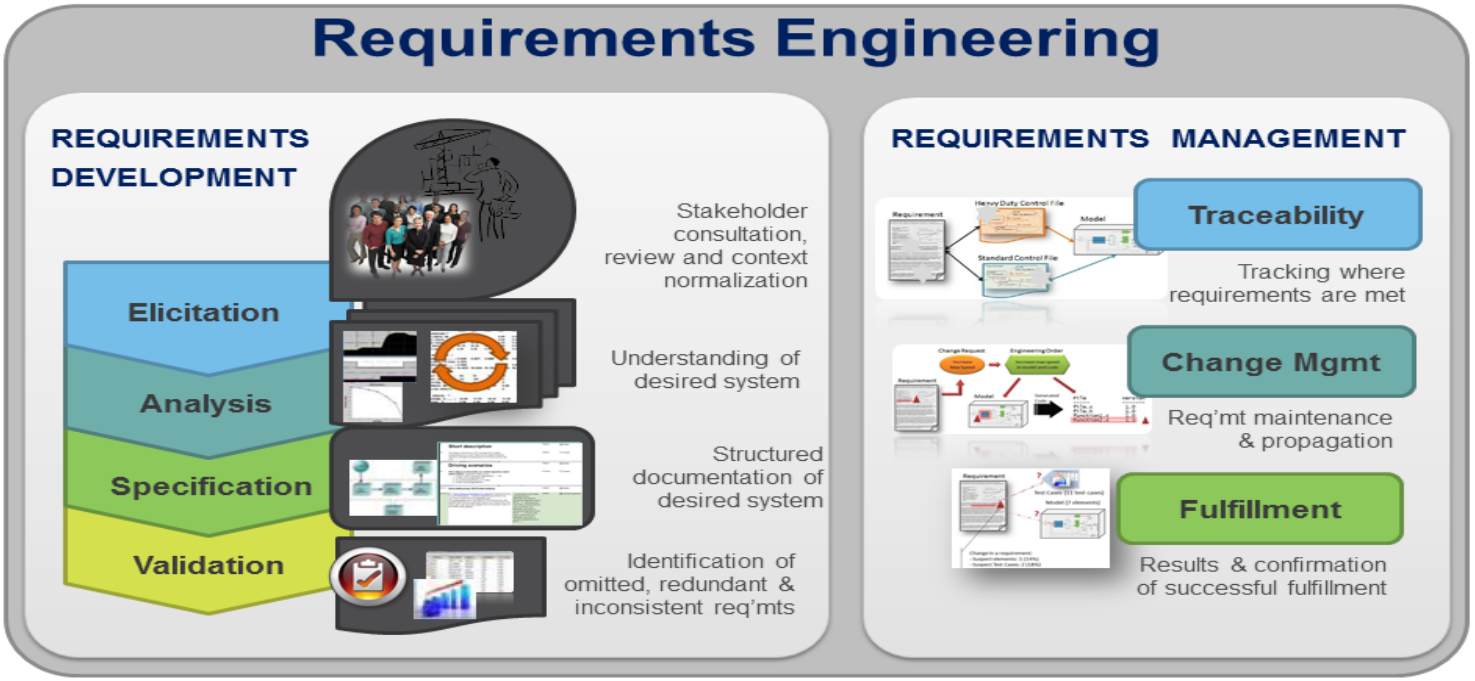


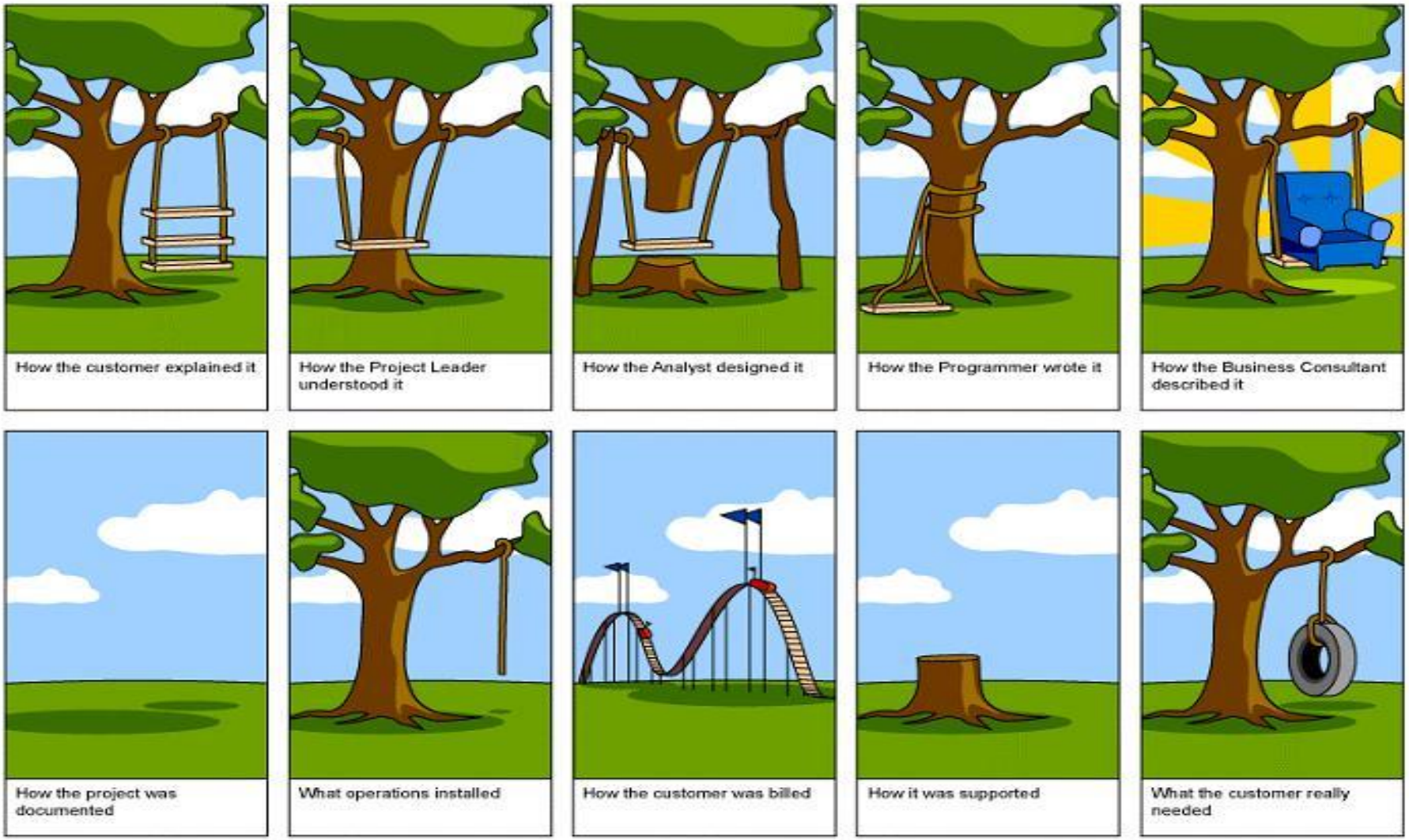
Lecture 3: Requirements Analysis

Software Requirements

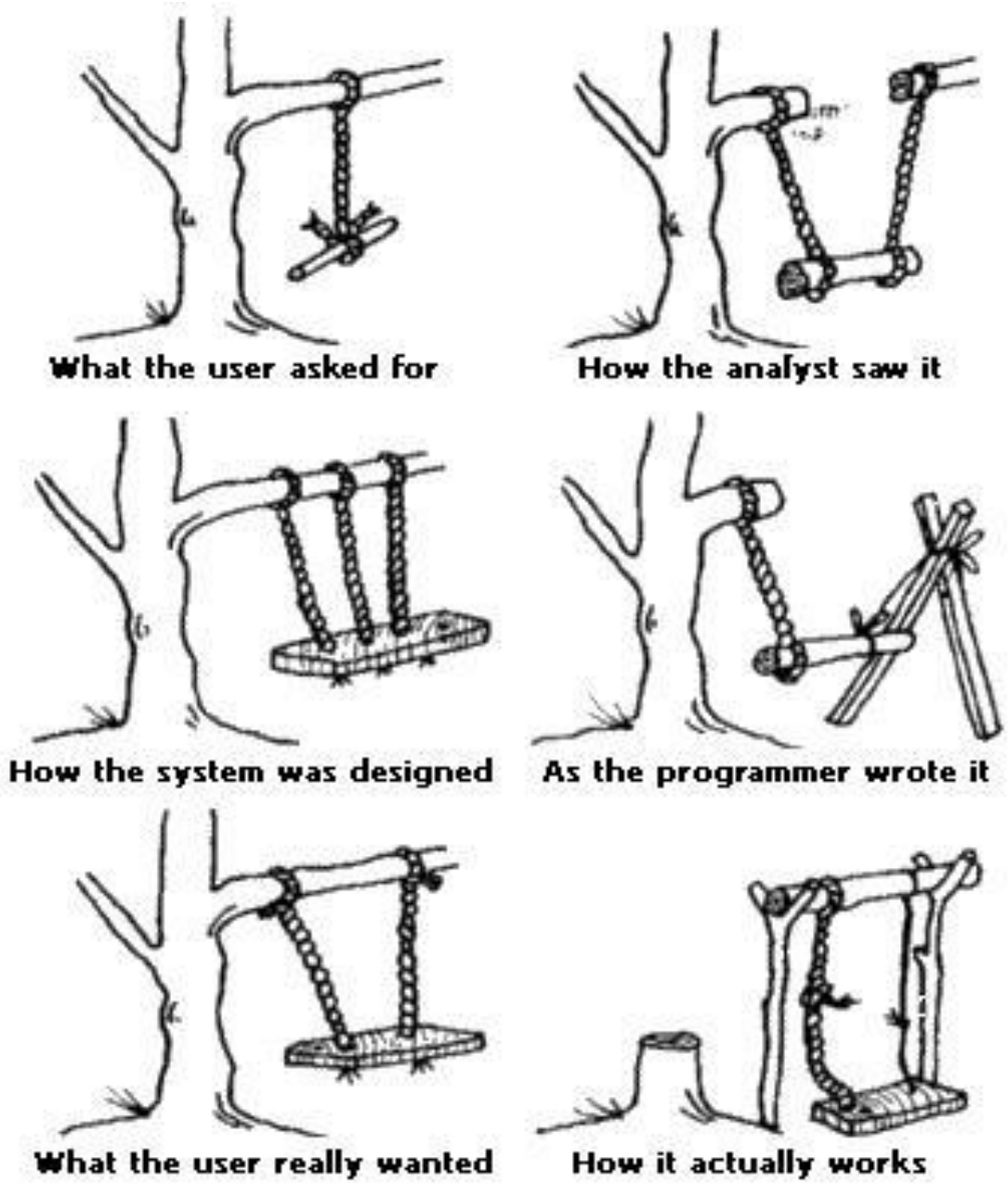
Objectives

- To introduce the concepts of **requirements**
- To describe **functional / non-functional requirements**
- To explain **techniques** for describing requirements
- To explain **how software requirements may be organised** in a software requirements document

Challenges faced by Software Engineers



Challenges faced by Software Engineers



Introduction

In engineering a **requirement** is a singular documented need of what a particular product or service should be or perform.

It is a statement that identifies a necessary **attribute, capability, characteristic**, or **quality** of a system in order for it **to have value and utility to a user.**

The sets of requirements are used as inputs into**:**

1. the design stages of product development.
2. the verification process, since tests should trace back to specific requirements.

Requirements show what elements and functions are necessary for the particular project.

The notion of a requirement

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”

Definition of “Requirement” [IEEE]:

- (1) a condition or capability needed by a user to solve a problem or achieve an objective
- (2) 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 documents

Types of requirement

- Requirements must be presented in alternative, but consistent forms that are understandable to different audiences. May range from a high-level abstract statement of a service or of a system constraint to a detailed mathematical functional specification.
    1. **User requirements**: Statements in natural language, diagrams of the services the system provides and its operational constraints that are written for customers
    2. **System requirements:** A structured document setting out detailed descriptions of the system's services and constraints. This document is usually written as a contract between the client and the contractor.

