IT3205: Fundamentals of Software Engineering (Compulsory)

BIT – 2nd Year

Semester 3

IT3205: Fundamentals of Software Engineering

Requirements Analysis & Specification

Duration: 8 hours

Learning Objectives

- Identify the types of requirements, which should be captured in a software project.
- Describe and apply different requirement analysis and specification techniques.
- Prepare a software requirement specification (SRS) for a given software problem.

Detailed Syllabus

3.1 System and software requirements

3.2 Types of software requirements
  3.2.1 Functional and non-functional requirements
  3.2.2 Domain requirements
  3.2.3 User requirements

Detailed Syllabus

3.3 Elicitation and analysis of requirements

- 3.3.1 Overview of techniques
- 3.3.2 Viewpoints
- 3.3.3 Interviewing
- 3.3.4 Scenarios
- 3.3.5 Use-cases
- 3.3.6 Ethnography

3.4 Requirements validation

3.5 Requirements specification

3.1 SYSTEM AND SOFTWARE REQUIREMENTS

What is a Requirement?

- Descriptions and specifications of a system
- 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

Requirements abstraction (Davis)

“If a company wishes to let a contract for a large software development project, it must define its needs in a sufficiently abstract way that a solution is not pre-defined. The requirements must be written so that several contractors can bid for the contract, offering, perhaps, different ways of meeting the client organization’s needs. Once a contract has been awarded, the contractor must write a system definition for the client in more detail so that the client understands and can validate what the software will do. Both of these documents may be called the requirements document for the system.”

System Requirements

- More detailed descriptions of system functions, services and constraints than user requirements.
- They are intended to be a basis for designing the system.
- They may be incorporated into the system contract.
- System requirements specification may include different models of the system.

  E.g. Object model, data-flow model

Requirements Verifiability

- Requirements should be written so that they can be objectively verified.
- see the following requirement statement;

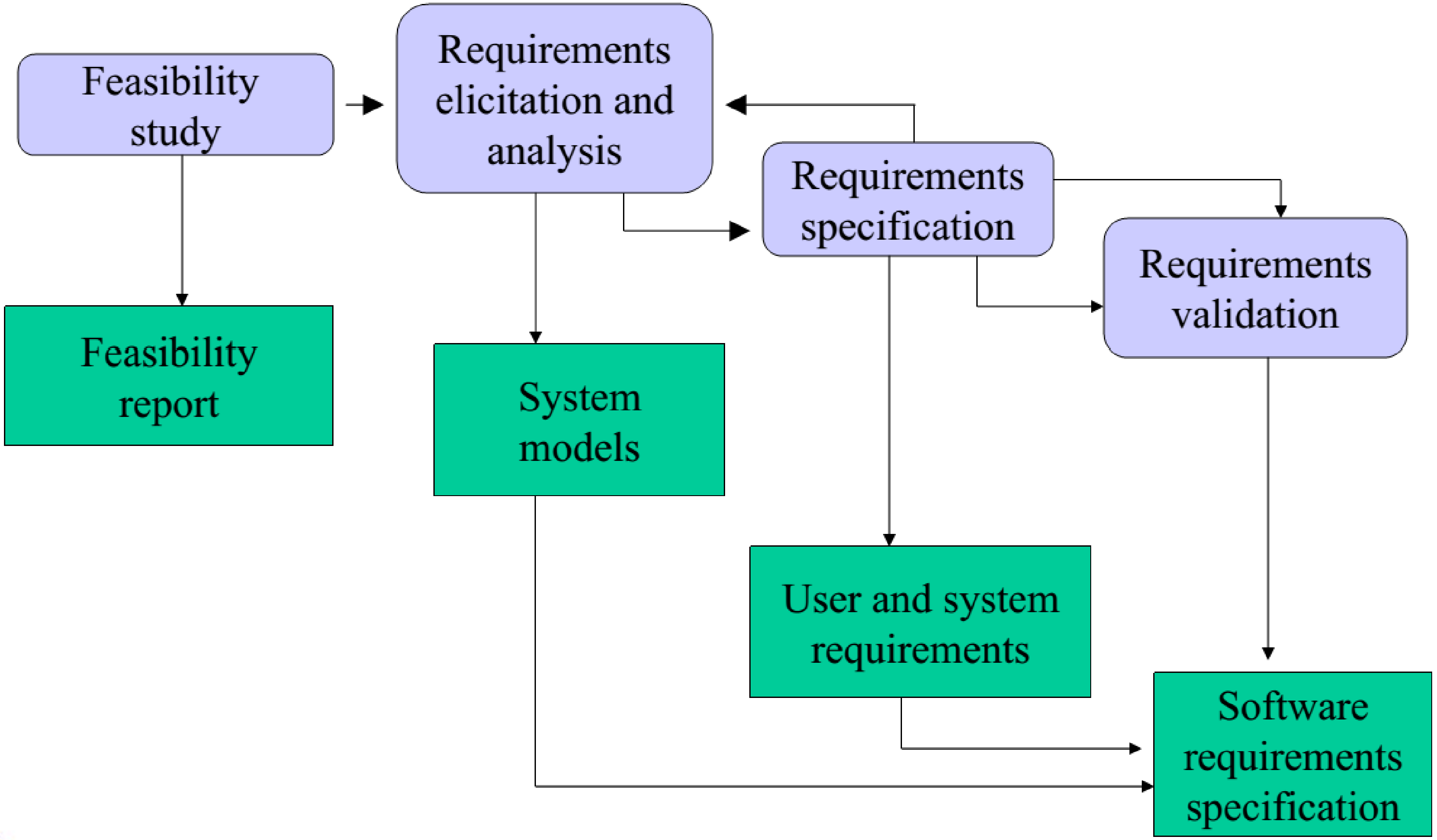
"Experienced controllers should be able to use all the system functions after a total of two hours training. After this training, the average number of errors made by an experienced users should not exceed two per day"

