Requirements Engineering

Based on the book Software Engineering, Ivan Sommerville, Chapter 4

Outline

- User requirements and system requirements
- Functional and non-functional requirements
- Requirement engineering activities
  - Feasibility study
  - Analysis and Elicitation
  - Specification
  - Validation
- The software requirements document
- Example/Sample of SRS

What is Requirements engineering ?

- The process of establishing the services that the customer requires from a system and the constraints under which the system operates and is developed.

- The ***requirements*** themselves are the descriptions of the system services and constraints on its operation that are generated during the requirements engineering process.

Requirements

- Requirements define the expected services of the system (service statement) constitute **functional requirements**
- Requirements define constraints the system must obey are **non functional requirements**
- A requirement may range from a high-level abstract statement of a service or of a system constraint (from user point of view or **user requirements**) to a detailed definition of a system function (how the system functions, **system requirements**).

User requirements example

- Example: user requirement- a high abstract level requirement of Mental Healthcare Patient Management system

R1. “The system shall generate monthly management reports showing the cost of drugs prescribed by each clinic during that month”

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 or diagrams** as these can be understood by all users.

System requirements

- More **detailed specifications** of system functions, services and constraints than user requirements (e.g. scenarios, actions, activities.)
- They are intended to be a **basis for designing** the system.
- They may be incorporated into the system **contract**.
- System requirements may be defined or illustrated using **system models**.

System requirements example

- System requirements (More details of the User requirement R1)
  - R1.1 On the last working day of each month, a summary of the drugs prescribed, their cost, and the prescribing clinics shall be generated.
  - R1.2 The system shall automatically generate the report for printing after 17.30 on the last working day of the month
  - R1.3 If drugs are available in different units, separate reports shall be created for each dose unit.
  - R1.4 Access to all cost reports shall be restricted to authorized users listed on a management access control list

Requirement levels and requirements types

- Different detail level of requirements address to different readers (customer, customer manager, system developer, system end-user, etc.)
- 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
- Two types of requirements: functional and non-functional requirements

Functional Requirements

- These are statements of services, tasks or functions the system should provide.
- A functional requirement is usually a “shall statement”

Example:

1. A user shall be able to search the appointments lists for all clinics
2. The system shall generate each day, for each clinic, a list of patients who are expected to attend appointments that day.

Question

- What functional requirements are there for an ATM machine?

ATM functional requirements

Example:

The system shall allow to withdraw money

The system shall be able to validate user cards

ATM functional requirements

- **R1.**The system shall be able to validate user cards
  - R1.1 The system shall ask user to input their PIN
  - R1.2 The system shall validate the PIN
- **R2.**The system shall allow to withdraw money
  - R2.1. The system shall ask an amount of money
  - **R2.2.** The system shall validate the amount is available in the user's account before releasing funds to the user
  - **R2.3.** The system shall debit the user's account upon withdrawal of funds
  - **R2.4**. The system shall be able to issue a specific amount of money to the user.

Requirements imprecision

- Problems arise when requirements are not precisely stated.
- Ambiguous requirements may be interpreted in different ways by developers and users.
- Consider the term 'search' in requirement 1
  - User intention – search for a patient name across all appointments in all clinics;
  - Developer interpretation – search for a patient name in an individual clinic. User chooses clinic then search.

Requirements 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.
- In practice, it is difficult to produce a complete and consistent requirements document.

Non Functional requirements

- These are constraints on the services or functions offered by the system such as timing constraints, “look and feel”, performance, security, standards, etc.

- Describe aspects of the system that are not directly related to the functional behaviour of the system.

Categories of non functional requirements

- Usability: define the ease of use of system. A system is more usable if it is easy to use (documentation, help facilities, training, etc.)
- Reliability: the frequency of system failure, correct output and function, system recovery from failure, availability of the system during its operational hours, dependable system
- Performance: the response time of the system, efficient use of resources
- Supportability: adaptability, maintainability

Additional non functional requirements

- Implementation requirements
  - Constraint on the implementation of the system, such as use of particular programming languages, hardware platform
- Legal requirements
  - Concerned with licensing, regulation, and certification issues.