Example

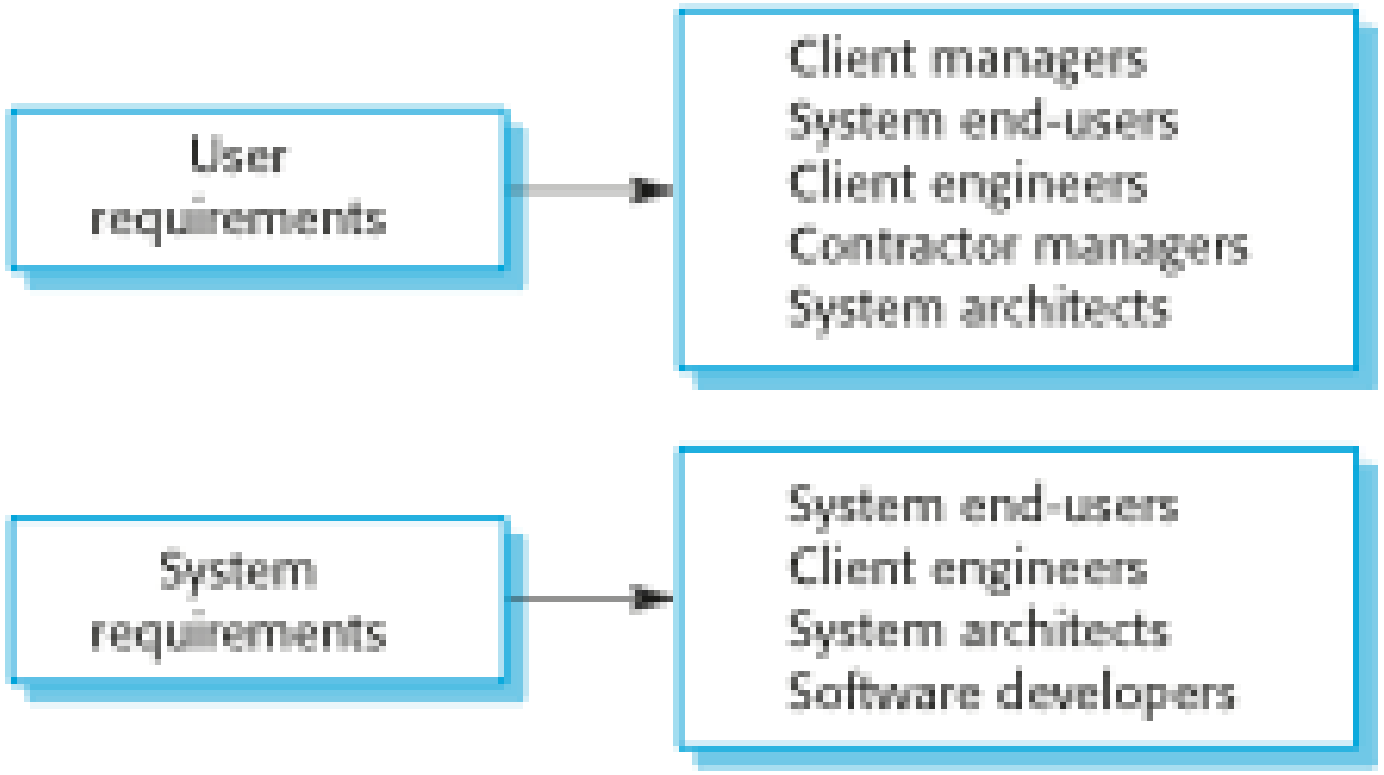
User requirement definition

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.

Readers of different types of requirements specification



Requirements Classification

User Requirements should describe functional and non-functional requirements so that they are understandable by system users who do not have detailed technical knowledge

Functional requirements:

- A functional requirement is a description of ***what* a system must do**.
- It is written down as statements of services that the software should provide, how the software should react to particular inputs and how the software should behave in particular situations.

Non-functional requirements:

- Non functional requirement specifies something about the software itself, and **how well it performs its functions**.
- **Constraints** on the services or functions offered by the software, such as, timing constraints, constraints on the development process, standards, and so on.

Domain requirements

- Constraints on the system from the domain of operation

Functional Requirements

- Describe functionality or system services.
- 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.
- Functional system requirements should describe the system services in detail.

- Functional requirements are statements that
  - Describe the functionality that the system should do
  - Describe the system function in detail, its inputs, outputs and exceptions
- Examples:
  - 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

Example Scenario for shared calendar

"The user types in all the names of the meeting participants together with some constraints such as the length of the meeting, roughly when the meeting needs to take place, and possibly where it needs to take place. The system then checks against the individuals' calendars and the central departmental calendar and presents the user with a series of dates on which everyone is free all at the same time. Then the meeting could be confirmed and written into people's calendars. Some people, though, will want to be asked before the calendar entry is made. Perhaps the system could email them automatically and ask that it be confirmed before it is written in."

Example Scenario for shared calendar

- **Functional requirements:**
  - The system shall allow the user to enter the details of a meeting
    - The system should validate the date of the meeting
  - The system shall allow the user to enter the participants of a meeting
  - The system should schedule the meeting at a time where all participants will be able to attend
    - The system should check each individual calendars to determine common free slots
    - The system should email the proposed time to all participants before confirmation
  - …

Example Scenario: *Navigation System for the Visually Impaired Persons*

- FR1: The system shall provide information about obstacles in the surrounding.
- FR2: The system shall provide information about buildings in the surrounding.
- FR3: The system shall allow the user to add a voice note at his/her current location.
- FR4: The system shall be able to guide the user towards his/her destination.
- FR5: The system shall be able to download maps.
- FR6: The system shall allow the maps to be customized.
- FR7: The system shall use voice to inform the user about nearby obstacles and buildings.
- FR8: The system shall use haptic feedback to get the user's attention for obstacles before any voice out.
- FR9: The system shall voice out context awareness information every thirty minutes.
- FR10: In case of low battery life, the system shall automatically switch to an emergency mode.

