Software Requirements

Book Referred : Ian Sommerville 8th Edition Chapter 6

Topics Covered

- Functional and Non-Functional Requirements
- User Requirements
- System Requirements
- The software requirements document

Objectives

- To introduce the concepts of user and system requirements
- To describe functional and non-functional requirements
- To explain two techniques for describing system requirements
- To explain how software requirements may be organised in a requirements document

Requirement Engineering

- The process of establishing the services that the customer requires from a system and the constraints under which it operates and is developed.
- The requirements themselves are the descriptions of the system services and constraints that are generated during the requirements engineering process.
- The process of finding out, analysing, documenting and checking these services and constraints is called *requirements engineering* (RE).

What is a Requirement?

- It may range from a high-level abstract statement of a service or of a system constraint to a detailed mathematical functional specification
- This is inevitable as requirements may serve a dual function
  - May be the basis for a bid for a contract - therefore must be open to interpretation
  - May be the basis for the contract itself - therefore must be defined in detail
  - Both these statements may be called requirements

Types of Requirement

- User requirements
  - High level abstract requirements
  - Statements in natural language plus tables and diagrams of the services the system provides and its operational constraints.
  - Written for customers
- System requirements
  - A structured document setting out detailed descriptions of the system services.
  - Described with help of system models.
  - Written as a contract between client and contractor

Definition and Specification

User requirement definition

1. LIBSYS shall keep track of all data required by copyright licensing agencies in the UK and elsewhere.

System requirements specification

1.1 On making a request for a document from LIBSYS, the requestor shall be presented with a form that records details of the user and the request made.
1.2 LIBSYS request forms shall be stored on the system for five years from the date of the request.
1.3 All LIBSYS request forms must be indexed by user, by the name of the material requested and by the supplier of the request.
1.4 LIBSYS shall maintain a log of all requests that have been made to the system.
1.5 For material where authors' lending rights apply, loan details shall be sent monthly to copyright licensing agencies that have registered with LIBSYS.

