Chapter 4: **Requirements Engineering**

- **Requirements** are the **descriptions** of the system **services** and **constraints** that are generated during the requirements **engineering process**

**-Functional:** What the system should do/provide and may also state what the system should not provide

Functional **User** Requirements: high-level statements of what the system should do

Functional **Service** Requirements: describe the system services in detail

**Problems** arise when the requirements aren’t precisely stated (can be interpreted in different ways), they should be complete (include descriptions of all facilities required) and consistent (no conflicts or contradictions)

**-Non-Functional:** Constraints on the services/functions such as reliability, standards, responsiveness, storage, and time

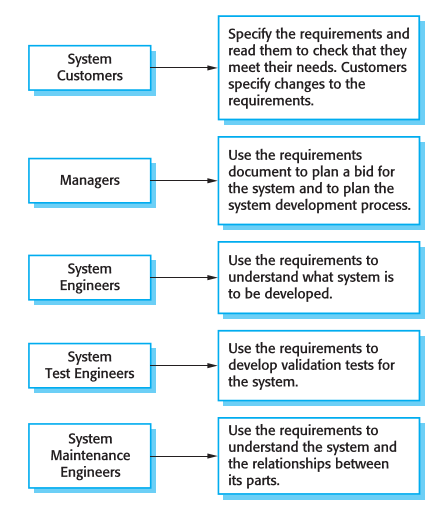
* may be **more critical** than functional (not met => useless system)
* **may affect** the overall architecture of a system rather than the individual components
* may generate a number of related functional requirements
* requirements may be very difficult to state precisely
* Types of Non-functional requirements:

**Product:** Specify that the delivered product must behave in a particular way (execution speed, reliability)

**Organizational:** consequence of organization policies and procedures (process standards used, implementation requirements)

**External**: arise from factors which are external to the system and its development process (interoperability requirements, legislative requirements)

**Metrics** for specifying them: Speed – Size – Ease of Use – Reliability – Robustness - Portability

**Goal:** general intention of the user such as ease of use

Stated by the client and is not helpful to developers as they leave scope for interpretation

**Software Requirements Document:**

**- Official statement** of what is required of the system developers

- Should include both a **definition** of user requirements and a **specification** of the system requirements

- It is **not** a design document as it should set **what** the system should do rather than **how** it should do it

**- Requirements Specification:**

- **User** requirements have to be **understandable** by end-users and customers who do not have a technical background

**- System** requirements are more detailed requirements and may include **more technical** information

-They may be part of a contract for the system development so they should be as **complete** as possible

**- Requirements Engineering:**

- **Iterative** activity that inter-relates Req. **elicitation**,Req. **analysis**, Req. **validation**, and Req. **management**

- **Elicitation and Analysis (**Requirements Discovery**):**

- Involves developers/staff working with customers (employees, managers, stakeholders,etc..) in order to get the application **domain**, the **services** that the system should provide and the system’s operational **constraints**

**- Problems:**

- Stakeholders **don’t know** what they really want

- Stakeholders express requirements in their **own terms**

**- Conflicting** requirements between stakeholders

- Organisational and **political** factors

- Requirements **change** during the analysis process (New stakeholders, business environment, etc..)

- **Interviewing:**

**-** Formal or informal interviews with stakeholders

- Can be closed interviews (pre-determined list of questions) or open interviews (various issues are explored). They are usually a mix of both

- Good for getting an overall understanding of what stakeholders do and how they might interact with the system

- For **effective** interviewing:

- Be open-minded

- Avoid pre-conceived ideas about the requirements

- Be willing to listen to stakeholders

- Prompt the interviewee to get discussions going using a requirements proposal or by working together on a prototype system

**- Requirements Validation:** demonstrating that the requirements define the system that the customer really wants

- Requirement **error costs are high** so validation is very important (100x cost of implementation error)

- **Techniques**:

Requirements **reviews**: Systematic manual analysis of the requirements

**Prototyping**: Using an executable model of the system to check requirements

**Test-case generation**: Developing tests for requirements to check testability

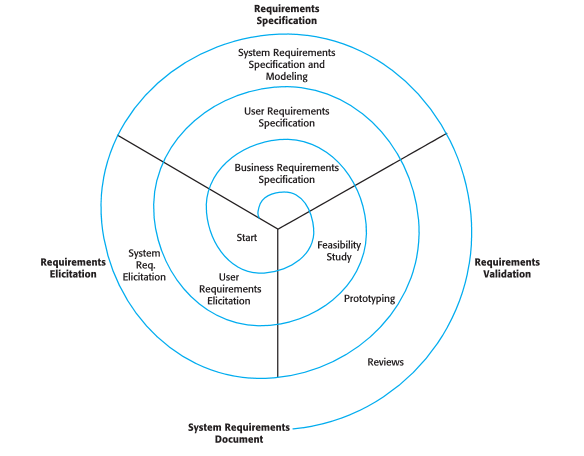
- Requirements Checking:

- **Validity**. Does the system provide the functions which best support the customer’s needs?

- **Consistency**. Are there any requirements conflicts?

- **Completeness**. Are all functions required by the customer included?

- **Realism**. Can the requirements be implemented given available budget and technology?

- **Verifiability**. Can the requirements be checked?

**- Requirements Management:** managing changing requirements

New hardware, interface the system with other systems, business

priorities change, and new regulations

The people who pay for a system and the users of that system

are rarely the same people (customers ordering focus

on budgets and organization