Requirements Engineering

- Establishing **what** the customer requires from a software system

Requirements Engineering

- The process of establishing the:
  - services that the customer requires from a system
  - constraints under which the system operates
  - constraints under which the system is developed.
- Requirements may be **functional** or **non-functional**
  - Functional requirements describe system services or features.
  - Non-functional requirements is a constraint on the system or on the development process.

What is a Requirement?

- It may range from a high-level abstract statement of a service (or of a system constraint) to a detailed mathematical functional specification.
- This is inevitable as requirements may serve a dual function:
  - May be the basis for a bid for a contract - therefore must be open to interpretation.
  - May be the basis for the contract itself - therefore must be defined in detail.
  - Both these statements may be called requirements.
Requirements Definition/Specification

- **Requirements Definition**
  - A statement in natural language of the services the system provides and its operational constraints.
  - Written for customers.

- **Requirements Specification**
  - A structured document setting out detailed descriptions of the system services.
  - Written as a contract between client and contractor.
  - Written for contractors and developers.

Requirements Readers

- **Requirements definition**
  - Client managers
  - System end-users
  - Client engineers
  - Contractor managers
  - System architects

- **Requirements specification**
  - System end-users
  - Client engineers
  - System architects
  - Software developers

Reasons for Inconsistency

- Large software systems must improve the current situation. It is hard to anticipate the effects that the new system will have on the organization.
- Different users have different requirements and priorities.
- System end-users and organizations who pay for the system have different requirements.
- Prototyping is often required to clarify requirements.
Problems with Natural Language

- Natural language relies on the specification readers and writers using the same words for the same concept.
- A natural language specification is over-flexible and subject to different interpretations.
- Requirements are not partitioned by language structures.

Natural Language Alternatives

- Structured natural language
- Graphical notations
- Mathematical specifications

The RE process
The Requirements Document

- The requirements document is the official statement of what is required of the system developers.
- Should include both a definition and a specification of requirements.
- It is NOT a design document. As far as possible, it should set out WHAT the system should do rather than HOW it should do it.

Requirements Document Requirements

- Specify external system behavior.
- Specify implementation constraints.
- Easy to change.
- Serve as reference tool for maintenance.
- Record forethought about the life cycle of the system i.e., predict changes.
- Characterize responses to unexpected events.

Requirements Document Structure

- Introduction (Requirements Definition)
  - Describe need for the system and how it fits with business objectives.
- Functional Requirements
  - Describe the services to be provided in detail.
- Non-functional Requirements
  - Define constraints on the system and the development process.
- System Evolution
  - Define fundamental assumptions on which the system is based and anticipated changes.
- Glossary
  - Define technical terms used.
- Index
Requirements Definition

- Should specify external behavior of the system.
- The requirements should not be defined using a computational model.

Writing Requirements

- Natural language is typically used when writing requirements definitions.
- This is universally understandable but three types of problem can arise:
  - Lack of clarity. Precision is difficult without making the document difficult to read
  - Requirements confusion. Functional and non-functional requirements tend to be mixed-up
  - Requirements amalgamation. Several different requirements may be expressed together

Definition and Specification

- Requirements definition
  - Customer-oriented descriptions of the system’s functions and constraints on its operation.
- Requirements specification
  - Precise and detailed descriptions of the system’s functionality and constraints.
  - Intended to communicate what is required to system developers and serve as the basis of a contract for the system development.
Functional Requirements using Structured Language

- A limited form of natural language may be used to express requirements.
- This removes some of the problems resulting from ambiguity and flexibility and imposes a degree of uniformity on a specification.
- Often best supported using a forms-based approach.

Examples of Functional Requirements

- The user shall be able to search either all of the initial set of databases or select a subset from it.
- The system shall provide appropriate viewers for the user to read documents in the document store.
- Every order shall be allocated a unique identifier (ORDER_ID) which the user shall be able to copy to the account’s permanent storage area.