Definition and Specification

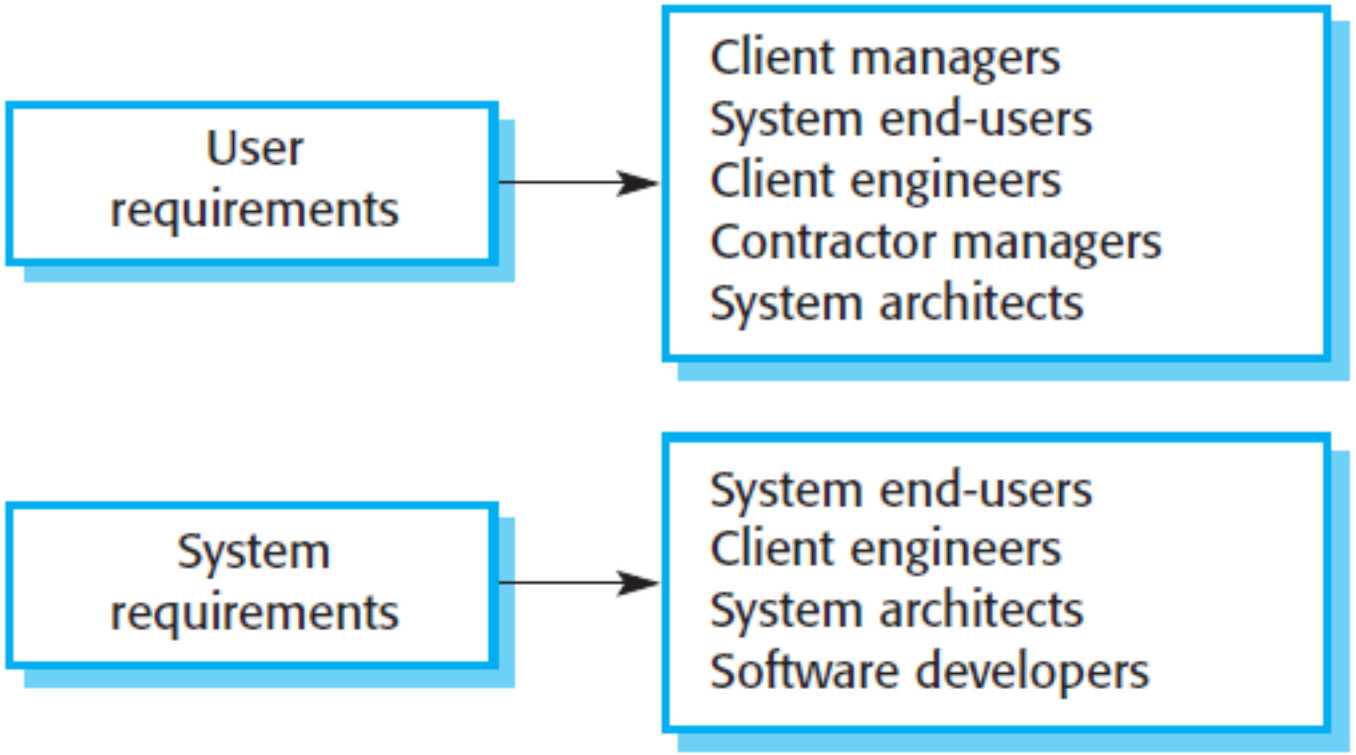
Requirements definition

1. The software must provide a means of representing and accessing external files created by other tools.

Requirements specification

1.1 The user should be provided with facilities to define the type of external files.
1.2 Each external file type may have an associated tool which may be applied to the file
1.3 Each external file type may be represented as a specific icon on the user's display.
1.4 Facilities should be provided for the icon representing an external file type to be defined by the user.
1.5 When a user selects an icon representing an external file, the effect of that selection is to apply the tool associated with the type of the external file to the file represented by the selected icon.

Requirement Readers



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.
- Non-functional requirements
  - Constraints on the services or functions offered by the system such as timing constraints, constraints on the development process, standards, etc.
- Domain requirements
  - Requirements that come from the application domain of the system and that reflect characteristics of that domain

Examples of Functional Requirements

- The user shall be able to search either all of the initial set of databases or select a subset from it.
- The system shall provide appropriate viewers for the user to read documents in the document store.
- Every order shall be allocated a unique identifier (ORDER_ID) which the user shall be able to copy to the account's permanent storage area.

Requirement Imprecision

- Problems arise when requirements are not precisely stated
- Ambiguous requirements may be interpreted in different ways by developers and users
- Consider the term ‘appropriate viewers’
  - User intention - special purpose viewer for each different document type
  - Developer interpretation - Provide a text viewer that shows the contents of the document

Requirement 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

Non-Functional Requirements

- Non-functional requirements (or, design characteristics), such as performance, security, or availability, usually specify or constrain characteristics of the system as a whole.
- Non-functional requirements are often more critical than individual functional requirements.
- However, failing to meet a non-functional requirement can mean that the whole system is unusable.
- For example, if an aircraft system does not meet its reliability requirements, it will not be certified as safe for operation; if an embedded control system fails to meet its performance requirements, the control functions will not operate correctly.