ACTIVITY

1. Write down five (5) functional requirements for an ATM.
2. Write down five (5) functional requirements for a car navigation system.

Non-functional requirements

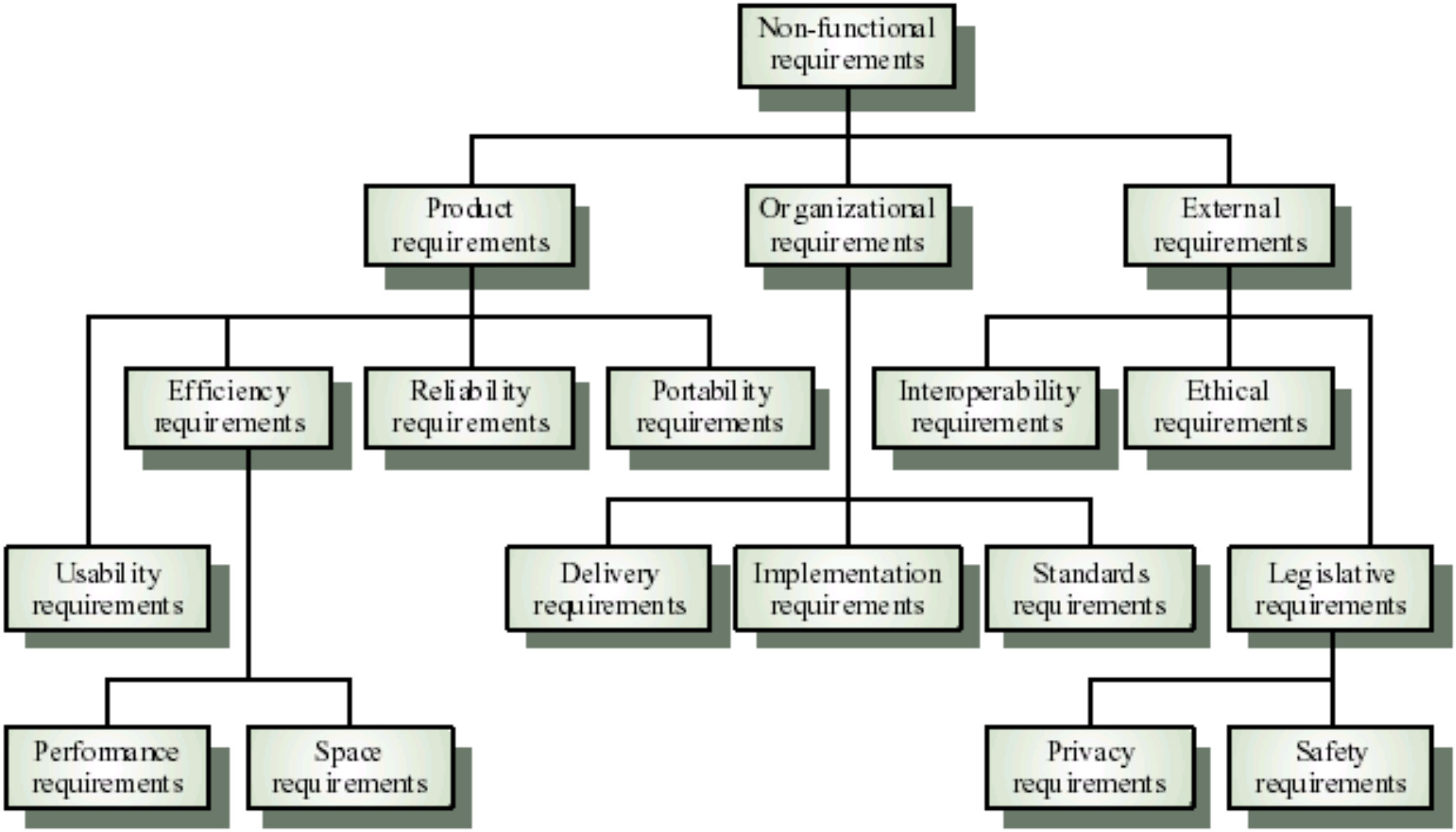
- These define system properties and constraints e.g. reliability, response time and storage requirements. Constraints are I/O device capability, system representations, etc.
- Process requirements may also be specified mandating a particular IDE, programming language or development method.
- Non-functional requirements may be more critical than functional requirements. If these are not met, the system may be useless.

Non-Functional Requirements

Non functional requirements are those requirements that act to constrain the solution.