The problem with this requirement is its use of vague terms such as ' errors should be minimised'

Software Specification

- The process of establishing what services are required and the constraints on the system’s operation and development
- Also known as Requirements engineering
- Consists of four main phases
  1. Feasibility (technical and otherwise) study
  2. Requirements elicitation and analysis
  3. Requirements specification (documentation)
  4. Requirements validation

The Requirements Engineering Process



Feasibility study

Requirements elicitation and analysis

Requirements specification

Requirements validation

Feasibility report

System models

User and system requirements

Software requirements specification

The Requirements Engineering Process

- **Feasibility Study**
  - A feasibility study is a short, focused study which aims to check the feasibility of the project considering resources, cost/benefit, technology availability etc.
  - A feasibility study decides whether or not the proposed system is worthwhile.
  - A short focused study that checks;
    - If the system contributes to organizational objectives
    - If the system can be engineered using current technology and within budget
    - If the system can be integrated with other systems that are used

The Requirements Engineering Process

- **Feasibility Study**
  - Based on information assessment (what is required), information collection and report writing.
  - Questions for people in the organization
    - What if the system wasn't implemented?
    - What are current process problems?
    - How will the proposed system help?
    - What will be the integration problems?
    - Is new technology needed? What skills?
    - What facilities must be supported by the proposed system?

The Requirements Engineering Process

- **Requirement elicitation and analysis**
  - In this activity, technical software development staff work with customers and system end-users to find out about the application domain, what services the system should provide, the required performance of the system, hardware constraints and so on.

The Requirements Engineering Process

- **Requirements specification**
  - A detailed and precise description of the system requirements is set out to act as a basis for a contract between client and software developer.
- **Requirements validation**
  - In this activity, checks should be carried out to make sure that the requirements are accurate and complete. It is necessary to check the correctness of the specification of requirements.

Requirements Documents

User requirements (Requirements definition)

These are statements in a natural language plus diagrams, of what services the system is expected to provide and constraints under which it must operate