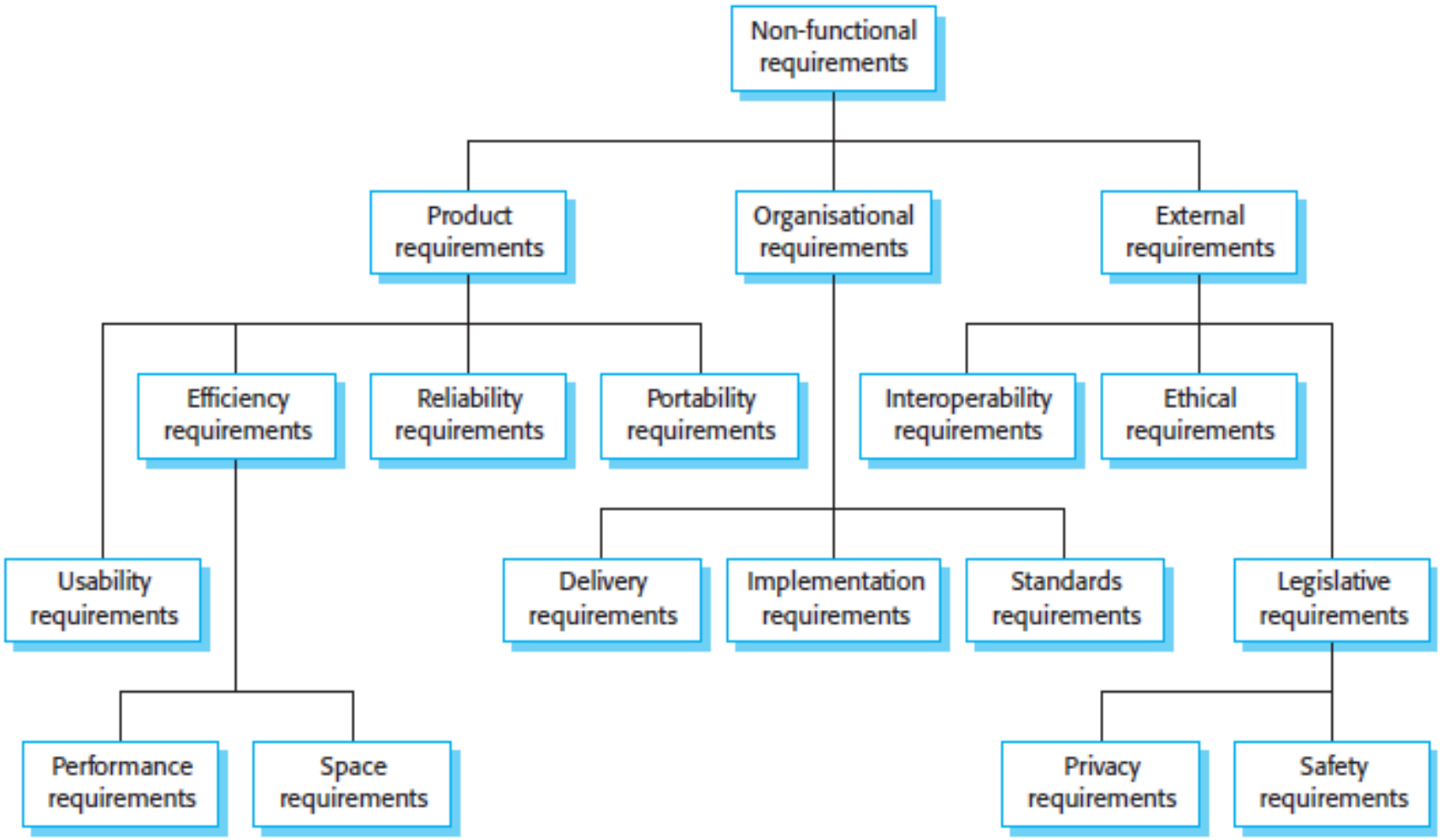
Non-functional Classification

- Product requirements
  - Requirements which specify that the delivered product must behave in a particular way e.g. execution speed, reliability, etc.
- Organisational requirements
  - Requirements which are a consequence of organisational policies and procedures e.g. process standards used, implementation requirements, etc.
- External requirements
  - Requirements which arise from factors which are external to the system and its development process e.g. interoperability requirements, legislative requirements, etc.

Non-functional Classification

- Product requirement
  - It shall be possible for all necessary communication between the system and the user to be expressed in the standard ADA character set.
- Organisational requirement
  - The system development process and deliverable documents shall conform to the process and deliverables defined in IEEE STANDARD-95 FORMAT.
- External requirement
  - The system shall not disclose any personal information about customers apart from their name and reference number to the operators of the system.

Non-functional Classification



FR vs. NFR

- Functional Requirement : specifies something that the system should do.
  - Specifies a behaviour or a function.
  - Includes Business Rules, Transaction corrections, adjustments and cancellations, Administrative functions, Authentication, Authorization levels, Audit Tracking, External Interfaces, Certification Requirements, Reporting Requirements, Historical Data, Legal or Regulatory Requirements
- Non-Functional Requirement : specifies how the system should behave and that it is a constraint upon the system's behaviour.
  - Specifies the quality attributes for a system.
  - Includes Scalability, Capacity, Availability, Reliability, Recoverability, Maintainability, Serviceability, Security, Regulatory, Manageability, Environmental, Data Integrity, Usability, Interoperability

Goals and Requirements

- Non-functional requirements may be very difficult to state precisely and imprecise requirements may be difficult to verify.
- Goal
  - A general intention of the user such as ease of use
- Verifiable non-functional requirement
  - A statement using some measure that can be objectively tested
- Goals are helpful to developers as they convey the intentions of the system users

Examples

- **A system goal**
  - The system should be easy to use by experienced controllers and should be organised in such a way that user errors are minimised.
- **A verifiable non-functional requirement**
  - Experienced controllers shall be able to use all the system functions after a total of two hours training. After this training, the average number of errors made by experienced users shall not exceed two per day.