- Quality attribute requirements (Dependability, Performance, Accuracy, Security)
- Project Constraints (Maximum cost, Maximum schedule)
- Design directives(sorting shall be done using Quicksort algorithm)
- Implementation directives (Software shall be written in C)

NFR Classification



Non-functional classifications

- 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 Requirements

a) **Design constraints:** include the fact that the software must run using a certain database system or that the software must fit into the memory of a 1GB machine.

b) **Performance requirements:** for example 95% of catalogue database queries shall be completed within 2 seconds.

c) **Security:** Access to student records shall be password-protected so that only authorized people can view them.

d) **Reliability requirements:** for instance, the acceptable failure rate or the amount of time that the system will operate properly without any crash encountered.

e) **Usability requirements:** The interface shall be menu-driven, so that it is easy to learn and use.

Non-Functional Requirements

f) Design directives: They tell the developer how to design certain functionality of the software. For example, "*The sorting of the student records shall be done using Quicksort algorithm*".

g) Implementation directives: They tell the developer what to use to implement certain functionality of the software. For example, "*The Software shall be written in java such that it can be portable onto different operating system platform*".

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.

Usability requirements

- The system should be easy to use by medical staff and should be organized in such a way that user errors are minimized. (Goal)
- 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. (Testable non-functional requirement)

Metrics for specifying non-functional requirements

In non-functional requirement, values are specified in order to be able to verify the requirement.

Property	Measure
Speed	Processed transactions/second User/event response time Screen refresh time
Size	Mbytes Number of ROM 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

Example Scenario for shared calendar

- **Non-Functional requirements:**
  - The system should determine the common free slots of participants for a meeting in less one minute.
    - The complexity of the algorithm should be less than $O(n^2)$
  - The system should be compatible with google calendar.

Example Scenario: *Navigation System for the Visually Impaired Persons*

- NFR1: The system shall not interfere with other devices or equipment used by the visually impaired or blind persons.
- NFR2: The system should provide simple interaction mechanisms of not more that three steps for the user to use.
- NFR3: The system should be able to guide the user towards his/her destination along the safest path.
- NFR4: The system should use a combination of Dijkstra's Algorithm and graph based for generating the safest path.
- NFR5: The system should be able to notify the user when an obstacle is at a distance, depending on the speed at which the user travels.

ACTIVITY

1. Write down five (5) non-functional requirements for an ATM.
2. Write down five (5) non-functional requirements for a car navigation system.

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 be new functional requirements, constraints on existing requirements or define specific computations.
- If domain requirements are not satisfied, the system may be unworkable.

Train protection system

- This is a domain requirement for a 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.

Characteristics of good requirement

Unitary (Cohesive)	The requirement addresses one and only one thing.
Complete	The requirement is fully stated in one place with no missing information.
Consistent	The requirement does not contradict any other requirement
Non-Conjugated (Atomic)	The requirement is atomic, i.e., it does not contain conjunctions. E.g., "The postal code field must validate American and Canadian postal codes" should be written as two separate requirements: (1) "The postal code field must validate American postal codes" and (2) "The postal code field must validate Canadian postal codes".

Characteristics of good requirement

Traceable	The requirement meets all or part of a business need as stated by stakeholders and authoritatively documented.
Feasible	The requirement can be implemented within the constraints of the project.
Unambiguous	The requirement is concisely stated without recourse to technical jargon, acronyms (unless defined elsewhere in the Requirements document).
Mandatory	The requirement represents a stakeholder-defined characteristic the absence of which will result in a deficiency that cannot be ameliorated.
Verifiable	The implementation of the requirement can be determined through methods like: inspection, demonstration and testing.

Guidelines for writing requirements(1)

- Invent a standard format for numbering the requirement. The simplest approach is to give every requirement a unique sequence number, such as UR-2 or SRS13. The most commonly used convention is hierarchical numbering. If the functional requirements appear in section 3.2 of your requirement document, they will all have labels that start by 3.2, for example, 3.2.1 "*The system shall compute the salary of employee based on the time-sheet*".

- Use language consistently; for example, define mandatory requirement using 'shall' and desirable requirement using 'should'.

Guidelines for writing requirements(2)

- Avoid, as far as possible, the use of computer jargon to describe requirement. However, it will be inevitable when providing technical details.
- To reduce ambiguity, avoid vague, subjective terms such as “user-friendly”, “easy”, “simple”, “rapid”, “efficient”, “support”, “several”, “superior”, “acceptable”, and “robust”. Find out what the customers really mean when they say “user-friendly” or “fast” or “robust” and state those expectations in the requirements.
- Avoid comparative words such as “improve”, “maximize”, “minimize”, and “optimize”. Quantify the degree of improvement that is needed, or state the maximum and minimum acceptable values of some parameters.

