Requirements engineering

Chapter 4

Requirements Engineering
in the textbook

- 4.1 Functional and non-functional requirements
- 4.2 The software requirements document
- 4.4 Requirements engineering processes
- 4.5 Requirements elicitation and analysis
- **4.3** Requirements specification
- 4.6 Requirements validation
- 4.7 Requirements management

What are requirements?

- Sommerville:
  The descriptions of what the system should do:
  - the services that the **customer** requires
  - the constraints on its operation
- IEEE standard glossary of software engineering terminology
  - A condition or capability needed by a **user** to solve a problem or achieve an objective.
- (Wiegers, *Software Requirements* 2): A property that a system must have to provide value to a **stakeholder**

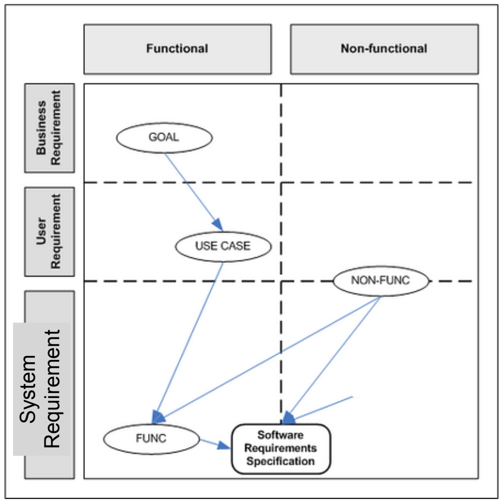
"the system" = the software system to be developed

What is requirements engineering?

- The process of
  - finding out (elicitation)
  - analyzing (analysis)
  - documenting (specification)
  - and checking (validation)

  the required services and constraints of the system to be developed
- Result: software requirements specification document.
- This is the traditional approach to handling requirements.

Relationship of several types of requirements



Levels of requirements specification

- Business Requirements
  - High level goals of the stakeholders for the system
  - Provides vision and scope
- User requirements
  - in terms of tasks the **users** must be able to perform with the system.
  - including the constraints under which the system must operate
  - Expressed in natural language and diagrams
- System requirements
  - even more detailed descriptions of the system's functions, services and constraints.
  - What the **developers** must implement

Example: User and system level requirements

Set up a league for a Fantasy Football website

USER LEVEL:
This function allows a user to set up a league that other players may join. The user who sets up the league is called the league manager, and decides who is able to join the league, and when the league draft will occur.

SYSTEM LEVEL:
1. The system shall provide a way for the user to enter the name of the league
2. The system shall verify that the league name is unique. If not, the system shall ask the user to re-enter (or cancel).
3. The system shall provide a way for the user to enter a password.
4. If the password does not pass the requirements, the system shall ask the user to re-enter (or cancel to exit).
5. The system shall prompt the user for a time and date for the draft
6. The date must occur at least 24 hours before the first game of the regular season (and sometime after the current time). If the date does not meet this criteria, the system shall prompt the user to reenter (repeat until there is a valid date).

4.1 Functional and non-functional requirements

- Functional requirements
  - Specific services or functions the system must provide.
  - how the system reacts to certain inputs
  - Software functionality that the developers must build into the product to enable users to accomplish their tasks.
- Non-functional requirements
  - Constraints on the services or functions offered by the system.
  - these often apply to the system as a whole rather than individual features or services.
  - **How** the system must function.
  - Example: performance, security, or availability requirements.

4.1.1 Functional requirements

- Describe functionality or system services.
- Functional **user** requirements may be high-level statements of what the system should do.
- Functional **system** requirements should describe the system services in detail, at the level a developer could use to implement the feature.

See slide 7 for an example of these

Example: User level Functional requirements

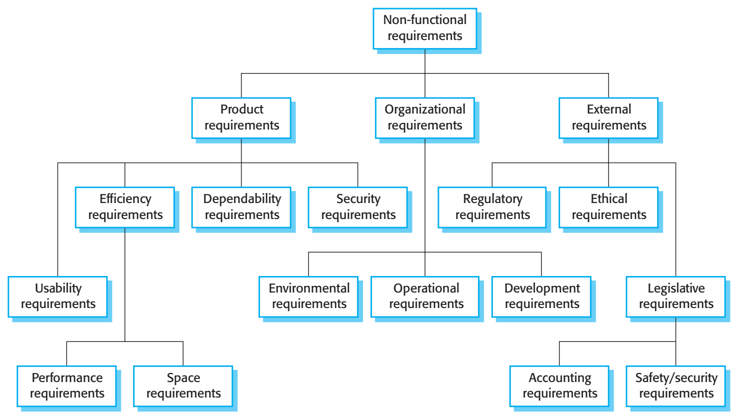
for a system used to maintain information about patients receiving treatment for mental health problems.

