Ch 4: Requirements Engineering

- ✧ Functional and non-functional requirements
- ✧ The software requirements document
- ✧ Requirements specification
- ✧ Requirements engineering processes
- ✧ Requirements elicitation and analysis
- ✧ Requirements validation
- ✧ Requirements management

What are requirements?

- ✧ The descriptions of what the system should do:
  - the services that the customer requires
  - the constraints on its operation

  (Sommerville)
- ✧ IEEE standard glossary of software engineering terminology
  - A condition or capability needed by a user to solve a problem or achieve an objective.
  - A condition or capability that must be met or possessed by a system or system component to satisfy a contract, standard, specification or other formally imposed document.
- ✧ A property that a product must have to provide value to a stakeholder (Wiegers, Software Requirements 2)

What is requirements engineering?

- ✧ The process of
  - finding out
  - analyzing
  - documenting
  - and checking the required services and constraints
- ✧ The traditional approach to handling requirements.

Levels of requirements

- ✧ Business Requirements
  - High level objectives/goals of the stakeholders for the product
  - Why they want the system, what goals they are trying to achieve.
  - Vision + scope
- ✧ User requirements
  - What the user will be able to do with the product.
  - Goals or tasks the users must be able to perform with the product.
  - Also the constraints under which it must operate
  - natural language + diagrams
- ✧ System requirements
  - Detailed descriptions of the system's functions, services and operational constraints.
  - detailed versions of the user requirements
  - What the developers must implement

Example: User and system requirements

User requirement definition

1. The MHC-PMS shall generate monthly management reports showing the cost of drugs prescribed by each clinic during that month.

System requirements specification

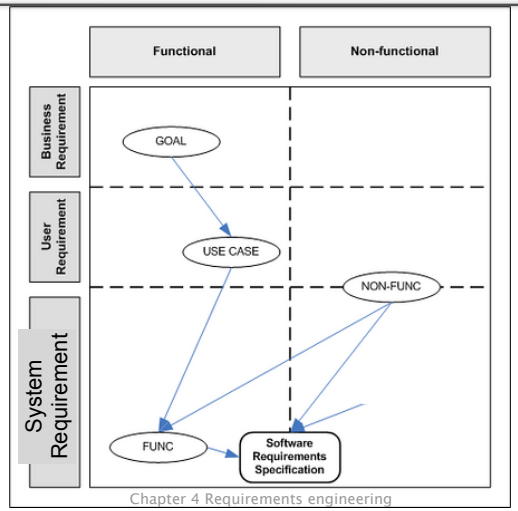
1.1 On the last working day of each month, a summary of the drugs prescribed, their cost and the prescribing clinics shall be generated.
1.2 The system shall automatically generate the report for printing after 17.30 on the last working day of the month.
1.3 A report shall be created for each clinic and shall list the individual drug names, the total number of prescriptions, the number of doses prescribed and the total cost of the prescribed drugs.
1.4 If drugs are available in different dose units (e.g. 10mg, 20 mg, etc.) separate reports shall be created for each dose unit.
1.5 Access to all cost reports shall be restricted to authorized users listed on a management access control list.

4.1 Functional and non-functional requirements

✧ Functional requirements
- Specific services or functions the system must provide.
- how it 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.
- often apply to the system as a whole rather than individual features or services.
- How the system must function.

Relationship of several types of requirements



4.1.1 Functional requirements

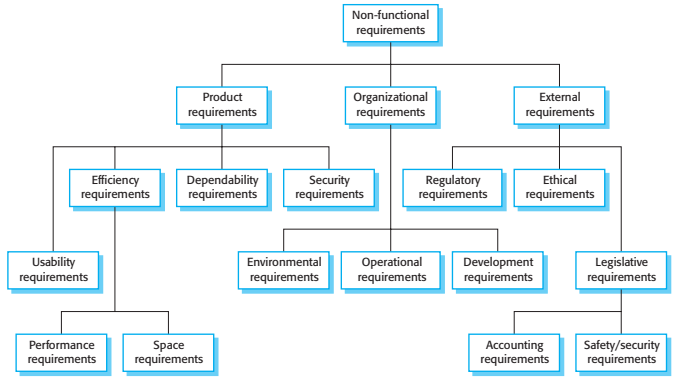
Example: Functional requirements for the MHC-PMS