Quality measures related to individual requirement statements

Imperatives: Words and phrases that command the presence of some feature, function, or deliverable. They are listed below in decreasing order of strength.	
Shall	Used to dictate the provision of a functional capability.
Must or must not	Most often used to establish performance requirement or constraints.
Is required to	Used as an imperative in SRS statements when written in passive voice.
Are applicable	Used to include, by reference, standards, or other documentation as an addition to the requirement being specified.
Responsible for	Used as an imperative in SRSs that are written for systems with pre-defined architectures.
Will	Used to cite things that the operational or development environment is to provide to the capability being specified. For example, The vehicle's exhaust system will power the ABC widget.
Should	Not used often as an imperative in SRS statements; however, when used, the SRS statement always reads weak. Avoid using Should in your SRSs.

Requirements Analysis (1)

- Analyse requirements in order to identify & Resolve:
  - **Ambiguities :** different interpretations of natural language

    E.g: If the central computer receives an alarm signal from the local process-control computer, it will shut down.
  - **Platitudes**: Vague FRs and non-measurable NFRs which leads to untestable/untraceable requirements

    E.g: "System shall be user-friendly", "System shall be reliable"
  - **Omissions:** Essential information left out

    E.g: "The LEVEL command shall display the average water level in the reservoir between two times typed in by the user."

Requirements Analysis (2)

- **Inconsistencies:** Contradictory statements

  "After validating input lists, only alphabetically sorted lists remain."

  Later on: "If a validated list is not in alphabetical order, an error message shall be displayed"

- **Mixtures of abstraction :** Different levels of detail in the same section

  "The PAYROLL subsystem performs all the functions related to staff payment. The OVERTIME module computes the overtime payment for the employee currently being processed based on the time-sheet for the current week.

- **FRs and NFRs mixed up**

  The OVERTIME subsystem shall be written in Java and computes the overtime for an employee based on his time-sheet.

ACTIVITY
Requirements with deficiencies

1. Identify the deficiencies in the following requirements
   - *"The product shall provide status messages at regular intervals not less than every 60 seconds."*
   - *"The product shall switch between displaying and hiding nonprinting characters instantaneously."*
   - *"The parser shall produce an HTML markup error report that allows quick resolution of errors when used by HTML novices."*
   - *"Charge numbers should be validated online against the master corporate charge number list, if possible."*
   - *"The product shall not offer search and replace options that could have disastrous results."*

Problems with natural language (1)

- **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(2)

- **Ambiguity**
  - The readers and writers of the requirement must interpret the same words in the same way. NL is naturally ambiguous so this is very difficult
- **Over-flexibility**
  - The same thing may be said in a number of different ways in the specification
- **Lack of modularisation**
  - NL structures are inadequate to structure system requirements

Alternative to Natural Language

- ***Structured language specification*:** This approach depends on defining standard forms or templates to express the requirements specification. More info

- ***Program description language*:** It uses a language like programming language but with more abstract features to specify the requirement by defining an operational model of the system.

- ***Graphical notations*:** A graphical language, supplemented by text annotations is used to define the functional requirements for the system. The most common graphical representation is Use-Case

- ***Formal specifications:*** Requirements are specified, by using mathematically precise formal logic languages or notations based on mathematical concepts such as finite-state machines or sets.

Software Requirements Specification (SRS)

- Software Requirements Specification (SRS):
  - It is the official statement of what is required of the system developers. The requirements are recorded in one or more form including natural language, formal, symbolic or graphical representations.

- **The primary purposes of documenting requirements in only one document is to ensure that:**
  - Customers and developers have the same understanding of what is to be built.
  - All developers have identical understanding of what is to be built.
  - Testers are testing for the same qualities that developers are building.
  - Management is applying resources to the same set of tasks that the developers are performing.

Requirements Specifications

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

   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 files type may be presented as a specific icon on the user's display

- The specification add details to the requirements definition and should be consistent with the requirements definition. Usually presented with system models

Goals of SRS

- A well-designed, well-written SRS accomplishes four major goals:
- **It provides feedback to the customer.** An SRS is the customer's assurance that the development organization understands the issues or problems to be solved and the software behavior necessary to address those problems. Therefore, the SRS should be written in natural language (versus a formal language), in an unambiguous manner that may also include charts, tables, data flow diagrams, decision tables, and so on.
- **It decomposes the problem into component parts.** The simple act of writing down software requirements in a well-designed format organizes information, places borders around the problem, solidifies ideas, and helps break down the problem into its component parts in an orderly fashion.
- **It serves as an input to the design specification.** As mentioned previously, the SRS serves as the parent document to subsequent documents, such as the software design specification and statement of work. Therefore, the SRS must contain sufficient detail in the functional system requirements so that a design solution can be devised. It serves as a product validation check.
- **It serves as the parent document** for testing and validation strategies that will be applied to the requirements for verification.