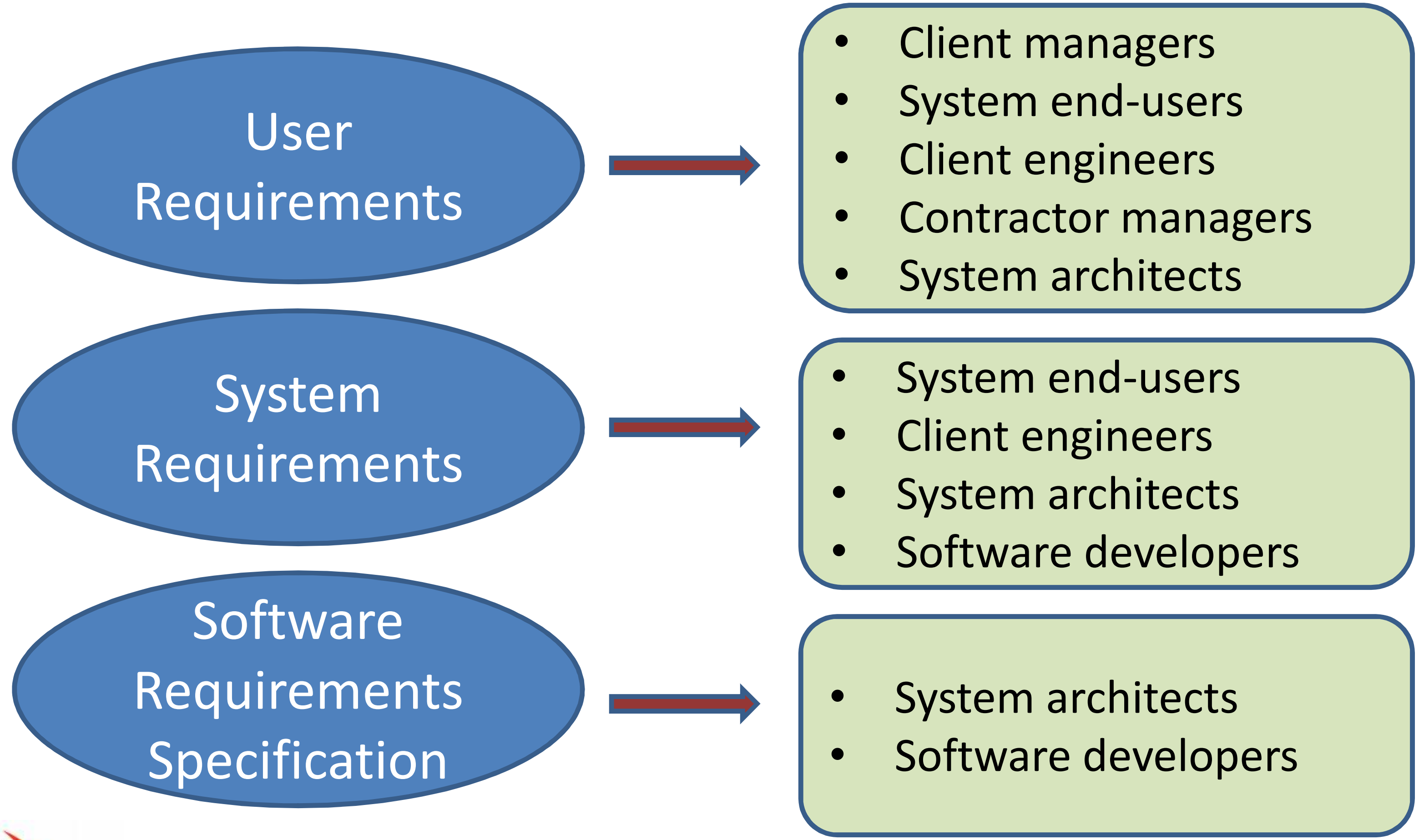
System requirements

The system services and constraints in detail. This is a detailed functional specification. It may serve as a contract between the system buyer and the software developer.

Software Requirements Specification (SRS)

A detailed and technical specification of the requirements. This is the basis for design and implementation. This is an abstract description of the software.

Readers of Requirements Documents



User Requirements

- Client managers
- System end-users
- Client engineers
- Contractor managers
- System architects

System Requirements

- System end-users
- Client engineers
- System architects
- Software developers

Software Requirements Specification

- System architects
- Software developers

The structure of a requirements document

Chapter	Description
Preface	This should define the expected readership of the document and describe its version history, including a rationale for the creation of a new version and a summary of the changes made in each version.
Introduction	This should describe the need for the system. It should briefly describe the system's functions and explain how it will work with other systems. It should also describe how the system fits into the overall business or strategic objectives of the organization commissioning the software.
Glossary	This should define the technical terms used in the document. You should not make assumptions about the experience or expertise of the reader.
User requirements definition	Here, you describe the services provided for the user. The nonfunctional system requirements should also be described in this section. This description may use natural language, diagrams, or other notations that are understandable to customers. Product and process standards that must be followed should be specified.
System architecture	This chapter should present a high-level overview of the anticipated system architecture, showing the distribution of functions across system modules. Architectural components that are reused should be highlighted.

The structure of a requirements document

Chapter	Description
System requirements specification	This should describe the functional and nonfunctional requirements in more detail. If necessary, further detail may also be added to the nonfunctional requirements. Interfaces to other systems may be defined.
System models	This might include graphical system models showing the relationships between the system components and the system and its environment. Examples of possible models are object models, data-flow models, or semantic data models.
System evolution	This should describe the fundamental assumptions on which the system is based, and any anticipated changes due to hardware evolution, changing user needs, and so on. This section is useful for system designers as it may help them avoid design decisions that would constrain likely future changes to the system.
Appendices	These should provide detailed, specific information that is related to the application being developed; for example, hardware and database descriptions. Hardware requirements define the minimal and optimal configurations for the system. Database requirements define the logical organization of the data used by the system and the relationships between data.
Index	Several indexes to the document may be included. As well as a normal alphabetic index, there may be an index of diagrams, an index of functions, and so on.

Software Requirements Document IEEE standard

1. **Introduction**
   - 1.1 Purpose of the requirements document
   - 1.2 Scope of the product
   - 1.3 Definitions, acronyms and abbreviations
   - 1.4 References
   - 1.5 Overview of the remainder of the document
2. **General Description**
   - 2.1 Product perspective
   - 2.2 Product functions
   - 2.3 User characteristics
   - 2.4 General constraints
   - 2.5 Assumptions and dependencies
3. **Specific requirements**
   - All functional and non-functional requirements
   - System models (eg. DFD, ERD, Use-Case, Class, Sequence diagrams)
   - External Interfaces, Performance, database requirements, design constraints
   - Security, quality characteristics
4. **Appendices**

3.2 TYPES OF SOFTWARE REQUIREMENTS

Type of Requirements

Functional Requirements

These are statements of services the system should provide, how the system should react to particular inputs and how the system should behave in particular situations.