1. A user shall be able to search the appointments lists for all clinics.
2. The system shall generate each day, for each clinic, a list of patients who are expected to attend appointments that day.
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 nonfunctional requirement



Non-functional classifications

- ✧ Product requirements
  - execution speed, reliability, etc.
- ✧ Organizational requirements
  - from policies and procedures
- ✧ External requirements
  - from factors external to the system and its development process

Non-functional requirements implementation

- ✧ May affect the overall architecture of a system, rather than a single component.
- ✧ A single non-functional requirement may generate a number of related functional requirements

Examples of nonfunctional requirements

Product requirement
The MHC-PMS 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
Users of the MHC-PMS 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.

Characteristics of excellent requirements

- ✧ Correct
  - The requirements should reflect the actual needs of the stakeholders.
- ✧ Unambiguous
  - They should not be able to be interpreted in different ways by different people.
- ✧ Complete
  - They should include descriptions of all facilities required.
- ✧ Consistent
  - There should be no conflicts or contradictions in the descriptions of the system facilities.
- ✧ Verifiable
  - The requirements should be written in a way so that the completed system could be tested against them.

Ambiguous requirements

- ✧ May be interpreted in different ways by different people
- ✧ Consider requirement 1 from 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 completeness and consistency

- ✧ Complete
  - Include descriptions of all services required.
- ✧ Consistent
  - Should not conflict or contradict one another.

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

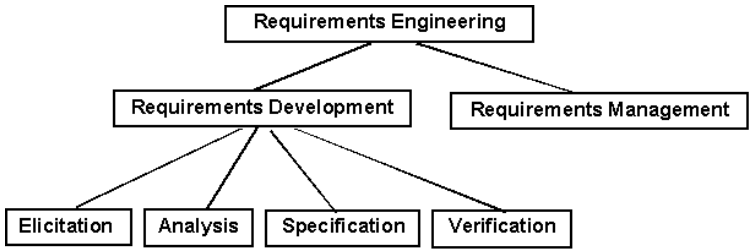
- The system should be easy to use by medical staff and should be organized in such a way that user errors are minimized.
- Medical staff shall be able to use all the system functions after four hours of training. After this training, the average number of errors made by experienced users shall not exceed 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

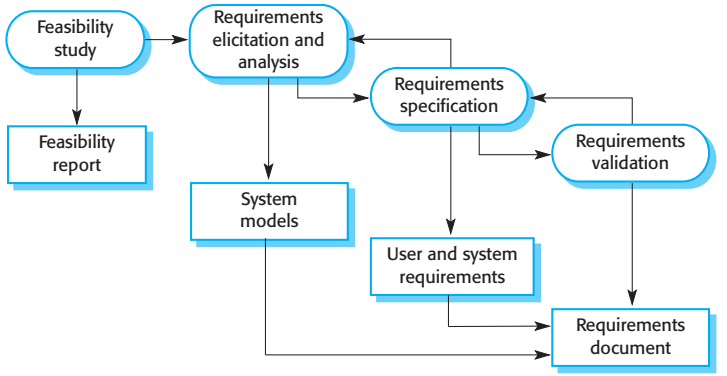
4.4 Requirements engineering discipline



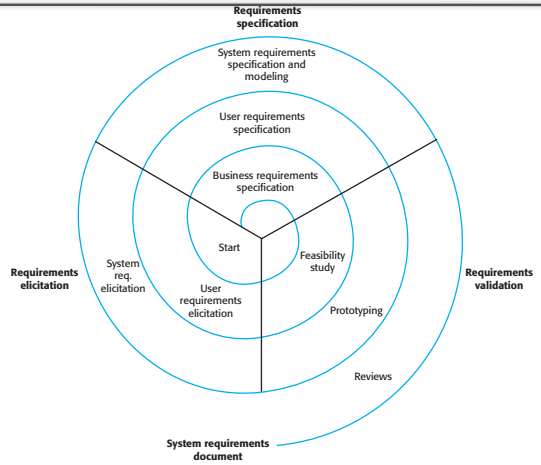
4.4 Requirements engineering processes Good practices

