Introduction to Software Engineering

Lecture 03.1: Requirements Engineering Part 1

Agenda

- Requirements Engineering
- Functional and Non-Functional Requirements
- Requirements Analysis

Requirements Engineering

What is a Software Requirement?

- A software requirement is a property which must be exhibited in order to solve some problem in the real world.
- Defines the user's expectations from the software.
- A feature that the software needs to satisfy.
- A service that software must provide
- A system constraint under which the software must operate.
- A system property or behavior that can be implicit or explicit.

Types of Requirements

1. Functional Requirements: user requirements, explicit requirements
2. Non-functional Requirements: quality requirements/attributes, implicit requirements
3. UI Requirements: user interface requirements, also known as usability requirement.
4. System Requirements: any requirements the system needs to satisfy to operate or meet the expectation of the users. That, for example, includes data requirements and environmental requirements.

Types of Requirements (cont...)

5. User Requirements: capture the characteristics of the intended user group.
6. Business Requirements: why the organization is undertaking the project. The business goals of the organization or the customer who asked for the product.
7. Market Requirements: requirements that satisfy marketing goals and market needs.

Requirements Engineering

What is requirements engineering?

- Is the process of establishing the services that the user requires from the system and the constraints under which it is to be developed and operated
- Is a set of activities concerned with identifying and communicating the purpose of a software-intensive system, and the contexts in which it will be used

Why Requirements Engineering?

Why requirements engineering is important?

- Software requirements have a significant effect on the software design (e.g. architecture) and implementation (e.g. technologies, and tools).
- Software requirements that have a measurable effects of the system architecture are known as the architecturally significant requirement (ASR)

Architecturally Significant Requirement (ASR)

What is an architecturally significant requirement?

- A requirement that will have a profound effect on the architecture.
- A requirement that strongly enforce specific architectural pattern.
- The presence or absence of such a requirement results in dramatically different architecture.
- Most of the time it is an implicit requirement.

Architecturally Significant Requirement (ASR)

How do we identify an architecturally significant requirement?

- An architecturally significant requirement is most likely:
  - Specifies a software system's quality attribute.
  - Refers to a software system's core features.
  - Impose constraints on a software system.
  - Defines the environment in which the software system will run.

Architecturally Significant Requirement (ASR)

What type of software requirements is the main source of ASR?

How can we measure the importance of an ASR?

Common Activities in Requirements Engineering

Requirements engineering is an iterative process that vary widely depending on the application domain, the people involved and the organisation developing the requirements. However there are some common activities.

- Requirements elicitation and analysis
  - What do the system stakeholders require or expect from the system?
- Requirements specification
  - Defining the requirements in detail
- Requirements validation
  - Checking the validity of the requirements

In-Class Exercise 01

Suggest one key functional, data, environmental, user and usability requirement for a system for use in a **university's self-service cafeteria** that allows users to pay for their food using a credit system.

In-Class Exercise 01

- Functional: The system will calculate the total cost of purchases.
- Data: The system must have access to the price of products in the cafeteria.
- Environmental: Cafeteria users will be carrying a tray and will most likely be in a reasonable rush. The physical environment will be noisy and busy, and users may be talking with friends and colleagues while using the system.

In-Class Exercise 01

User: The majority of users are likely to be under 25 and comfortable dealing with technology.

Usability: The system needs to be simple so that new users can use the system immediately, and memorable for more frequent users. Users won't want to wait around for the system to finish processing, so it needs to be efficient and to be able to deal easily with user errors.

Software System Stakeholders

- Any person or organization who is affected by the system in some way and so who has a legitimate interest
- Stakeholder types
  1. End users
  2. System managers
  3. System owners
  4. System Developers
  5. External stakeholders

Hospital Management System: Mentcare

- Patients whose information is recorded in the system.
- Doctors who are responsible for assessing and treating patients.
- Nurses who coordinate the consultations with doctors and administer some treatments.
- Medical receptionists who manage patients' appointments.
- IT staff who are responsible for installing and maintaining the system.

Hospital Management System: Mentcare

- A medical ethics manager who must ensure that the system meets current ethical guidelines for patient care.
- Healthcare managers who obtain management information from the system.
- Medical records staff who are responsible for ensuring that system information can be maintained and preserved and that record keeping procedures have been properly implemented.

Primary and Secondary Users

- End users of any system could be categorized into two main groups
- Primary User: The primary user is someone who interacts with the system directly. He is in direct contact with the system interface and thus is usually most affected by it.
- Secondary User: The secondary user does not directly interact with the system (user interface of the system) but is still affected by it.
- Example in a hospital management system doctors, nurses, and other health care providers are the primary users. On the other hand patients are the secondary users of the system.