1. The system shall generate each day, for each clinic, a list of patients who are expected to attend appointments that day.
2. A user shall be able to search the appointments lists for all clinics.
3. Each staff member using the system shall be uniquely identified by his or her 8-digit employee number.

4.1.2 Non-functional requirements

- Define system properties and constraints.
  - performance
  - reliability
  - security
  - usability
  - must run on certain platform or operating system
  - must be written in a certain programming language
- Non-functional requirements may be more critical than functional requirements.

Types of non-functional requirements



categories described on next slide

Non-functional classifications

- Product requirements
  - How fast is must execute, how much memory it can use
  - Acceptable failure rate
  - How access will be controlled, how easy it must be to use
- Organizational requirements (developer+customer)
  - hardware+software systems it must work with
  - what operational process it must fit into
  - language+IDE+process that must be used
- External requirements (outside factors)
  - what it must do to be approved by a regulator
  - what legislation it must follow to be legal
  - what it must do to be ethical

"it" = the system being developed

Examples of nonfunctional requirements

Product requirement
The system shall be available to all clinics during normal working hours (Mon–Fri, 08.30–17.30). Downtime within normal working hours shall not exceed five seconds in any one day.

Organizational requirement
Users of the system shall authenticate themselves using their health authority identity card.

External requirement
The system shall implement patient privacy provisions as set out in HStan-03-2006-priv.

Non-functional requirements implementation

- Non-functional requirements may affect the overall architecture of a system rather than the individual components.
  - For example, to ensure that performance requirements are met, you may have to organize the system to minimize communications between components.
- A single non-functional requirement, such as a security requirement, may generate a number of related functional requirements that define system services that are required.
  - It may also generate requirements that restrict existing requirements.

Characteristics of excellent requirements

- **Correct**
  - They should reflect the actual needs of the stakeholders (no gold plating).
- **Unambiguous**
  - They should not be able to be interpreted in different ways by different people.
- **Complete**
  - All services required by the user(s) must be defined.
  - Each requirement must fully describe the functionality to be delivered.
- **Consistent**
  - There should be no conflicts or contradictions in the descriptions of the system facilities.
- **Verifiable**
  - They should be written in a way so that the completed system could be tested against them.

Ambiguous requirements

- Capable of being interpreted in different ways by different people (it has more than one interpretation).
- Consider this requirement from a previous example:

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

  - User intention – given a name, search across all appointments **in all clinics**;
  - Developer interpretation – given a name and a clinic, search in **the individual clinic only**. User chooses clinic then search.

Requirements must be verifiable

- Must be able to determine if finished system meets the requirement(s).
- May be difficult for non-functional requirements
- For example: “Easy to use” cannot be measured or tested.

Non-verifiable vs. verifiable requirement

- [non-verifiable] The system should be easy to use by medical staff and should be organized in such a way that user errors are minimized.
- [verifiable] 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.

Metrics for specifying nonfunctional requirements

- A few examples from Table 4.5:

Property	**Measure**
Speed	Processed transactions/second User/event response time
Ease of Use	Training time
Reliability	Mean time to failure Rate of failure occurrence

4.2 The software requirements document

- Software Requirements Specification (SRS)
  - Official statement of
  - What will be implemented
- Should include:
  - User requirements
  - Detailed system requirements
  - Including functional and non-functional requirements
- It is NOT a design document.
  - Should not indicate HOW the features will be implemented

Software Requirements Doc Users/Uses

- Set of users (readers):
  - System customers
  - Project managers
  - System developers
  - System test engineers
  - System maintenance engineers
- Uses:
  - Understand scope of system
  - Project planning
  - Design and implementation
  - System testing
  - User documentation

Requirements document variability

- Level of detail, length, and format depends on:
  - Type of system being developed
  - Size of system
  - Development process (software process) used
  - Presence of safety-critical features (formal notation)
- Incremental development: Incremental SRS.
  - Revise SRS at beginning of each iteration
  - Reviewed and approved requirements document: Baseline SRS
  - Must have a baseline SRS for each iteration/cycle
  - Changes to the baseline during development must have special approval