- ✧ Elicitation
  - Interview and observe users to identify use cases
  - 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, record source of requirement
- ✧ Validation
  - Inspect the requirements
  - Write test cases

Fig 2.4 The requirements engineering process



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



Requirements engineering

- ✧ Goal is Software Requirements Specification
  - Collection of requirements (usually a document)
  - High quality
- ✧ Iteration is a key for requirements engineering success.

4.2 The software requirements document

- Software Requirements Specification (SRS)
  - Official statement
  - What will be implemented
- Should include:
  - User requirements
  - Detailed system requirements
  - Functional and non-functional
- It is NOT a design document.

Requirements document variability

- Level of detail, length, and format depends on:
  - Type of application
  - Size of system
  - Development process used
- Incremental development: Incremental SRS.
- Baseline SRS:
  - Reviewed and approved
  - Must have baseline for each development effort
  - Changes to the baseline must be approved

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

The structure of a requirements document

- Standard: IEEE-STD-830-1998
- The standard has 5 sections
  - Section 4: Considerations for producing a good SRS
  - Section 5: The parts of an SRS
- Three main sections:
  - Introduction
  - Overall description
  - Specific requirements
  - Also supporting info: table of contents, indices, appendices

IEEE SRS template

Table of Contents

SRS template from Wiegers 2003

SRS writing: good practices

- ✧ Label sections, subsections, requirements consistently
  - Don't ever renumber/relabel requirements
  - 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 implementation
- ✧ Cross reference other documents (avoid duplication)
- ✧ User interface elements
  - Don't include screenshots in SRS

Actors in Requirements Development

- Requirements Analyst
- Requirements Engineer
  - Work with customers to gather, analyze, 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.

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
- What are some tools that the requirements analyst can use?
  - Scenarios: describes interaction, outline/form
  - Use Cases: diagrams with actors and interactions

Scenario for collecting medical history in MHC-PMS (part 1)

Initial assumption: The patient has seen a medical receptionist who has created a record in the system and collected the patient's personal information (name, address, age, etc.). A nurse is logged on to the system and is collecting medical history.

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

The nurse chooses the menu 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 (nurse selects conditions from menu), medication currently taken (selected from menu), allergies (free text), and home life (form).

Scenario for collecting medical history in MHC-PMS (part 2)

What can go wrong: The patient's record does not exist or cannot be found. The nurse should create a new record and record personal information.

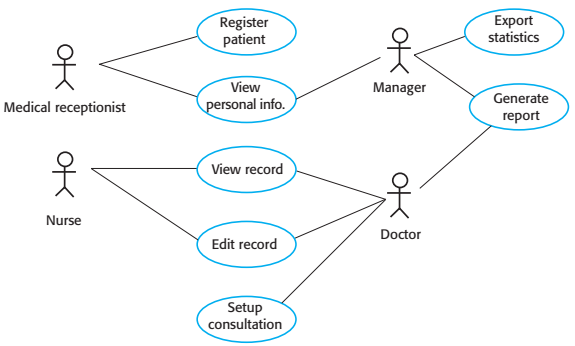
Patient conditions or medication are not entered in the menu. The nurse should choose the 'other' option and enter free text describing the condition/medication.

Patient cannot/will not provide information on medical history. The nurse should enter free text recording 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 and handed to the patient.

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

System state on completion: User is logged on. The patient record including medical history is entered in the database, a record is added to the system log showing the start and end time of the session and the nurse involved.