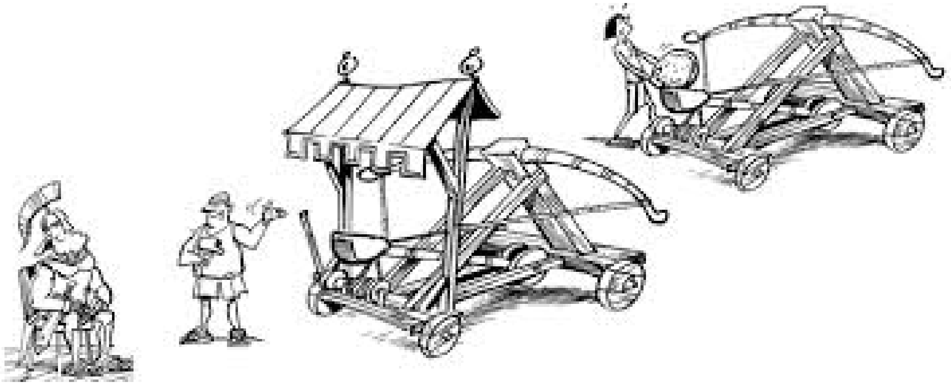
Non-functional Requirements

These are constraints on the services or functions offered by the system.

Domain Requirements

These are requirements that come from the application domain of the system and that reflect characteristics of the domain.

3.2.1 FUNCTIONAL AND NON-FUNCTIONAL REQUIREMENTS



Functional Requirements

- Describe functionality or services that the system is expected to provide.
- Depend on the type of software, expected users and the type of system where the software is used.
- Functional user requirements may be high-level statements of what the system should do but functional system requirements should describe the system services in detail.

Functional Requirements

Example:

Some functional requirements of a university library system

1. The user should be able to search for a library item by specifying a key word.
2. The library staff member should be able to issue a library item by scanning the bar codes of the library item and the student card.
3. Students can reserve a library item on-line.
4. The user shall be able to search either all of the initial set of databases or select a subset from it.
5. The system shall provide appropriate viewers for the user to read documents in the document store.
6. 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.

Functional Requirements - Issues

- Requirements Imprecision
  - Problems arise when requirements are not precisely stated.
  - Ambiguous requirements may be interpreted in different ways by developers and users.

    E.g. 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.*

Non-functional Requirements

- These define system properties and constraints
  - System Properties : reliability, response time and store occupancy.
  - Constraints : I/O device capability, data representations used in system interfaces.
- More critical than functional requirements. If these are not met, the system is useless.
- Relate to the system as a whole rather than to individual system features.

Non-functional Requirements

- Not always concerned with the software system to be developed.
- Constrain the process which may be used to develop the system.
- Non-functional requirements arise through user needs, because of;
  - Budget constraints
  - Organizational policies
  - The need for interoperability with other s / w and h / w systems
  - External factors e.g. safety regulations

Non-functional Requirements

- Non-functional requirements may be very difficult to state precisely and imprecise requirements may be difficult to verify.
- May be written to reflect the goals of the customer.
  - Goal
    - A general intention of the user such as ease of use.
    - Goals are helpful to developers as they convey the intentions of the system users.

Non-functional Requirements

- Verifiable non-functional requirement
  - A statement using some measure that can be objectively tested.

Example

A system goal

- The system should be easy to use by experienced controllers and should be organized in such a way that user errors are minimized.