IEEE Standard SRS template

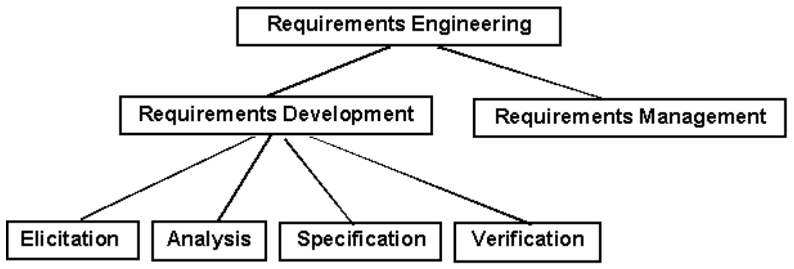
Table of Contents

IEEE Std 830-1998 (on TRACS)

SRS writing: good practices

- Label sections, subsections, requirements consistently
  - Don't ever renumber/relabel requirements
  - Use sequential numbers OR
  - Hierarchical numbers or labels (1.1.2.3 or ship.table.col.sort)
- Use "TBD" as a placeholder for missing info
  - Resolve before accepting as baseline SRS
- Cross reference other documents (avoid duplication)
- User interface elements
  - Don't include screenshots in SRS (they are design specifications)

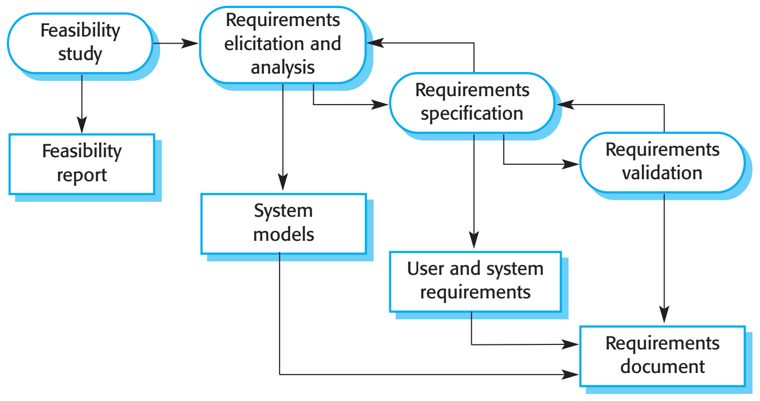
4.4 Requirements engineering processes



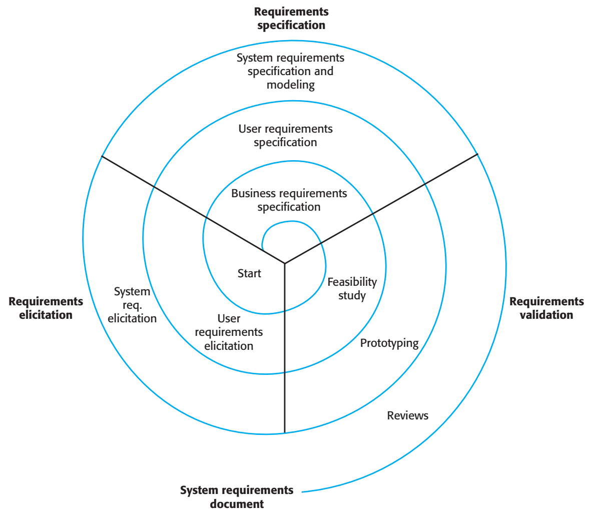
Requirements development
Sub-disciplines

- Elicitation
  - Interview and observe users to identify requirements
  - Hold facilitated elicitation workshops
- Analysis
  - Organize requirements (into groups)
  - Use models to depict the requirements
- Specification
  - Carefully record requirements in a repository (document)
  - Uniquely label, and record source of requirement
- Validation
  - Inspect the requirements
  - Verify the quality of the requirements

Fig 2.4 The requirements development process



A spiral view of the requirements development process (figure 4.12)


Requirements specification
System requirements specification and modeling
User requirements specification
Business requirements specification
Start
Feasibility study
Requirements elicitation
System req. elicitation
User requirements elicitation
Prototyping
Requirements validation
Reviews
System requirements document

Actors in Requirements Development

- Requirements Analyst (or Requirements Engineer)
  - Work with customers to gather, analyze, and document requirements
  - Developer may work in this role
- Stakeholders
  - customers, end users, legal staff
  - maybe members of developer organization

Stakeholders in MHC-PMS

- **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.
- A **medical ethics manager** who must ensure that the system meets current ethical guidelines for patient care.
- **Health care 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.

4.5.1 Requirements Elicitation

- What is the goal of this discipline?

