** Can the requirement be changed without a large impact on other requirements?
Discuss

Invent some requirements which are

• Not testable, or
• Not understandable, or
• Not traceable, or
• Not modifiable
Requirements management

• Requirements management is the process of managing changing requirements during the requirements engineering process and system development

• Requirements are inevitably incomplete and inconsistent
  – New requirements emerge during the process as business needs change and a better understanding of the system is developed;
  – Different viewpoints have different requirements and these are often contradictory
Requirements change

• The priority of requirements from different viewpoints changes during the development process
• System customers may specify requirements from a business perspective that conflict with end-user requirements
• The business and technical environment of the system changes during its development
Requirements evolution
Enduring and volatile reqs

• **Enduring requirements** derived from the core activity of the customer organisation. May be derived from domain models
  – E.g. a hospital always has doctors, nurses, etc.

• **Volatile requirements** change during development or when the system is in use
  – Eg., in a hospital, requirements derived from health-care policy can periodically change
## Requirements classification

<table>
<thead>
<tr>
<th>Requirement Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mutable requirements</td>
<td>Requirements that change because of changes to the environment in which the organisation is operating. For example, in hospital systems, the funding of patient care may change and thus require different treatment information to be collected.</td>
</tr>
<tr>
<td>Emergent requirements</td>
<td>Requirements that emerge as the customer's understanding of the system develops during the system development. The design process may reveal new emergent requirements.</td>
</tr>
<tr>
<td>Consequential</td>
<td>Requirements that result from the introduction of the computer system. Introducing the computer system may change the organisations processes and open up new ways of working which generate new system requirements</td>
</tr>
<tr>
<td>Compatibility</td>
<td>Requirements that depend on the particular systems or business processes within an organisation. As these change, the compatibility requirements on the commissioned or delivered system may also have to evolve.</td>
</tr>
</tbody>
</table>
Requirements management planning

• During the requirements engineering process, you have to plan:
  – **Requirements identification**
    • How requirements are individually identified;
  – **A change management process**
    • The process followed when analysing a requirements change;
  – **Traceability policies**
    • The amount of information about requirements relationships that is maintained;
  – **CASE tool support**
    • The tool support required to help manage requirements change;
Traceability

• Traceability is concerned with the relationships between requirements, their sources and the system design

• **Source traceability**
  – Links from requirements to stakeholders who proposed these requirements;

• **Requirements traceability**
  – Links between dependent requirements;

• **Design traceability**
  – Links from the requirements to the design;
Tracking techniques

- Assign a unique id to each requirement
- Cross reference of requirements in a traceability matrix
- Hypertext management of requirements in order to support effective navigation mechanisms
A traceability matrix

<table>
<thead>
<tr>
<th>Req. id</th>
<th>1.1</th>
<th>1.2</th>
<th>1.3</th>
<th>2.1</th>
<th>2.2</th>
<th>2.3</th>
<th>3.1</th>
<th>3.2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>D</td>
<td>R</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.2</td>
<td></td>
<td></td>
<td></td>
<td>D</td>
<td>D</td>
<td>D</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.3</td>
<td></td>
<td></td>
<td>R</td>
<td></td>
<td></td>
<td>R</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.1</td>
<td></td>
<td></td>
<td></td>
<td>R</td>
<td>D</td>
<td>D</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>D</td>
</tr>
<tr>
<td>2.3</td>
<td></td>
<td></td>
<td></td>
<td>R</td>
<td>D</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>R</td>
</tr>
<tr>
<td>3.2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>R</td>
</tr>
</tbody>
</table>
Tools for managing requirements

• Example: DOORS, EclipseRS plugin

• Pro
  – Requirements tracking during the lifecycle
  – Tracking with respect to the software architecture
  – Managing the requirements documents

• Vs
  – Poor mechanisms supporting analysis and decomposition
  – Requirements knowledge distributed in several tools
  – Still not mature, pragmatically inefficient
Example: DOORS
DOORS

• DOORS is a requirements tracking tool
• It helps in building and managing a requirements database
• It has been developed by Telelogic, a Swedish company acquired by IBM in 2008
**User Regts**

3 Requirements

- **3.1 Capability Requirements**
  - **3.1.1 Carrying Capacity**
    - **3.1.1.1 Number of People**
      - Four average size adults shall be able to travel in comfort for a period of 3 hours. This level of comfort is defined as being equivalent to the standard of comfort provided by the top 40% of cars produced in 1999.

```
The top level of cars are those in the price range $20,000 to $40,000 at 1999 prices.
```

- Five average size adults shall be able to travel in comfort for a period of 3 hours.
- Users shall have easy entry and exit.

**Technical Regts**

- SR-104 2.14.1.0.1 from /Sports utility vehicle 4x2/Requirements/Functional Requirements
  - The car shall be able to carry 4 average size adults in average comfort for a period of 3 hours.
  - Last modified 11 February 1997

- SR-114 2.14.5.0.1 from /Sports utility vehicle 4x2/Requirements/Functional Requirements
  - The car shall be able to...

**Design**

- D-342
  - Full seats shall be created for two passengers in both front and back.

- D-344
  - There shall be space for a fifth passenger in the back that will not meet the comfort requirement.

**Test Cases**

- Test Number 18
  - Market Research
    - Test Result: Passed

- Test Number 12
  - Verify Number of People
    - Test Result: Untested

- Test Number 6
  - Verify support for Customers
    - Test Result: Untested
CASE tool support

- **Requirements storage**
  - Requirements should be managed in a secure repository

- **Change management**
  - Change management is a workflow whose stages can be defined and the information flows between these stages partially automated

- **Traceability management**
  - Automated retrieval of the links between requirements.
Reqs change management

• Should apply to all proposed changes to the requirements.

• Principal stages
  – Problem analysis. Discuss requirements problem and propose change;
  – Change analysis and costing. Assess effects of change on other requirements;
  – Change implementation. Modify requirements document and other documents to reflect change.
Summary

• There are several types of requirements
• Requirement engineering is the discipline which studies the methods for managing requirements
• The phase of requirements elicitation is crucial for writing a good requirement document
Questions

• What is a requirement?
• Which languages are used to write the requirements?
• What is a non-functional requirement?
• Which are the main techniques for the phase of requirements elicitation?
• What is a traceability matrix?
References

• SWEBOK. Chapter on “Requirements Engineering”
• 830-1998 Recommended Practice for Software Requirements Specifications (ANSI/IEEE)
• Aurum and Wohlin, Engineering and Managing Software Requirements, Springer, 2005
• D. Leffingwell and D. Widrig, Managing Software Requirements: A Unified Approach, Addison-Wesley, 2000
• K. Wiegers, Automating Requirements Management, in Software Development, 1999
Tools for reqs management

- www.volere.co.uk/tools.htm
- sourceforge.net/projects/eclipsesrs
- www.telelogic.com/corp/products/doors/index.cfm
- www.analysttool.com
- www.processimpact.com/articles/rm_tools.html
- www.limov.com
Questions?