Form-based Functional Specifications

- Definition of the function or entity.
- Description of inputs and where they come from.
- Description of outputs and where they go to.
- Indication of other entities required.
- Pre and post conditions (if appropriate).
Example of a Form-based Functional Specification

<table>
<thead>
<tr>
<th>Function</th>
<th>Add node</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Add 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>
</tbody>
</table>

Non-functional Requirements

- Define system properties and constraints e.g., reliability, response time and storage requirements. Constraints are I/O device capability, system representations, etc.
- Process requirements may also be specified mandating a particular CASE system, programming language or development method.
- Non-functional requirements may be more critical than functional requirements. If these are not met, the system is useless.

Requirements Rationale

- It is important to provide rationale with requirements.
- This helps the developer understand the application domain and why the requirement is stated in its current form.
- Particularly important when requirements have to be changed. The availability of rationale reduces the chances that change will have unexpected effects.
Non-functional Requirement Types

- Performance requirements
- Space requirements
- Usability requirements
- Efficiency requirements
- Reliability requirements
- Portability requirements
- Interoperability requirements
- Ethical requirements
- Legislative requirements
- Implementation requirements
- Standards requirements
- Delivery requirements
- Safety requirements
- Privacy requirements
- Product requirements
- Organizational requirements
- External requirements

Non-functional requirements examples

- **Product requirement**
  - 4.4.8 It shall be possible for all necessary communication between the APSE and the user to be expressed in the standard Ada character set

- **Organisational requirement**
  - 9.3.2 The system development process and deliverable documents shall conform to the process and deliverables defined in XYZCo-SP-STAN-95

- **External requirement**
  - 7.6.5 The system shall not disclose any personal information about customers apart from their name and reference number to the operators of the system

User requirements

- Should describe functional and non-functional requirements so that they are understandable by system users who don’t have detailed technical knowledge
- User requirements are defined using natural language, tables and diagrams
System-level Requirements

- Some requirements place constraints on the system as a whole rather than specific system functions.
- Example
  - The time required for training a system operator to be proficient in the use of the system must not exceed 2 working days.
- These may be emergent requirements which cannot be derived from any single sub-set of the system requirements.

Editor Grid Requirement

2.6 Grid facilities To assist in the positioning of entities on a diagram, the user may turn on a grid in either centimeters or inches, via an option on the control panel. Initially, the grid is off. The grid may be turned on and off at any time during an editing session and can be toggled between inches and centimeters at any time. A grid option will be provided on the reduce-to-fit view but the number of grid lines shown will be reduced to avoid filling the smaller diagram with grid lines.

Defining Requirements

- Editor requirement mixes up functional and non-functional requirements and is incomplete.
- Easy to criticize but hard to write good requirements definitions.
- Use of a standard format with pre-defined fields to be filled means that information is less likely to be missed out.
Editor Grid Requirement

2.6 Grid facilities

2.6.1 The editor shall provide a grid facility where a matrix of horizontal and vertical lines provide a background to the editor window. This grid shall be a passive grid where the alignment of entities is the user’s responsibility. Rationale: A grid helps the user to create a tidy diagram with well-spaced entities. Although an active grid, where entities ‘snap-to’ grid lines can be useful, the positioning is imprecise. The user is the best person to decide where entities should be positioned.

2.6.2 When used in ‘reduce-to-fit’ mode (see 2.1), the number of units separating grid lines must be increased. Rationale: If line spacing is not increased, the background will be very cluttered with grid lines.

Specification: ECLIPSE/WS/Tools/DE/FS Section 5.6

Node Creation Requirement

3.5.1 Adding nodes to a design

3.5.1.1 The editor shall provide a facility where users can add nodes of a specified type to a design. Nodes are selected (see 3.4) when they are added to the design.

3.5.1.2 The sequence of actions to add a node should be as follows:
1. The user should select the type of node to be added.
2. The user moves the cursor to the approximate node position in the diagram and indicates that the node symbol should be added at that point.
3. The symbol may then be dragged to its final position.