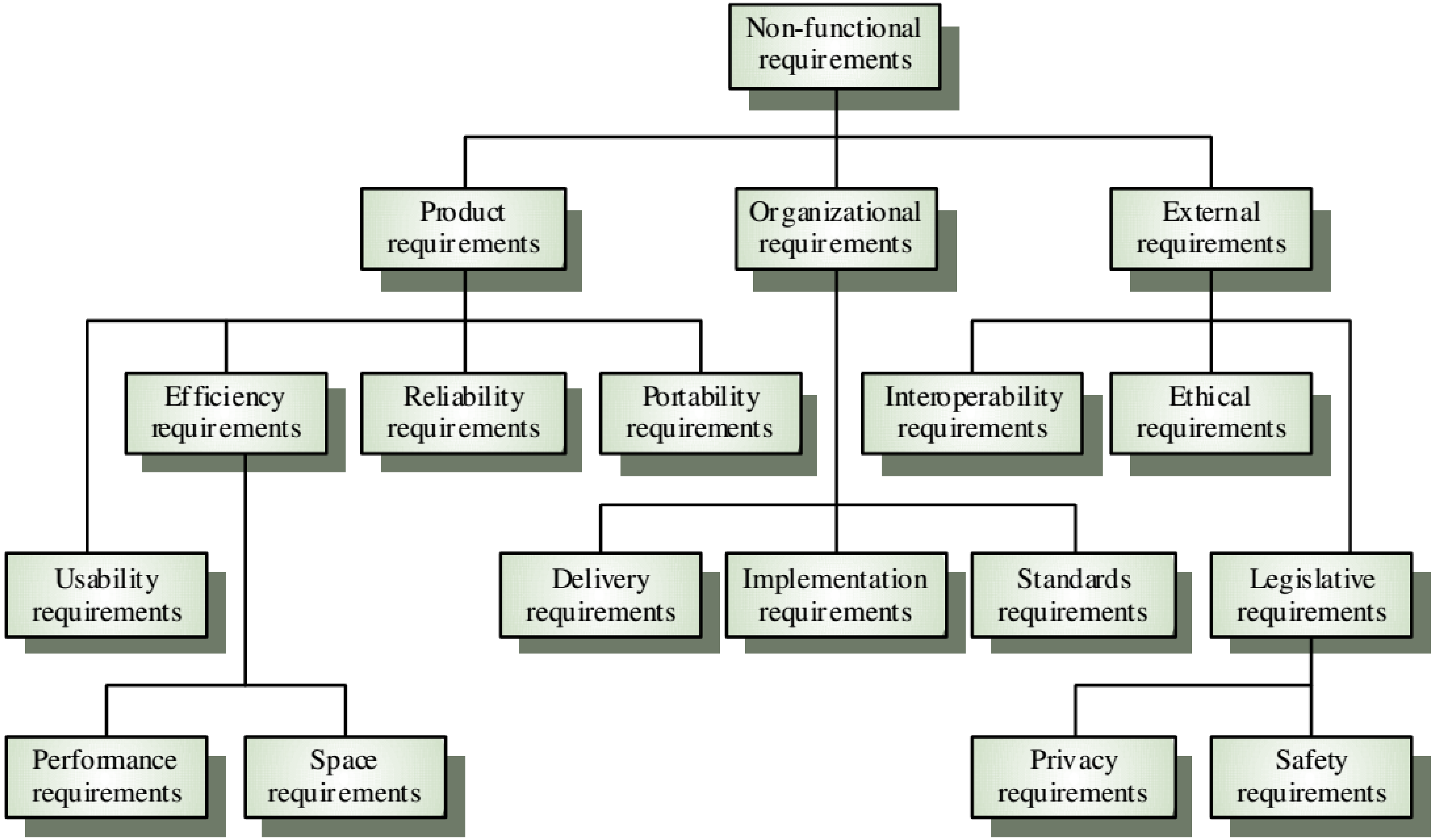
A verifiable non-functional requirement

- 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

Non-functional Requirements

- Requirements Interaction
  - Conflicts between different non-functional requirements are common in complex systems.
  - Spacecraft system
    - To minimise weight, the number of separate chips in the system should be minimised.
    - To minimise power consumption, lower power chips should be used.
    - However, using low power chips may mean that more chips have to be used. Which is the most critical requirement?

Non-functional Requirements - Classification



Non-functional Requirements - Classification

1. Product Requirements

- Requirements which specify that the delivered product must behave in a particular way.
  - Performance requirements
  - Reliability
  - Portability
  - Interface requirements

Example: The system should be easy to use by non experienced users and hence should provide a graphical user interface.

Non-functional Requirements - Classification

2. Organizational Requirements

- Requirements which are a consequence of organizational policies and procedures like process standards used and implementation requirements.
  - Delivery requirements
  - Implementation requirements
  - Standards requirements

Example: The system development process and deliverable documents shall confirm to the process and deliverables defined in ISO 9000.

Non-functional Requirements - Classification

3. External Requirements

- Requirements which arise from factors which are external to the system and its development process such as interoperability requirements, legislative requirements etc.
  - Interoperability requirements
  - Ethical requirements
  - safety requirements

Example: The system shall not disclose and personnel information about customers apart from their name and reference number to the operators of the system.

Non-functional 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 (MTTF) Probability of unavailability Mean time between failures (MTBF) Mean time to recover (MTTR)
Robustness	Percentage of events causing failure Probability of data corruption on failure
Portability	Percentage of target dependent statements

3.2.2 DOMAIN REQUIREMENTS

Domain Requirements

- The system’s operational domain imposes requirements on the system.
  - For example, a train control system has to take into account the braking characteristics in different weather conditions.
- Domain requirements can be new functional requirements, constraints on existing requirements or define specific computations.
- If domain requirements are not satisfied, the system may be unworkable.

Domain Requirements

Example – Train Protection System

The deceleration of the train shall be computed as:

Dtrain = Dcontrol + Dgradient

where Dgradient is 9.81ms2 * compensated gradient/alpha and where the values of 9.81ms2 /alpha are known for different types of train.

It is difficult for a non-specialist to understand the implications of this and how it interacts with other requirements.

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

3.2.3 USER REQUIREMENTS

© Andy Nortnik * www.ClipartOf.com/15804

User Requirements

- Should describe functional and non-functional requirements in such a way that they are understandable by system users who don’t have detailed technical knowledge.
- User requirements are defined using natural language, tables and diagrams as these can be understood by all users.
- Should not be defined using an implementational model.