FURPS+ Requirements means
Functionality, Usability, reliability, Performance, Supportability, and more

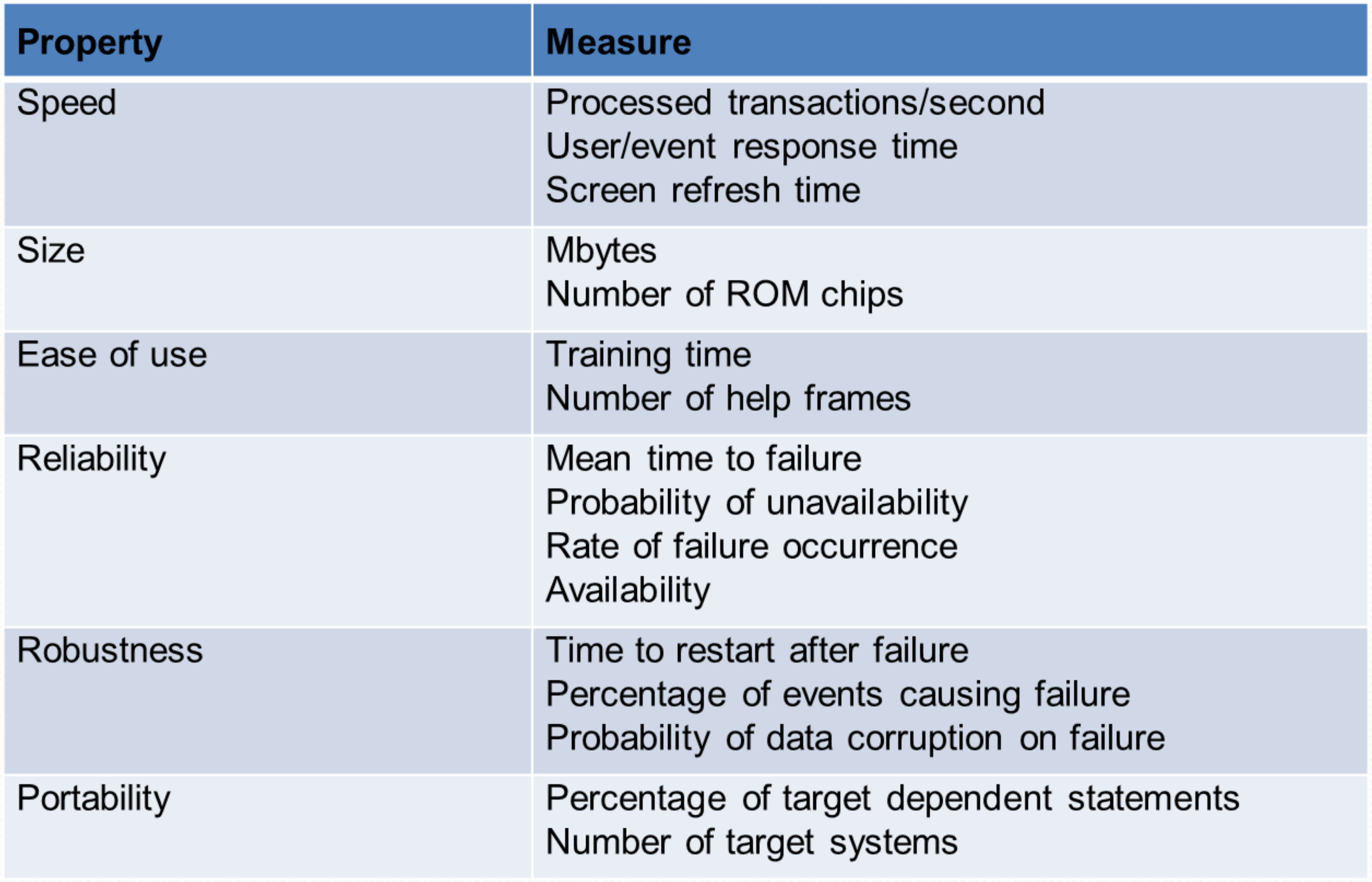
Example

- Non functional of Mental Healthcare Patient Management System
  - Availability:
    - The MHC-PMS shall be available to all clinics during normal working hours (Mon–Fri, 08.30–17.30). Downtime within normal working hours shall not exceed five seconds in any one day.
  - Legal requirements:
    - The system shall implement patient privacy provisions as set out in HStan-03-2006-priv