Rationale: The user is the best person to decide where to position a node on the diagram. This approach gives the user direct control over node type selection and positioning.

Specification: ECLIPSE/WS/Tools/DE/FS. Section 3.5.1

Requirements Traceability

- Requirements traceability means that related requirements are linked in some way and that requirements are (perhaps) linked to their source.
- Traceability is a property of a requirements specification which reflects the ease of finding related requirements.
Traceability Techniques

- Assign a unique number to all requirements.
- Cross-reference related requirements using this unique number.
- Use HTML hyperlinks to implement traceability.

Requirements Verifiability

- Requirements should be written so that they can be verified objectively.
- The problem with this requirement is its use of vague terms such as “errors shall be minimized”
  - The system should be easy to use by experienced controllers and should be organized in such a way that user errors are minimized.
- The error rate should be quantified.
  - Experienced controllers should be able to use all the system functions after a total of two hours training. After this training, the average number of errors made by experienced users should not exceed two per day.

Requirements Measures

<table>
<thead>
<tr>
<th>Property</th>
<th>Measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Speed</td>
<td>Processed transactions/second</td>
</tr>
<tr>
<td></td>
<td>Screen refresh time</td>
</tr>
<tr>
<td>Size</td>
<td>K Bytes</td>
</tr>
<tr>
<td></td>
<td>Number of RAM chips</td>
</tr>
<tr>
<td>Ease of use</td>
<td>Training time</td>
</tr>
<tr>
<td></td>
<td>Number of help frames</td>
</tr>
<tr>
<td>Reliability</td>
<td>Mean time to failure</td>
</tr>
<tr>
<td></td>
<td>Probability of unavailability</td>
</tr>
<tr>
<td></td>
<td>Rate of failure occurrence</td>
</tr>
<tr>
<td></td>
<td>Availability</td>
</tr>
<tr>
<td>Robustness</td>
<td>Time to restart after failure</td>
</tr>
<tr>
<td></td>
<td>Percentage of events causing failure</td>
</tr>
<tr>
<td></td>
<td>Probability of data corruption on failure</td>
</tr>
<tr>
<td>Portability</td>
<td>Percentage of target dependent statements</td>
</tr>
<tr>
<td></td>
<td>Number of target systems</td>
</tr>
</tbody>
</table>
Requirements Validation

- Concerned with demonstrating 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.
- Prototyping is an important technique of requirements validation.

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?

Requirements Reviews

- Regular reviews should be held while the requirements definition is being formulated.
- Both client and contractor staff should be involved in reviews.
- Reviews may be formal (with completed documents) or informal.
- Good communications between developers, customers and users can resolve problems at an early stage.
**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?

**Requirements Evolution**

- Requirements always evolve as a better understanding of user needs is developed and as the organization’s objectives change.
- It is essential to *plan for change* in the requirements as the system is being developed and used.

---

![Diagram of requirements evolution](image-url)
### Requirements Classes

- **Enduring requirements**: Stable requirements derived from the core activity of the customer organization.
  - E.g., a hospital will always have doctors, nurses, etc.
- **Volatile requirements**: Requirements which change during development or when the system is in use.
  - E.g., in a hospital, requirements derived from health-care policy.

### Changing a Requirements Document

- The requirements document should be organized so that requirements changes can be made without extensive rewriting.
- External references should be minimized and the document sections should be as modular as possible.
- Changes are easiest when the document is electronic.

### Controlled Evolution

![Diagram showing the evolution of requirements and systems](image)
“If a company wishes to let a contract for a large software development project, it must define its needs in a sufficiently abstract way that a solution is not pre-defined. The requirements must be written so that several contractors can bid for the contract, offering, perhaps, different ways of meeting the client organisation’s needs. Once a contract has been awarded, the contractor must write a system definition for the client in more detail so that the client understands and can validate what the software will do. Both of these documents may be called the requirements document for the system.”