Requirements Measures

Property	Measure
Speed	Processed transactions/second User/Event response time Screen refresh time
Size	K bytes Number of RAM 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
Portability	Percentage of target-dependent statements Number of target systems

Domain Requirements

- Derived from the application domain and describe system characteristics and features that reflect the domain.
- Domain requirements reflect the environment in which the system operates so.
- E.g. domains such as medical, e-commerce, banking etc.
- Requirements are specialized.
- Domain requirements are important because they often reflect fundamentals of the application domain.
- Sometimes, Characteristics of the application domain mean that the requirements specification has to include a description of how to carry out some computation. E.g. using some mathematical formula.
- Can be understood by means of “Ethnography”.

Domain Requirements Example

- For example, the requirements for the insulin pump system that delivers insulin on demand include the following domain requirement:
- The system safety shall be assured according to standard IEC 60601-1:Medical Electrical Equipment – Part 1:General Requirements for Basic Safety and Essential Performance.
- This requirement means that the developers must be familiar with that standard to ensure that they do not violate it. It constrains both the design of the device and the development process.

Domain Requirements Example

- For example, the domain requirement below is included in the requirements specification for an automated train protection system. This system automatically stops a train if it goes through a red signal. This requirement states how the train deceleration is computed by the system. It uses domain-specific terminology. To understand it, you need some understanding of the operation of railway systems and train characteristics.
- The deceleration of the train shall be computed as:
- D (train) = D (control) + D (gradient)
- Where D (gradient) is 9.81ms2 * compensated gradient/alpha **and** the values of 9.81ms2 /alpha are known for different types of train.
- The requirement for the train system illustrates a major problem with domain requirements. They are written in the language of the application domain (mathematical equations in this case) and it is often difficult for software engineers to understand them

Domain Requirements Example

- Library System Domain Requirements
  - There shall be a standard user interface to all databases which shall be based on the Z39.50 standard.
  - Because of copyright restrictions, some documents must be deleted immediately on arrival. Depending on the user's requirements, these documents will either be printed locally on the system server for manually forwarding to the user or routed to a network printer.

Domain Requirements Example

- Inventory System Domain Requirements
  - 1.1. Inventory System must have basic functions: storing, tracking, updating (data or things that we keep in the inventory), and must be able to generate reports.
  - 1.2. Every operation that occurs in the inventory system must concern of Data Integrity. For example, we do not want number of product in our inventory system to become -1, or enter the product without product name.
  - 1.3. Every operation that occurs in the inventory system must be recorded, and the system should generate report from time to time.
  - 1.4. Regarding security issue, the inventory system must have an authorization module to prevent unauthorized access.
  - 1.5. Authorized person must be able to access the System 24/7 except the system is under maintenance.
  - 1.6. Back up unit is required for unexpected system failure event.

Domain Requirements problems

- Understandability
  - Requirements are expressed in the language of the application domain
  - This is often not understood by software engineers developing the system
- Implicitness
  - Domain specialists understand the area so well that they do not think of making the domain requirements explicit

Problems with Natural Language

- Lack of clarity
  - Precision is difficult without making the document difficult to read
- Requirements confusion
  - Functional and non-functional requirements tend to be mixed-up
- Requirements amalgamation
  - Several different requirements may be expressed together

Problems with Natural Language

4.5 LIBSYS shall provide a financial accounting system that maintains records of all payments made by users of the system. System managers may configure this system so that regular users may receive discounted rates.

2.6 Grid facilities To assist in the positioning of entities on a diagram, the user may turn on a grid in either centimetres or inches, via an option on the control panel. Initially, the grid is off. The grid may be turned on and off at any time during an editing session and can be toggled between inches and centimetres at any time. A grid option will be provided on the reduce-to-fit view but the number of grid lines shown will be reduced to avoid filling the smaller diagram with grid lines.

Alternatives for NL Specification

Notation	Description
Structured natural language	This approach depends on defining standard forms or templates to express the requirements specification.
Design description languages	This approach uses a language like a programming language but with more abstract features to specify the requirements by defining an operational model of the system. This approach is not now widely used although it can be useful for interface specifications.
Graphical notations	A graphical language, supplemented by text annotations is used to define the functional requirements for the system. An early example of such a graphical language was SADT (Ross, 1977) (Schoman and Ross, 1977). Now, use-case descriptions (Jacobsen, et al., 1993) and sequence diagrams are commonly used (Stevens and Pooley, 1999).
Mathematical specifications	These are notations based on mathematical concepts such as finite-state machines or sets. These unambiguous specifications reduce the arguments between customer and contractor about system functionality. However, most customers don't understand formal specifications and are reluctant to accept it as a system contract.