Audiences for SRS(1)

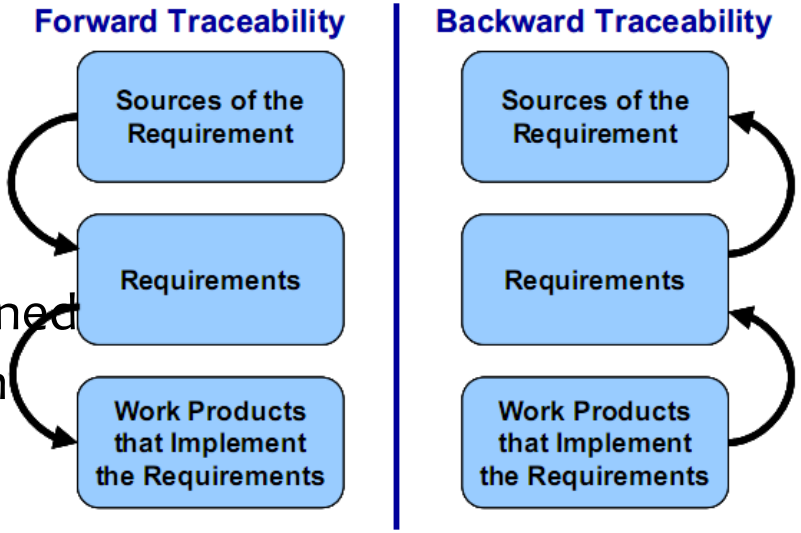
- Customers and the marketing department rely on the SRS to know what product they can expect to be delivered.
- Project managers base their plans and estimates of schedule, effort, and resources on the product description contained in the SRS.
- The software development team relies on the SRS to understand what is to be built.
- The testing group uses the product behavior descriptions contained in the SRS to derive test plans, cases, and procedures.

Audiences for SRS(2)

- Software maintenance and support staff refer to the SRS to understand what each part of the product is supposed to do.
- The publications group writes user documentation, such as manuals and help screens, based on both the SRS and the user interface design.
- Training personnel can use both the SRS and the user documentation to help them develop educational materials.

Characteristics of Document

- **Correct :** Free from errors
- **Unambiguous, precise & clear :** One and only one interpretation for every requirement
- **Complete :** All requirements for the problem identified & specified
- **Testable :** By quantitative means
- **Consistent :** No conflicting requirements
- **Modifiable:** This attribute is more concerned with the format & style of the SRS rather than the actual requirements themselves
- **Traceable :** Backward & Forward traceability



Guidelines for SRS Document (1)

- An example organization of an SRS is as follows:
- Introduction
  - Purpose
  - Definitions
  - System overview
  - References
- Overall description
  - Product perspective
    - System Interfaces
    - User Interfaces
    - Hardware interfaces
    - Software interfaces
    - Communication Interfaces
    - Memory Constraints
    - Operations
    - Site Adaptation Requirements
  - Product functions
  - User characteristics
  - Constraints, assumptions and dependencies

Guidelines for SRS Document (2)

- Specific requirements
  - External interface requirements
  - Functional requirements
  - Performance requirements
  - Design constraints
    - Standards Compliance
  - Logical database requirement
  - Software System attributes
    - Reliability
    - Availability
    - Security
    - Maintainability
    - Portability
  - Other requirements

Guidelines for SRS Document (3)

Introduction	
Purpose of document	*Intended audience, how is the SRS document to be used?*
Scope	*What are the boundaries of the specified system?*
Definitions & notations:	*High-level definitions used throughout document e.g. Acct. No., TRX, Ref. No.*
References	*Sources of information concerning the current system.*
General Description	
Product perspective	*Context diagram, relationship between system & its environment*
Product functions	*Brief description of the primary functions*
Characteristics of eventual users	*Required background or training*
Constraints	*Hardware limitations, Implementation & language requirements*

Guidelines for SRS Document (4)

System Architecture	*Present a high-level overview of the anticipated system architecture showing the distribution of functions across system modules*
Specific Requirements	*Functional Requirements: Can be expressed by using natural language, formal, symbolic or graphical representations* *Non functional Requirements: Can be expressed using natural language, formal, symbolic or graphical representations.*
Appendices	*Provide detailed, specific information which is related to the application which is being developed. (Hardware descriptions)*
Index	*Page numbers of important section and sub-sections*

References

- "Software Engineering", 9th Edition, by Ian Sommerville, Addison-Wesley. (Chapter 4 Requirements Engineering)