  Identify needs and constraints of stakeholders
- What methods are used to carry it out?
  - Interviews: meet with stakeholders one on one
  - Elicitation workshops: panel or forum of stakeholders
  - Ethnography: observation/immersion in user environment
- What are some tools that the requirements analyst can use?
  - Scenarios: stories that describe specific interactions
  - Use Cases: generalized descriptions of interactions

Scenarios and Use cases

- What is a scenario?
  - Description of one (or more) specific interaction(s) between a specific user and the system
  - A narrative, a story, in language the user understands
- What is a use case?
  - Description of a single general interaction of an actor (or role) with the system.
  - Includes all alternatives that may occur during the interaction
- What is a use case diagram?
  - Consists of an actor (stick figure) and the name of the use case in an oval
  - Description of each use case is documented elsewhere

Scenario for “collecting medical history” in MHC-PMS

John, the patient, meets with Rita the receptionist.
Rita searches for John's record in the system by family name ("Doe"). There's already a Doe in the system so she uses his first name and birthdate to find his record.
Rita chooses to add medical history to John's record.
Rita follows the prompts from the system and enters information about John's previous consultation with a psychiatrist that he had because he had been feeling suicidal. She also enters information about his existing medical conditions (he has diabetes) and the medication he is currently taking (Metformin). He has no allergies, so she leaves that blank. She fills in information about his home life (he is single, recently divorced, living alone).
John's record is then entered into the database, and this fact is recorded into the system log showing the start and end time of the session and Rita's name.

Use case for “collecting medical history” in MHC-PMS, page 1

Initial assumption: The patient has seen a medical receptionist who has already created a record in the system containing his/her personal information (name, address, age, etc.). A nurse is logged into the system.

Normal Interaction:
The nurse searches for the patient by family name. If there is more than one patient with the same surname, the first name and date of birth are used to identify the patient.

The nurse chooses an option to add medical history.

The nurse then follows a series of prompts from the system to enter information about consultations elsewhere on mental health problems (free text input), existing medical conditions (selected from a list), medication currently taking (selected from a list), allergies (free text) and home life (filling out a form).

Use case for “collecting medical history” in MHC-PMS, page 2

What can go wrong:
The patient's record cannot be found: The nurse should create a new record containing the patient's personal information.

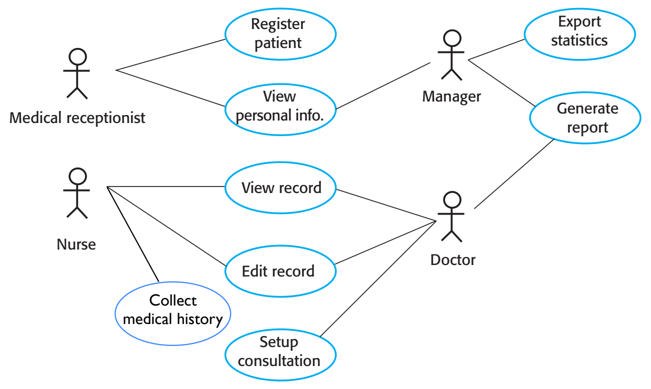
The patient conditions or medication are not on the list:The nurse should choose the 'other' option and describe the condition/medication (free text).

The patient cannot/will not provide information on medical history:The nurse should enter a description of the patient's inability/unwillingness to provide information. The system should print the standard exclusion form stating that the lack of information may mean that treatment will be limited or delayed. This should be signed by the patient, and a copy given to the patient.

Other activities:
Record may be consulted but not edited by other staff members while information is being entered.

System state on completion: The patient record, including the medical history is entered into the database. This fact is recorded in the system log, along with the start and end time of the session and the name of the nurse involved.

Use case diagram for the MHC-PMS



Each oval represents a single use case.
Each one should also have a textual description.

Requirements Analysis

- What is the goal of this discipline?

  Develop good quality, detailed requirements

- What methods are used to carry it out?
  - Modeling: represent requirements in a model, refine models
    - user interactions with system and/or the problem space
  - Prototypes: use to clarify and explore requirements with users
- What are some tools that the requirements analyst can use?
  - Use cases and scenarios
  - Various UML models (ch. 5)

4.3 Requirements Specification

- What is the goal of this discipline?

  Translate collected user needs and constraints into written requirements

- What format can the specifications take?
  - Natural language sentences
  - Structured specifications (forms/templates)
  - Graphical notations (UML diagrams, etc.)
  - Mathematical specifications: clear, but not universal

Specifying user and system requirements

- Both User and System level requirements must be specified
  - See slide 7
- User requirements have to be understandable by end-users and customers who do not have a technical background.
  - Natural language is good for user requirements
  - Be careful to avoid technical jargon or formal notation.
- System requirements are more detailed requirements and may include more technical information.