3.3 ELICITATION AND ANALYSIS OF REQUIREMENTS

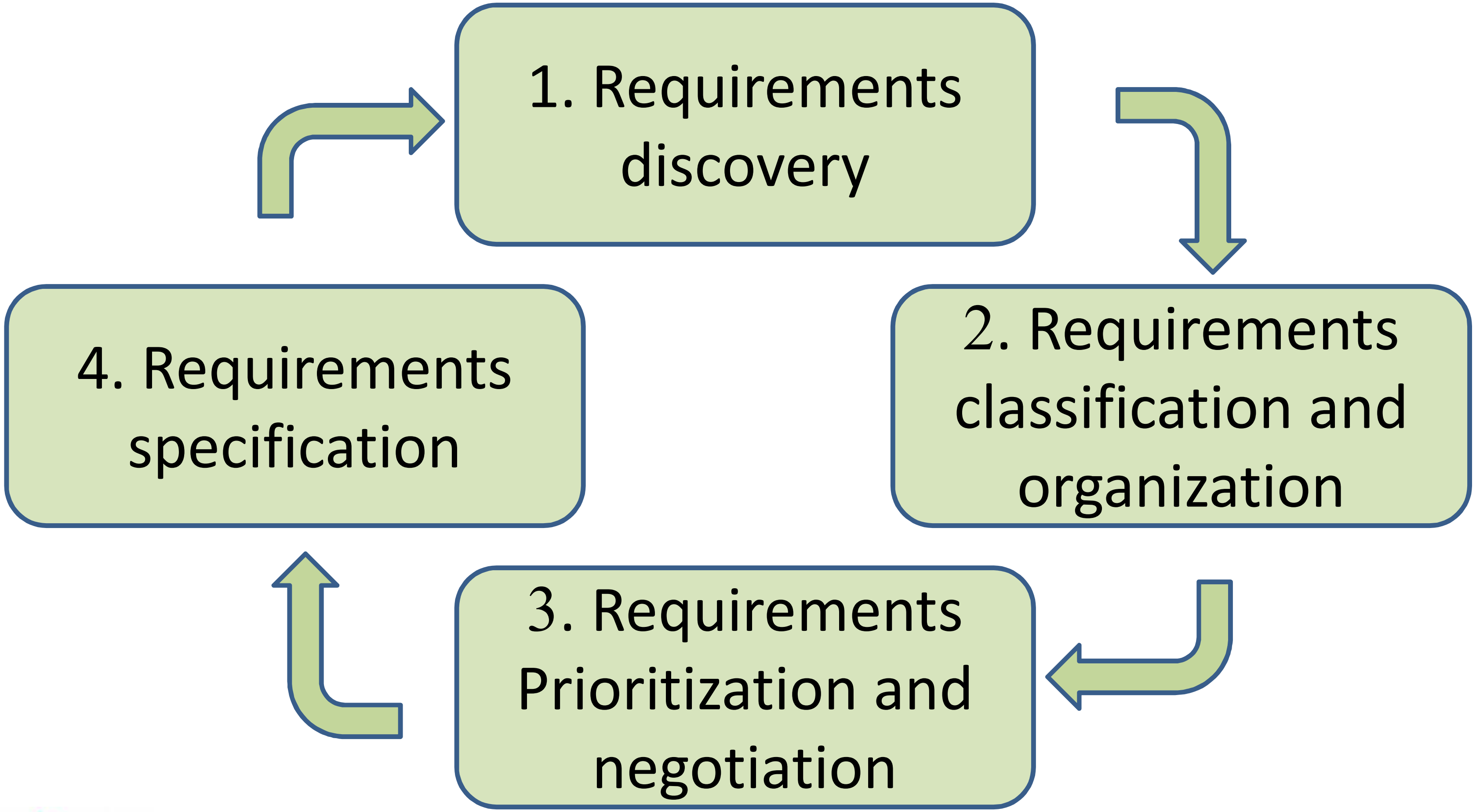
Requirements Elicitation and Analysis

- Sometimes called requirements elicitation or requirements discovery.
- Involves technical staff working with customers to find out about the application domain, the services that the system should provide and the system's operational constraints.
- May involve end-users, managers, engineers involved in maintenance, domain experts, trade unions, etc. They are called stakeholders.

Requirements Elicitation and Analysis

- Software engineers work with a range of system stakeholders to find out about the application domain, the services that the system should provide, the required system performance, hardware constraints, other systems, etc.
- Stages include:
  - Requirements discovery,
  - Requirements classification and organization,
  - Requirements prioritization and negotiation,
  - Requirements specification.

Requirements Elicitation and Analysis Process



1. Requirements discovery

2. Requirements classification and organization

3. Requirements Prioritization and negotiation

4. Requirements specification

Process Activities

1. Requirements discovery
   - Interacting with stakeholders to discover their requirements. Domain requirements are also discovered at this stage.
2. Requirements classification and organization
   - Groups related requirements and organizes them into coherent clusters.
3. Prioritization and negotiation
   - Prioritizing requirements and resolving requirements conflicts.
4. Requirements specification
   - Requirements are documented and input into the next round of the spiral.

Problems of Requirements Elicitation

- Stakeholders don’t know what they really want.
- Stakeholders express requirements in their own terms.
- Different stakeholders may have conflicting requirements.
- Organizational and political factors may influence the system requirements.
- The requirements change during the analysis process. New stakeholders may emerge and the business environment change.

Requirement Analysis Tools

- Traditional structured methods
  - Entity relationship diagrams (ERD)
  - Data Flow Diagrams (DFDs)
  - Entity State Transition diagrams
- Object Oriented methods (O-O)
  - Use Case Diagrams
  - Class Diagrams
  - Sequence Diagrams
  - State Transition Diagrams

Viewpoints

- A viewpoint is way of collecting and organizing a set of requirements from a group of stakeholders who have something in common.
- Each viewpoint therefore includes a set of system requirements.
- Viewpoints might come from end-users, managers, etc.
- They help identify the people who can provide information about their requirements and structure the requirements for analysis.