Functional and Non-Functional Requirements

Functional Requirements

- Statements of services the system should provide, how the system should react to particular inputs and how the system should behave in particular situations.
- Mostly an explicit statement.
- A feature of a services the user expect the software system to provide.
- May state what the system should not do.
- Functional testing is the most common testing method to verify/validate the functional requirements.

Functional Requirements Examples

FunReq01: A user shall be able to search the appointments lists for all clinics.

FunReq02: The system shall generate each day, for each clinic, a list of patients who are expected to attend appointments that day.

FunReq03: Each staff member using the system shall be uniquely identified by his or her 8-digit employee number.

Requirements Imprecision

- Problems arise when functional requirements are not precisely stated.
- Ambiguous requirements may be interpreted in different ways by developers and users.
- Consider the term **”search”** in requirement 1
  - User intention – search for a patient name across all appointments in all clinics;
  - Developer interpretation – search for a patient name in an individual clinic. User chooses clinic then search.

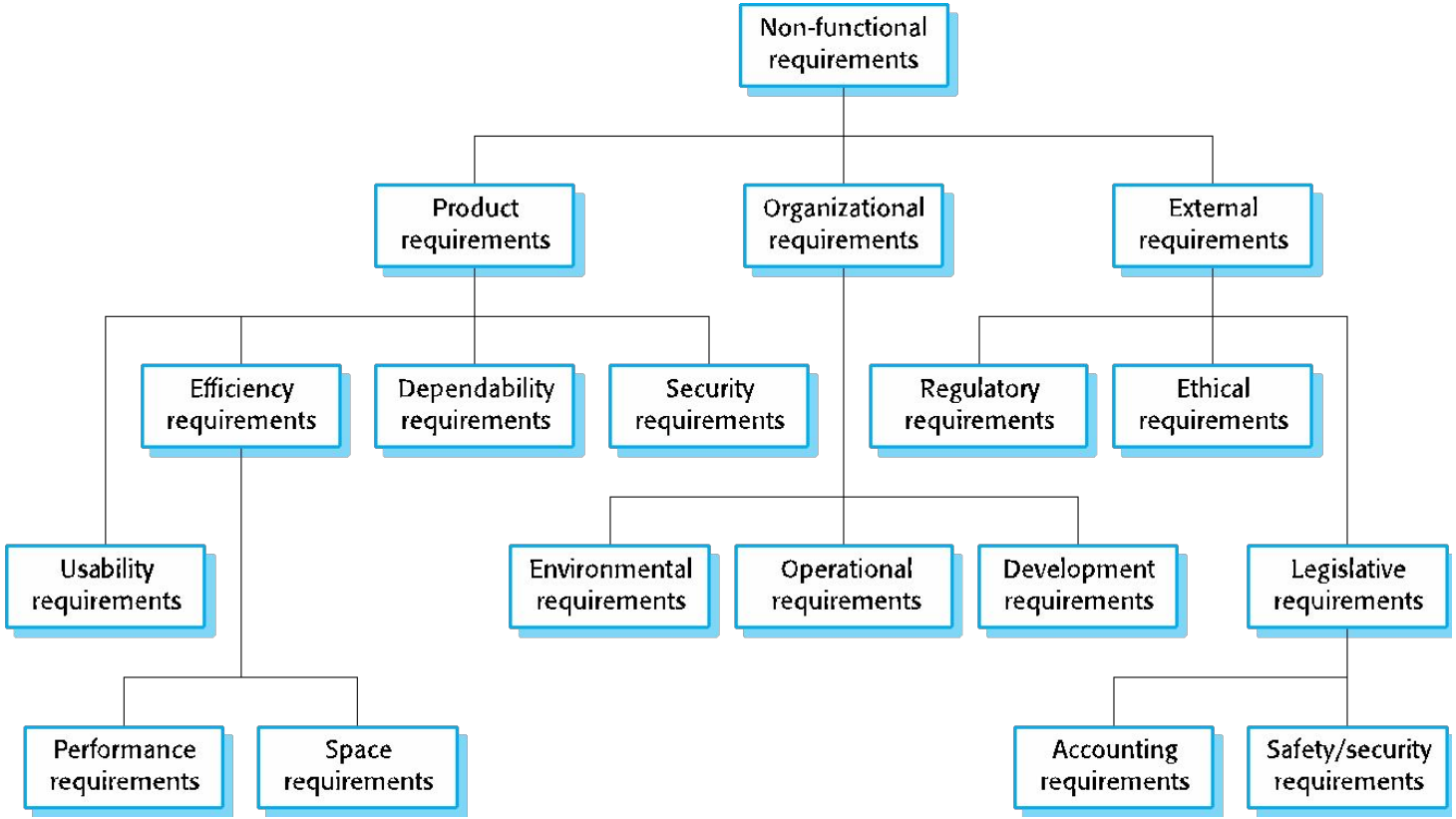
Requirements Completeness and Consistency

- In principle, requirements should be both complete and consistent.
- Complete: They should include descriptions of all facilities required.
- Consistent: There should be no conflicts or contradictions in the descriptions of the system facilities.
- How do we know that the system requirements are complete? Is that even possible?