Natural language specification

- Natural language is expressive, intuitive and universal.
  - The requirements can be understood by users and customers.
- Natural language is vague and ambiguous.
  - Assumed meaning often depends on the background of the reader.
- What are good guidelines for writing specifications in natural language?
  - Use a standard format (see next page)
  - Use active voice (the system shall ...)

Example requirements for the insulin pump software system

3.2 The system shall measure the blood sugar and deliver insulin, if required, every 10 minutes. *(Changes in blood sugar are relatively slow so more frequent measurement is unnecessary; less frequent measurement could lead to unnecessarily high sugar levels.)*

3.6 The system shall run a self-test routine every minute with the conditions to be tested and the associated actions defined in Table 1. *(A self-test routine can discover hardware and software problems and alert the user to the fact the normal operation may be impossible.)*

- These are user requirements
- Each is one sentence, along with a "rationale" statement

Structured specifications

- An approach to writing requirements using templates to write them in a standard way.
- Works well for some types of requirements but is often too rigid for writing business system requirements.
  - Name and description of the function or entity.
  - Description of inputs and where they come from.
  - Description of outputs and where they go to.
  - Description of the action to be taken.
  - Information about the information needed for the computation and other entities used.
  - Pre and post conditions (if appropriate).
  - The side effects (if any) of the function.

A structured specification of a requirement for an insulin pump (p1)

Insulin Pump/Control Software/SRS/3.3.2	
Function	Compute insulin dose: safe sugar level.
Description	Computes the dose of insulin to be delivered when the current measured sugar level is in the safe zone between 3 and 7 units.
Inputs	Current sugar reading (r2); the previous two readings (r0 and r1).
Source	Current sugar reading from sensor. Other readings from memory.
Outputs	CompDose—the dose in insulin to be delivered.
Destination	Main control loop.

A structured specification of a requirement for an insulin pump (p2)

Action	CompDose is zero if the sugar level is stable or falling or if the level is increasing but the rate of increase is decreasing. If the level is increasing and the rate of increase is increasing, then CompDose is computed by dividing the difference between the current sugar level and the previous level by 4 and rounding the result. If the result, is rounded to zero then CompDose is set to the minimum dose that can be delivered.
Requirements	Two previous readings so that the rate of change of sugar level can be computed.
Pre-condition	The insulin reservoir contains at least the maximum allowed single dose of insulin.
Post-condition	r0 is replaced by r1 then r1 is replaced by r2.
Side effects	None.

- This is a system requirement

4.6 Requirements Validation

- What is the goal of this discipline? **Ensure requirements demonstrate desired quality characteristics**
- What are the desired characteristics?
  - See slide 16
- What methods are used to carry it out?
  - Requirements reviews (inspections): analyzed formally by stakeholders and developers
  - Test case generation: can reveal problems in requirements: ambiguity, vagueness, omissions
- How successful is this process?
  - Somewhat, it's a very difficult problem.

Requirements Development

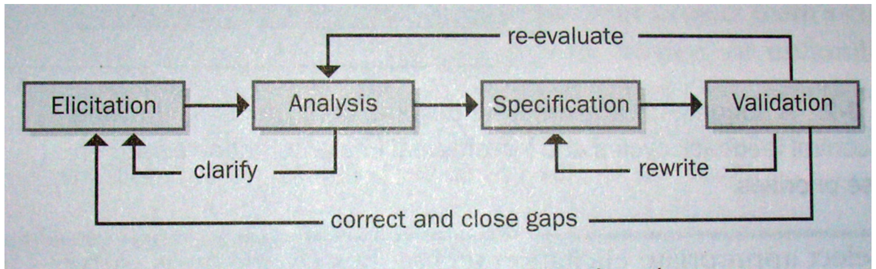


Figure 3-1 Requirements development is an iterative process.

4.7 Requirements Management

- Problem: the requirements specification document will need to change after development begins.
  - Errors may be found in the requirements
  - Users needs change
  - Business needs change
- What are the effects of changing the set of requirements during development?
  - Rework: re-do design and implementation, if already started.
  - Rewrite part of the requirements specification doc

Requirements Management

- Who should decide what changes should be accepted?
  - Developers?
  - Customers/Users?
  - Project managers?
  - Requirements Analyst?
- Change Control board
- How do they decide?
  - change is proposed, validated against existing requirements
  - proposal is evaluated for impact and cost
  - if approved, requirements doc, design and implementation are updated

Requirements Management summary

- Includes all activities that maintain the integrity, accuracy, and currency of the requirements document as the project progresses.
  - Controlling changes to requirements baseline (Change control board)
  - Controlling versions of the requirements document (revision control/version control/source control software)