Interviewing

- Formal or informal interviews with stakeholders are part of most RE processes.
- Types of interview
  - Closed interviews based on pre-determined list of questions
  - Open interviews where various issues are explored with stakeholders.
- Effective interviewing
  - Be open-minded, avoid pre-conceived ideas about the requirements and are willing to listen to stakeholders.
  - Prompt the interviewee to get discussions going using a springboard question, a requirements proposal, or by working together on a prototype system.

Interviews in practice

- Normally a mix of closed and open-ended interviewing.
- Interviews are good for getting an overall understanding of what stakeholders do and how they might interact with the system.
- Interviews are not good for understanding domain requirements
  - Requirements engineers cannot understand specific domain terminology;
  - Some domain knowledge is so familiar that people find it hard to articulate or think that it isn't worth articulating.

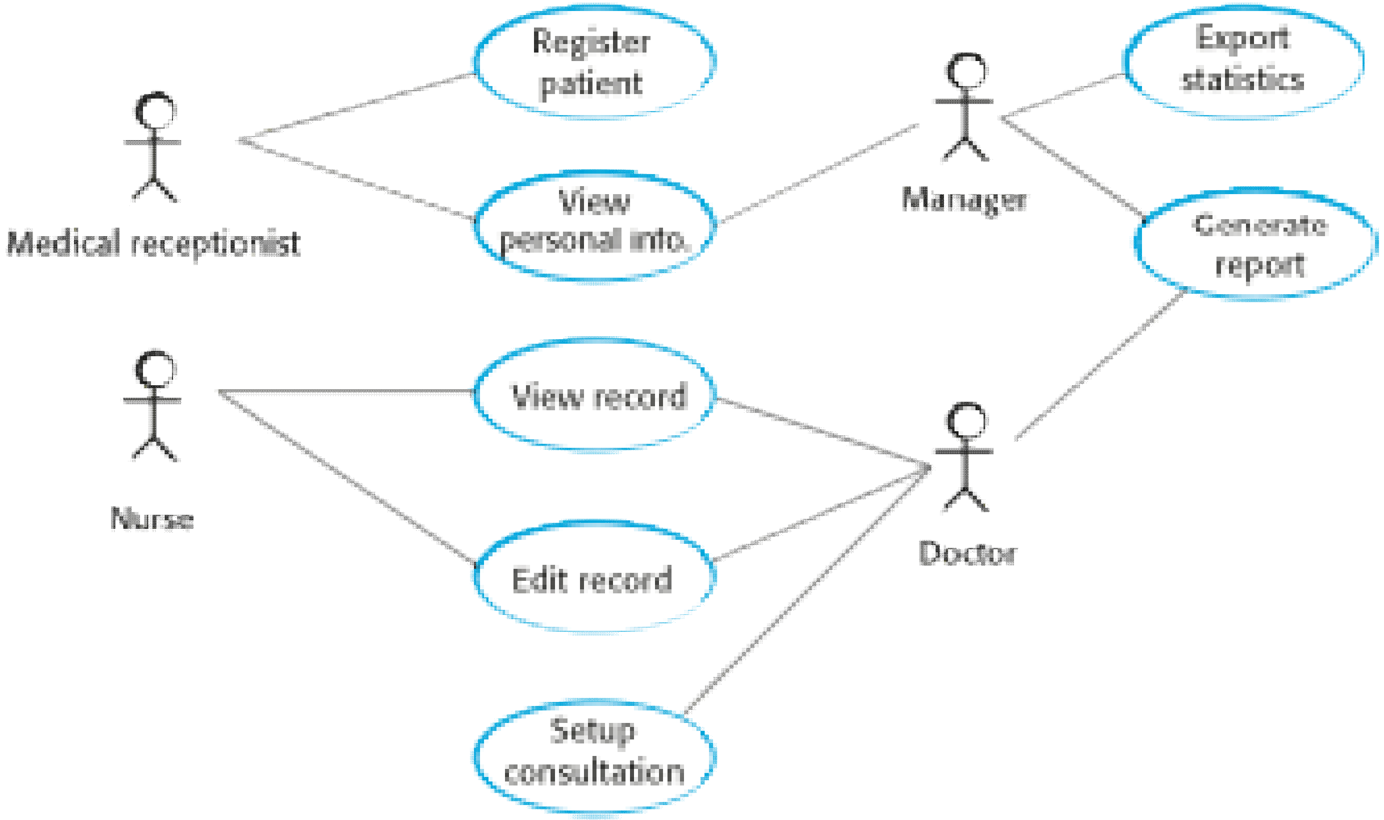
Scenarios

- Scenarios are real-life examples of how a system can be used.
- They should include
  - A description of the starting situation;
  - A description of the normal flow of events;
  - A description of what can go wrong;
  - Information about other concurrent activities;
  - A description of the state when the scenario finishes.

Use cases

- Use-cases are a scenario based technique in the UML which identify the actors in an interaction and which describe the interaction itself.
- A set of use cases should describe all possible interactions with the system.
- High-level graphical model supplemented by more detailed tabular description.
- Sequence diagrams may be used to add detail to use-cases by showing the sequence of event processing in the system.