Use cases for the MHC-PMS

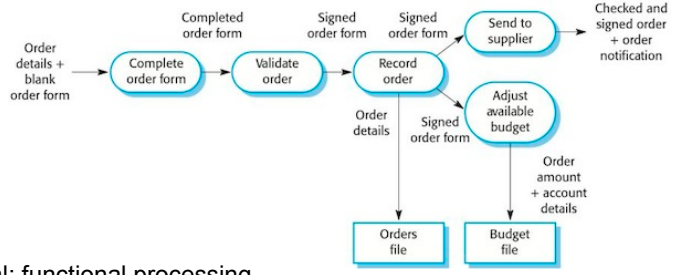


Requirements Analysis

- What is the goal of this discipline?
  Develop requirements of sufficient quality and detail
- What methods are used to carry it out?
  - Modeling: represents requirements in a model, refine
  - Prototypes: use to clarify and explore requirements
- What are some tools that the requirements analyst can use?
  - Data flow diagrams (DFD) (ch5)
  - Entity-Relationship diagram (database design)
  - State transition diagrams (UML-ch5)
  - Class diagrams (UML-ch5)
  - Activity diagrams (UML-ch5)

Most of these models are supported by CASE tools

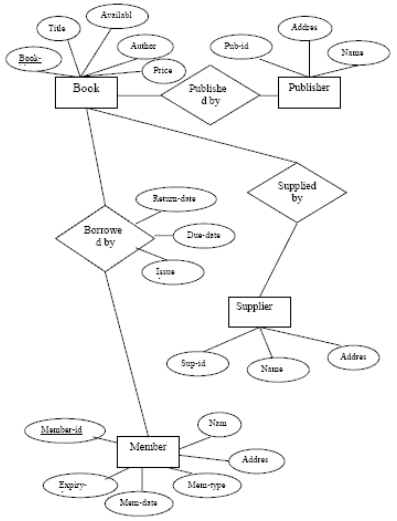
Example Data Flow Diagram: Order Processing



Oval: functional processing
Rectangle: data store
Labeled arrow: data movement

Example Entity Relationship Diagram: Library management

Rectangle: entity
Diamond: relationship
Oval: attributes



Requirements Specification

- What is the goal of this discipline?
  Translate collected user needs and constraints into written requirements
- What format do the specifications take?
  - Natural language sentences: pros and cons
  - Structured specifications: add uniformity to nat. language
  - Graphical notations (UML)
  - Design description languages: clear, not universal
  - Mathematical specifications: clear, not universal
- What are good guidelines for writing specifications?
  - Use a standard format
  - 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.)

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.

Requirements Validation

- What is the goal of this discipline?
  Ensure requirements demonstrate desired quality characteristics
- What are the desired characteristics?
  - See slide 14
- 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

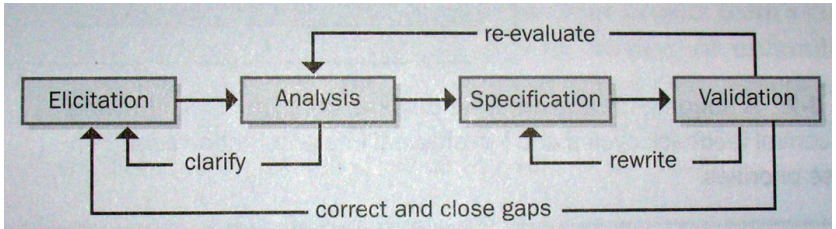


Figure 3-1 Requirements development is an iterative process.

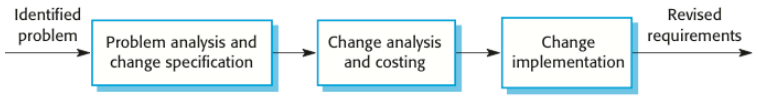
Requirements Management

- 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 requirements
  - proposal is evaluated for impact and cost
  - if approved, requirements doc, design and implementation are updated

Figure 4.18 Requirements change management



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)