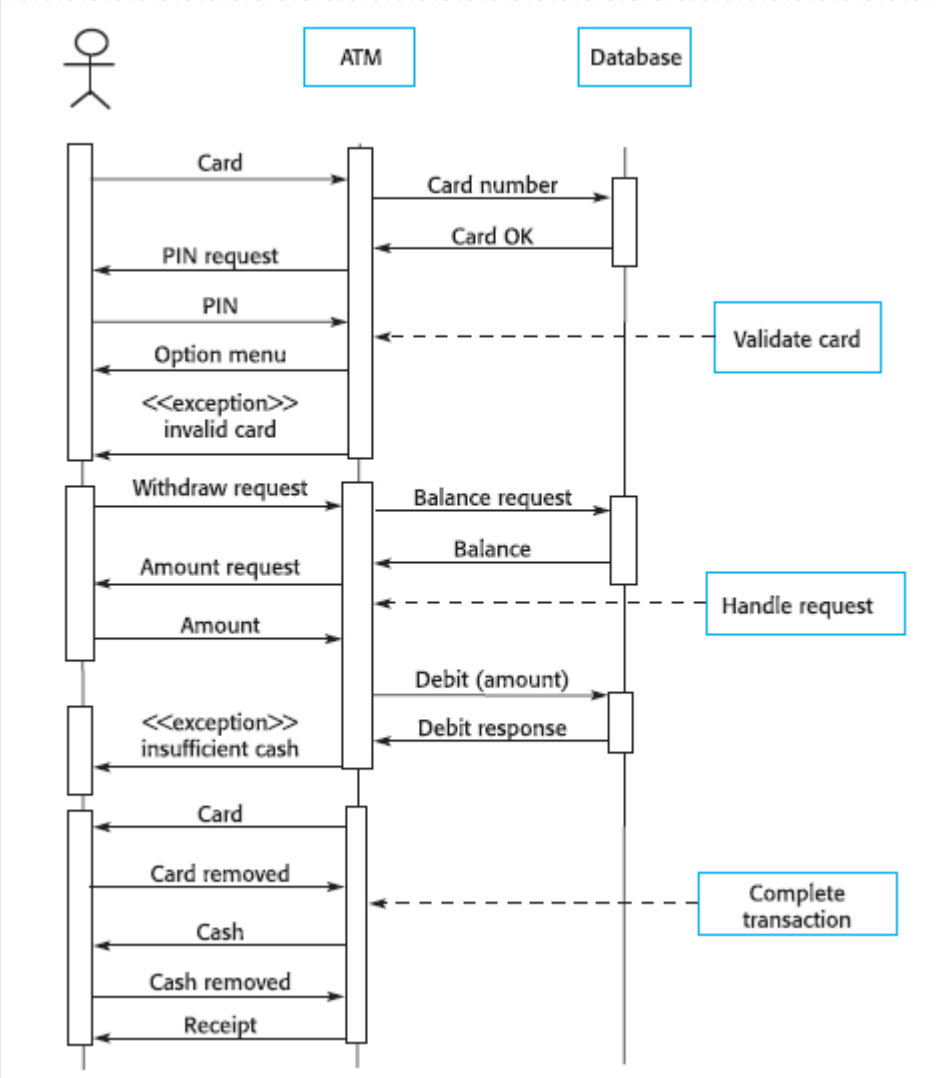
Using Structure Language Specification (form based approach)

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
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.	
Requires	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

Using tabular form & Mathematical notations

Condition	Action
Sugar level falling ($r2 < r1$)	CompDose = 0
Sugar level stable ($r2 = r1$)	CompDose = 0
Sugar level increasing and rate of increase decreasing ($(r2 - r1) < (r1 - r0)$)	CompDose = 0
Sugar level increasing and rate of increase stable or increasing. ($(r2 - r1) > (r1 - r0)$)	CompDose = round $((r2 - r1)/4)$ If rounded result = 0 then CompDose = MinimumDose

Using graphical notations



Software Requirement Specification

- Refer to the IEEE format given in book.

Thank You!!!

Any Questions???