Use cases for the MHC-PMS



Ethnography

- A social scientist spends a considerable time observing and analysing how people actually work.
- People do not have to explain or articulate their work.
- Social and organisational factors of importance may be observed.
- Ethnographic studies have shown that work is usually richer and more complex than suggested by simple system models.

Scope of Ethnography

- Requirements that are derived from the way that people actually work rather than the way I which process definitions suggest that they ought to work.
- Requirements that are derived from cooperation and awareness of other people's activities.
  - Awareness of what other people are doing leads to changes in the ways in which we do things.
- Ethnography is effective for understanding existing processes but cannot identify new features that should be added to a system.

Focused Ethnography

- Developed in a project studying the air traffic control process
- Combines ethnography with prototyping
- Prototype development results in unanswered questions which focus the ethnographic analysis.
- The problem with ethnography is that it studies existing practices which may have some historical basis which is no longer relevant.

3.4 REQUIREMENTS VALIDATION

Requirements Validation

- Concerned with demonstrating that the requirements define the system that the customer really wants.
- Requirements error costs are high so validation is very important
  - Fixing a requirements error after delivery may cost up to 100 times the cost of fixing an implementation error.

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 validation 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 Reviews

- Regular reviews should be held while the requirements definition is being formulated.
- Both client and contractor staff should be involved in reviews.
- Reviews may be formal (with completed documents) or informal. Good communications between developers, customers and users can resolve problems at an early stage.

Requirements Reviews

- Review Checks;
  - Verifiability
    - Is the requirement realistically testable?
  - Comprehensibility
    - Is the requirement properly understood?
  - Traceability
    - Is the origin of the requirement clearly stated?
  - Adaptability
    - Can the requirement be changed without a large impact on other requirements?

3.5 REQUIREMENTS SPECIFICATION

Requirements Specification

- The process of writing the user and system requirements in a requirements document.
- 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.
- The requirements may be part of a contract for the system development
  - It is therefore important that these are as complete as possible.

Ways of writing a system requirements specification

Notation	Description
Natural language	The requirements are written using numbered sentences in natural language. Each sentence should express one requirement.
Structured natural language	The requirements are written in natural language on a standard form or template. Each field provides information about an aspect of the requirement.
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 now rarely used although it can be useful for interface specifications.
Graphical notations	Graphical models, supplemented by text annotations, are used to define the functional requirements for the system; UML use case and sequence diagrams are commonly used.
Mathematical specifications	These notations are based on mathematical concepts such as finite-state machines or sets. Although these unambiguous specifications can reduce the ambiguity in a requirements document, most customers don’t understand a formal specification. They cannot check that it represents what they want and are reluctant to accept it as a system contract

Requirements and Design

- In principle, requirements should state what the system should do and the design should describe how it does this.
- In practice, requirements and design are inseparable
  - A system architecture may be designed to structure the requirements;
  - The system may inter-operate with other systems that generate design requirements;
  - The use of a specific architecture to satisfy non-functional requirements may be a domain requirement.
  - This may be the consequence of a regulatory requirement.

Natural Language Specification

- Requirements are written as natural language sentences supplemented by diagrams and tables.
- Used for writing requirements because it is expressive, intuitive and universal. This means that the requirements can be understood by users and customers.

Guidelines for Writing Requirements

- Invent a standard format and use it for all requirements.
- Use language in a consistent way. Use shall for mandatory requirements, should for desirable requirements.
- Use text highlighting to identify key parts of the requirement.
- Avoid the use of computer jargon.
- Include an explanation (rationale) of why a requirement is necessary.

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.

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.)*

Alternatives for Natural Language

1. Pseudo code
    - Requirements may be defined using a language like a programming language but with more flexibility of expression. Such a language is called pseudo code.
2. Graphical languages
3. Mathematical notation

Alternatives for Natural Language

- Most appropriate in two situations
  1. Where an operation is specified as a sequence of actions and the order is important
  2. When hardware and software interfaces have to be specified
- Disadvantages are :
  1. These may not be sufficiently expressive to define domain concepts
  2. The specification will be taken as a design rather than a specification

Structured Specifications

- An approach to writing requirements where the freedom of the requirements writer is limited and requirements are written in a standard way.
- This works well for some types of requirements e.g. requirements for embedded control system but is sometimes too rigid for writing business system requirements.

Form-based Specifications

- Definition of the function or entity.
- Description of inputs and where they come from.
- Description of outputs and where they go to.
- Information about the information needed for the computation and other entities used.
- Description of the action to be taken.
- Pre and post conditions (if appropriate).
- The side effects (if any) of the function.

Tabular Specification

- Used to supplement natural language.
- Particularly useful when you have to define a number of possible alternative courses of action.
- For example, the insulin pump systems bases its computations on the rate of change of blood sugar level and the tabular specification explains how to calculate the insulin requirement for different scenarios.

Summary

- 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.
- They often relate to the emergent properties of the system and therefore apply to the system as a whole.

Summary

- 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.
- The requirements engineering process is an iterative process including requirements elicitation, specification and validation.
- You can use a range of techniques for requirements elicitation including interviews, scenarios, use-cases and ethnography.
- Requirements validation is the process of checking the requirements for validity, consistency, completeness, realism and verifiability.