Requirements Engineering

- **Requirements:** descriptions of services that a system should provide & the constraints on its operation.

- **Requirements Engineering:** finding out, analyzing, documenting, & checking these services & constraints

- Issues can arise when failure to distinguish btwn user requirements & system requirements

- **User Requirements:** statements in natural language & diagrams of what services the system is supposed to provide

- **System Requirements:** more detailed descriptions of the software system's functions, services, and operational constraints. (exactly what is to be implemented)

User requirements definition

1. The Mentcare system 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 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, 20mg, etc.) separate reports shall be created for each dose unit.
1.5 Access to drug cost reports shall be restricted to authorized users as listed on a management access control list.

4.1 Functional & Non-Functional Requirements

- **Functional Requirements:** statements of services the system should provide, how it should react to particular inputs, and how it should behave in particular situations.

- **Non-functional Requirements:** constraints on the services or functions (i.e. timing, standards)

- **Functional Requirements:**

  - Imprecision in the requirements specification can lead to disputes between customers & software developers

  - Functional requirements should be complete and consistent

↳ However larger systems w/ many stakeholders are likely to have inconsistent needs

• Non-functional Requirements:

- often more critical than functional.
- Implementation of non-functional requirements is often more spread out in the system because:
  1. They may affect overall architecture rather than individual components
  2. An individual requirement may entail several functional requirements to be implemented
- Non-functional requirements can come from:

1. Product Requirements

↳ Efficiency, Dependability, Security, Usability

Performance, Space

2. Organizational Requirements

↳ Environmental, Operational, Development (from customer & developer organizations)

3. External Requirements

↳ Regulatory, Ethical, Legislative

- Common problem: stakeholders propose them as general goals but leave specifics up for interpretation

customers find it difficult to translate goals into measurable requirements

cost of effectively verifying measurable non-functional reqs. can be very high

4.2 Requirements Engineering Process

- Requirements engineering:
  1. elicitation & analysis
  2. specification
  3. validation

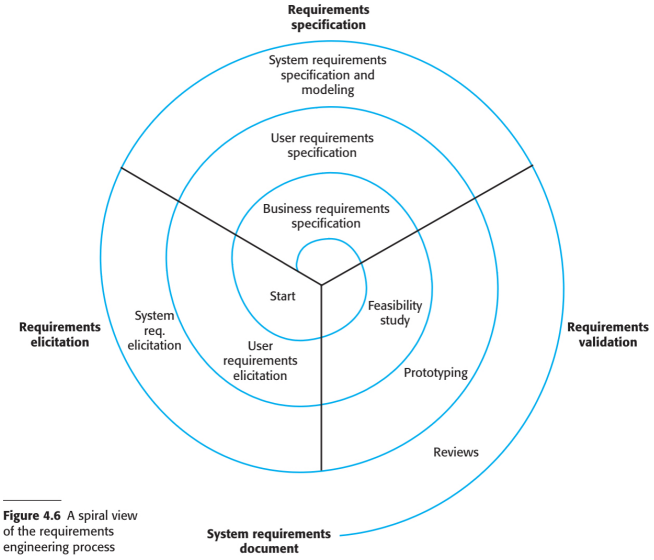


Figure 4.6 A spiral view of the requirements engineering process

4.3 Requirements Elicitation

- Engineers work w/ stakeholders to find out about application domain, work activities, services & system features, req'd performance, hardware constraints

- Difficulties:
  1. Don't know what they want, don't know whats feasible
  2. Non specific; implicit familiarity
  3. Multiple stakeholders w/ multiple requirements
  4. Internal politics
  5. Businesses evolve

- Process activities:
  1. Requirements discovery & understanding
  2. Requirements classification & organization
  3. Requirements prioritization & negotiation
  4. Requirements documentation

• Requirements Elicitation Techniques:

- Two approaches:

1. **Interviewing**; talk to people abt what they do
2. **Observation**; watch people do their job

- Interviewing:

1. **Closed**; predefined agenda & questions
2. **Open**; no agenda

↳ Difficulties:

1. Specialist jargon misunderstandings
2. Forgetful or assumptions which are non-obvious to outsider

↳ Effective interviewing:

1. Open-minded
2. Prompt interviewee

- Ethnography:

helps discover implicit requirements which reflect actual ways ppl work

↳ Effective for discovering:

1. Reqs. derrived from the way people work
2. Reqs. derrived from cooperation

↳ Adv: reveal processes often missed by other methods

Disadv: not effective for discovering broader organizational/domain reqs.

