Software Engineering

Professor Dr. Safa Amir Najim
Computer Information System Dept.
College of CS and IT
University of Basrah
2019-2020

Software Requirement

Chapter 3

Requirements: Introduction

- *Requirements* =services the system is expected to provide + constraints placed on the system
- *Requirements engineering* = gathering, negotiating, analyzing, and documenting requirements
- The requirements could be expressed at various levels of abstraction
- The way requirements are defined has a major impact on the development of the software product.

- **Classification of requirements:**
  - ***User requirements**:* higher level description of services requested and constraints imposed
  - ***System requirements**:* a more detailed, structured description of services and constraints. Usually included in the contract between the developer and the client
- An even more detailed description of requirements can be provided in a *software design specification(closer to implementation)*

Examples of *user requirements definition and system requirements specification*

Requirements definition

1. The software must provide a means of representing and accessing external files created by other tools.

Requirements specification

1.1 The user should be provided with facilities to define the type of external files.
1.2 Each external file type may have an associated tool which may be applied to the file.
1.3 Each external file type may be represented as a specific icon on the user's display.
1.4 Facilities should be provided for the icon representing an external file type to be defined by the user.
1.5 When a user selects an icon representing an external file, the effect of that selection is to apply the tool associated with the type of the external file to the file represented by the selected icon.

Types of software system requirements

- *Functional requirements,* describe the requested functionality/behaviour of the system: services (functions), reactions to inputs, exceptions, modes of operations
- *Non-functional requirements,* are constraints on the services offered by the system, and on the development process
- *Domain requirements,* can be either functional or non-functional and reflect the particularities of the application domain

Functional requirements:

- Depend on the system, the software, and the users
- Can be expressed at different levels of detail (user/system requirements)
- For a system, it is desirable to have a complete and consistent set of functional requirements
    - *Completeness:* all required system facilities are defined
    - *Consistency:* there are no contradictions in requirements

Non-functional requirements:

- Many apply to the system as a whole
- More critical than individual functional requirements
- More difficult to verify

<u>Kinds of non-functional requirements:</u>

- Product requirements
- Organizational requirements
- External requirements

Product requirements:	specify that the delivered product *must behave* in a particular way *e.g. execution speed, reliability, etc.*
Organizational requirements:	are a consequence of *organizational policies* and procedures *e.g. process standards used, implementation requirements, etc.*
External requirements:	arise from *factors which are external* to the system and its development process *e.g. interoperability requirements, legislative requirements, etc.*

Requirements Measures

Property	***Measure***
Speed	Processed transactions/second User/Event response time Screen refresh time
Size	K Bytes; Number of RAM chips
Ease of use	Training time Rate of errors made by trained users Number of help frames

Requirements Measures

Property	***Measure***
Reliability	Mean time to failure Probability of unavailability Rate of failure occurrence
Robustness	Time to restart after failure Percentage of events causing failure Probability of data corruption on failure
Portability	Percentage of target dependent statements Number of target systems

User Requirements

A statement in natural language plus diagrams of the services the system provides and its operational constraints. Based on information from Client, and written for him (or even by him)

- Should be understood by the user, and should not address design and implementation aspects
- Should focus on the key facilities required

Problems with requirements written in natural language:

- Lack of clarity, ambiguity, various interpretations possible
- Confusion, lack of separation between different types of requirements
- Mixture of several requirements in the same statement
- Hard to modularize and thus hard to find connections between requirements

Guidelines for writing requirements:

- Create and use a standard format for the entire software requirements specification
- Highlight important parts of the requirement statements
- Use consistently the language (difference between “should” and “shall”)

Editor example: detailed user requirement

3.5.1 Adding nodes to a design

3.5.1.1 **The editor shall provide a facility for users to add nodes of a specified type to their 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 should move the cursor to the approximate node position in the diagram and indicate that the node symbol should be added at that point.
3. The user should then drag the node symbol 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

System Requirements

- More detailed specifications of user requirements
- Included in the contract with the client
- Used by developers as basis for design
- May be specified using various models (object-oriented models, data-flow diagrams, formal specs, etc.)
- Should indicate WHAT the system is required to do (not HOW) and under what conditions and constraints

Editor example: System requirement

ECLIPSE/Workstation/Tools/DE/FS/3.5.1

Function Add node

Description 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.

Inputs Node type, Node position, Design identifier.

Source Node type and Node position are input by the user, Design identifier from the database.

Outputs Design identifier.

Destination The design database. The design is committed to the database on completion of the operation.

Requires Design graph rooted at input design identifier.

Pre-condition The design is open and displayed on the user's screen.

Post-condition The design is unchanged apart from the addition of a node of the specified type at the given position.

Side-effects None

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

H.W.

- What is the difference between requirements analysis and specification?
- Why is it hard to define and specify requirements?
- What is the difference between functional and non-functional requirements?