Non-Functional Requirements

1. Constraints on the services or functions offered by the system such as timing constraints, constraints on the development process, standards, etc.
2. Often apply to the system as a whole rather than individual features or services.
3. Domain requirements, this means constraints on the system from the domain of operation.
4. Non-functional requirements may be more critical than functional requirements. If these are not met, the system may be useless.

Types of Non-Functional Requirements



Quality Attributes (Non Functional Requirements)

- Quality Attributes and Non-functional Requirements (NFRs) are not the same.
- NFRs refers to the set of constraints, qualities, and characteristics the system must satisfy.
- Quality attributes are the largest subset of requirements in the NFRs set.
- Most of the time the success of the software system depends on the quality attributes.

Common Quality Attributes By Category

- Design Qualities
  - Conceptual Integrity
  - Reusability
  - Modifiability
- Run-time Qualities
  - Availability
  - Performance
  - Security
  - Scalability
- System Qualities
  - Maintainability
  - Testability
- User Qualities
  - Supportability
  - Usability

Why Quality Attributes are Important?

- They have a significant influence on the software architecture.
- Quality attributes are architecturally significant requirements (ASRs)
- Systems are frequently redesigned (rearchitect) because they fail to achieve some quality attributes.
- In large-scale software systems quality attributes can never be satisfied in isolation.

Non-Functional Requirements

Product requirement (Run-Time Quality Attribute)
The Mentcare system shall be available to all clinics during normal working hours (Mon–Fri, 0830–17.30). Downtime within normal working hours shall not exceed five seconds in any one day.

Organizational requirement (Run-Time Quality Attribute)
Users of the Mentcare system shall authenticate themselves using their health authority identity card.

External requirement (Not a Quality Attribute)
The system shall implement patient privacy provisions as set out in HStan-03-2006-priv.

In-Class Exercise 02

Give two or more examples User Quality attribute in the Mentcare system

In-Class Exercise: Solution

UsaReq01: The system should be easy to use by medical staff and should be organized in such a way that user errors are minimized. (Goal)

UsaReq02: Medical staff shall be able to use all the system functions after four hours of training. After this training, the average number of errors made by experienced users shall not exceed two per hour of system use. (Testable non-functional requirement)

Non Functional Requirements

- Non-functional requirements may be very difficult to state precisely and imprecise requirements may be difficult to verify.
- Testing and validating non-functional requirements is very difficult.
  - How could we measure the availability of the system?
  - How could we measure the security of the system?
  - How could we measure the user experience?

Property	Measure
Speed	Processed transactions/second User/event response time Screen refresh time
Size	Mbytes Number of ROM chips
Ease of use	Training time Number of help frames
Reliability	Mean time to failure Probability of unavailability Rate of failure occurrence Availability
Robustness	Time to restart after failure Percentage of events causing failure Probability of data corruption on failure
Security	Number of discovered vulnerabilities