- **Stories & Scenarios:**
  - A description of how the system can be used for a particular task (high-level description)
  - Adv: easily relatable
  - Disadv: undetailed
  - **Scenarios:** more specific & detailed stories of user interaction sessions.
  - Scenarios include:
    1. Description of what the system & users expect when the scenario starts
    2. Description of normal flow of events in the scenario
    3. Description of what can go wrong & how problems

can be handled

4. Information abt other activities that might be going on at the same time.

5. Description of the system state when the scenario ends

4.4 Requirements Specification

- The process of writing down the user & system requirements in a requirements document

- User reqs. describe functional & non-functional reqs. in natural language

- System requirements should only describe external behavior of system & its operational constraints — NOT design / implementation

- This is not always possible b/c:

  1. Initial architecture design may be needed to structure reqs.

  2. Interoperation w/ existing systems can influence design

  3. Specific architectures may be required to satisfy non-func. reqs.

- **Natural Language Specification:**
  - Adv: Expressive, intuitive, universal

- Disadv: vague, ambiguous

- To minimize misunderstandings:

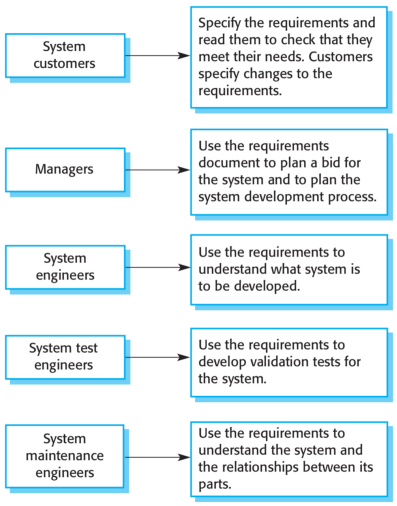
1. Invent standard format for all req. definitions
2. Consistently distinguish btwn. mandatory & desirable (shall vs should)
3. Text highlighting to distinguish key parts
4. Avoid technical language
5. Rationale ("why") w/ each req. (and who proposed it)

- **Structured Specifications:**
  - Using templates to standardize requirements specs.
  - Templates can feature
    1. Description of **function** being specified
    2. Description of **inputs** & their origins
    3. Description of **outputs** & their destinations
    4. Required **information**
    5. **Action** to be taken
    6. **Pre/post conditions**; what must be true before & after function call

7. Side effects

- Adv: removes some problems w/ natural language, variability reduced, organized reqs.
- Disadv: difficult to write clear & unambiguous
  ↳ can be fixed w/ technical specificity in system reqs.

- **Use Cases:**
  - Describing interactions btwn users and a system using a graphical model & structured text
  - Uses high level case diagram
  - Set of use cases represents all possible interactions in system requirements
- **Software Requirements Document:**
  - official statement of what devs. should implement



4.5 Requirements Validation

- The process of checking that reqs. define the system the customer really wants
- Errors can lead to extensive rework costs
- Checks:
  1. **Validity**; reflect real needs
  2. **Consistency**; conflicts
  3. **Completeness**
  4. **Realism**; technologies, budget, schedule
  5. **Verifyability**; can be checked ff
- Validation techniques:
  1. **Requirements Reviews**: check for errors & inconsistencies
  2. **Prototyping**: executable model
  3. **Test-case Generation**: tests devised during validation
- Difficult to confirm all needs are met

4.6 Requirements Change

- Changes arise:
  1. Business & technological environment changes after installata updates, priorities, regulations

2. People who pay for a system are rarely users of that system

3. Stakeholders

- **Requirements Management Planning:**
  - how a set of evolving reqs. will be managed
  - decisions:
    1. **Requirements Identification**
    2. **Change-Management Process**: set of activities that assess impact & cost of changes
    3. **Traceability Policies**: records of reqs. & relationships
    4. **Tool Support**: to process info
  - Tool Support for:
    1. **Requirements Storage**
    2. **Change Management**
    3. **Traceability Management**

- **Requirements Change Management:**
  - formal process means all change proposals are treated same & changes to reqs. docs are made in controlled way
  - 3 principle stages:
    1. **Problem analysis & change specification**: what is proposed change

2. Change analysis & Costing : effects of proposed change

3. Change implementation : make change

- always update docs before making change

Key Points

- Requirements for a software system set out what the system should do and define constraints on its operation and implementation.
- Functional requirements are statements of the services that the system must provide or are descriptions of how some computations must be carried out.
- Non-functional requirements often constrain the system being developed and the development process being used. These might be product requirements, organizational requirements, or external requirements. They often relate to the emergent properties of the system and therefore apply to the system as a whole.
- The requirements engineering process includes requirements elicitation, requirements specification, requirements validation, and requirements management.
- Requirements elicitation is an iterative process that can be represented as a spiral of activities—requirements discovery, requirements classification and organization, requirements negotiation, and requirements documentation.
- Requirements specification is the process of formally documenting the user and system requirements and creating a software requirements document.
- The software requirements document is an agreed statement of the system requirements. It should be organized so that both system customers and software developers can use it.
- Requirements validation is the process of checking the requirements for validity, consistency, completeness, realism, and verifiability.
- Business, organizational, and technical changes inevitably lead to changes to the requirements for a software system. Requirements management is the process of managing and controlling these changes.