Metrics for specifying non-functional requirements

Requirements Analysis

Requirements Elicitation

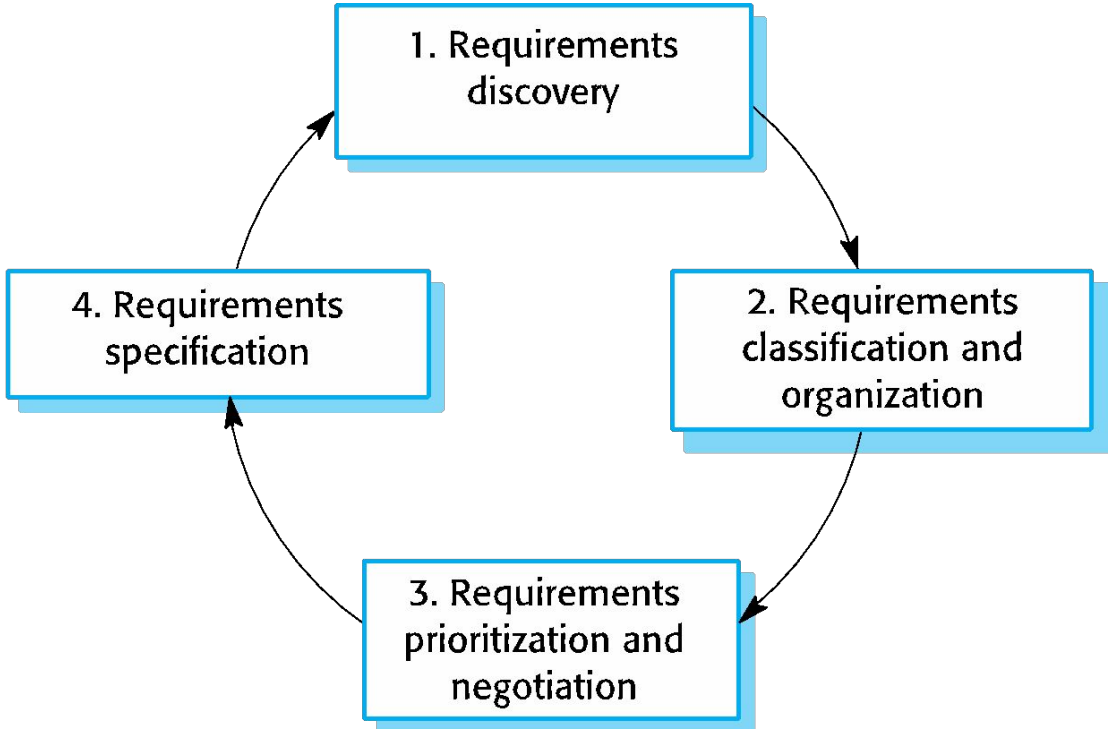
- Sometimes called requirements elicitation, or requirements discovery.
- Involves technical staff working with customers to find out about the application domain, the services that the system should provide and the system's operational constraints.
- May involve end-users, managers, engineers involved in maintenance, domain experts, trade unions, etc. These are called stakeholders.
- How difficult is Requirement Discovery?

In-Class Exercise 03

List three functional requirements for each of the following software system:

1. Course Management Systems
2. Hotel Management Systems
3. Slaughterhouse management system

Main Stages in Requirements Analysis



Requirement Analysis

- Requirements discovery
  - Interacting with stakeholders to discover their requirements. Domain requirements are also discovered at this stage.
- Requirements classification and organisation
  - Groups related requirements and organises them into coherent clusters.
- Prioritisation and negotiation
  - Prioritising requirements and resolving requirements conflicts.
- Requirements specification
  - Requirements are documented to construct the system specification

Requirements Discovery Techniques

- Interviewing: Formal or informal interviews with stakeholders are part of most RE processes.
- Ethnography: A social scientist spends considerable time observing and analyzing how people work.
- Stories and scenarios: domain experts could develop a description of how a system may be used for a particular task or by a particular user. End users could participate in the designing of the stories

Interviewing Stakeholders

- Closed interviews based on pre-determined list of questions
- Open interviews where various issues are explored with stakeholders.
- Normally a mix of closed and open-ended interviewing.
- Interviews are good for getting an overall understanding of what stakeholders do and how they might interact with the system.
- Interviewers need to be open-minded without pre-conceived ideas of what the system should do
- You need to prompt the user to talk about the system by suggesting requirements rather than simply asking them what they want.

Issues with Interview

- Application specialists may use language to describe their work that isn't easy for the requirements engineer to understand.
- Interviews are not good for understanding domain requirements
- Requirements engineers cannot understand specific domain terminology;
- Some domain knowledge is so familiar that people find it hard to articulate or think that it isn't worth articulating (how do you tie your shoes?)

Ethnography

A social scientist spends considerable time observing and analyzing how people work.

People do not have to explain or articulate their work.

Social and organizational factors of importance may be observed.

Ethnographic studies have shown that work is usually richer and more complex than suggested by simple system models.

Scenarios and User Stories

Scenarios and user stories are real-life examples of how a system can be used.

Stories and scenarios are a description of how a system may be used for a particular task.

Because they are based on a practical situation, stakeholders can relate to them and can comment on their situation concerning the story.

Examples: ATM User Stories

As a Customer I want to Login to my account using card and PIN code So that I can perform the transactions.

I want to to deposit check in my bank account through ATM So that I may save my time and perform transactions later.

As a Customer I want to transfer money from my account to another bank account through ATM So that I may save my time.

In-Class Group Activity: 15- 20 minutes

1- each group member should list five functional requirements relevant to his/her group project

2- The members of each group will check their lists of functional requirements to discuss and agree on the list of requirements

3- The members of each group will create a list of 5-7 questions that they can ask other potential users of the project to help in the requirements discovery process.

Questions

References

- Chapter 4, Software-Engineering-10th-Ian-Sommerville
- Chapter 2, The IEEE Guide to the Software Engineering Body of Knowledge
- Part 2, Software Architecture in Practice, 3rd edition, by Bass, Clements, and Kazman