Non-functional requirements implementation

- Non-functional requirements may affect the overall architecture of a system rather than the individual components.
  - For example, to ensure that performance requirements are met, the system may be organized to minimize communications between components.
- A single non-functional requirement, such as a security requirement, may generate a number of related functional requirements that define system services that are required.
  - For example create the authentication function

Metrics for measuring non-functional requirements



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

Requirements Engineering Process

Requirements engineering

- Aims at defining the requirements of the system under construction.
- Four activities:
  - Feasibility study
  - Requirements elicitation and analysis
  - Requirements specification
  - Requirements validation

Requirements engineering process

Requirements specification

Requirements elicitation

Requirements validation

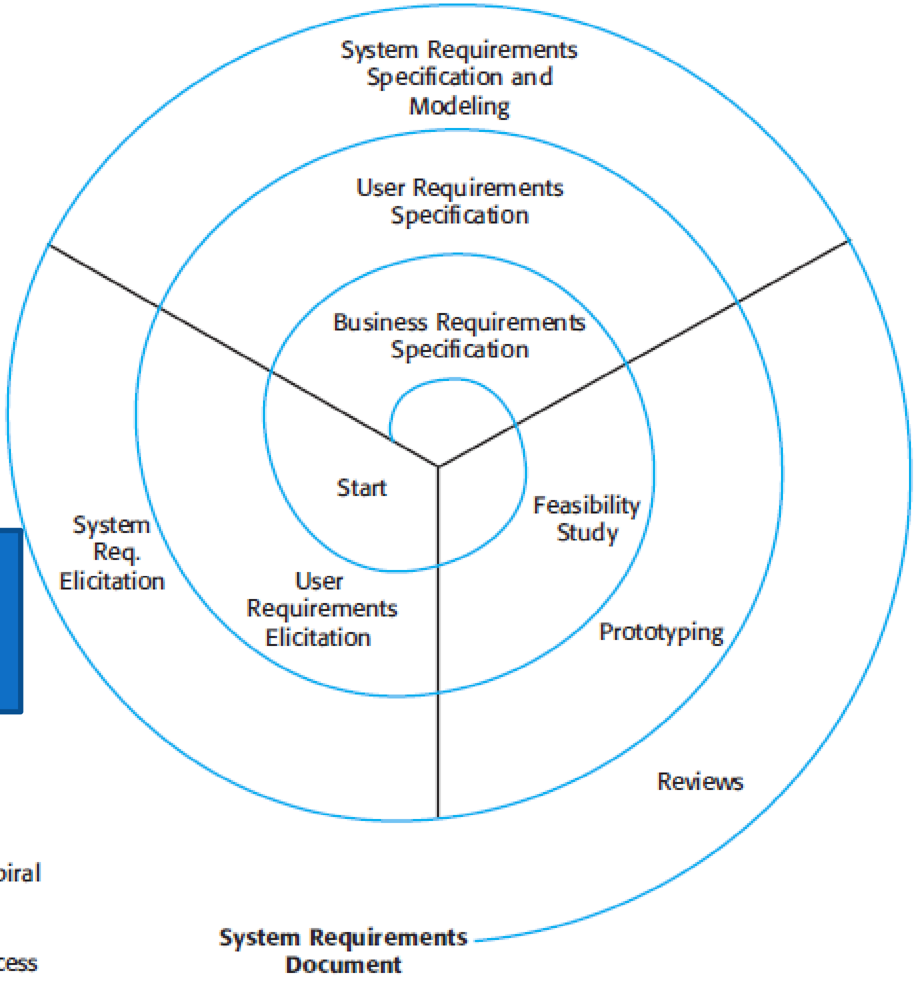

System Requirements Specification and Modeling
User Requirements Specification
Business Requirements Specification
Start
Feasibility Study
System Req. Elicitation
User Requirements Elicitation
Prototyping
Reviews
System Requirements Document

Figure 4.12 A spiral view of the requirements engineering process

Feasibility studies

- A feasibility study decides whether or not the proposed system is useful to the business.
- A short focused study that checks
  - If the system contributes to organisational 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.

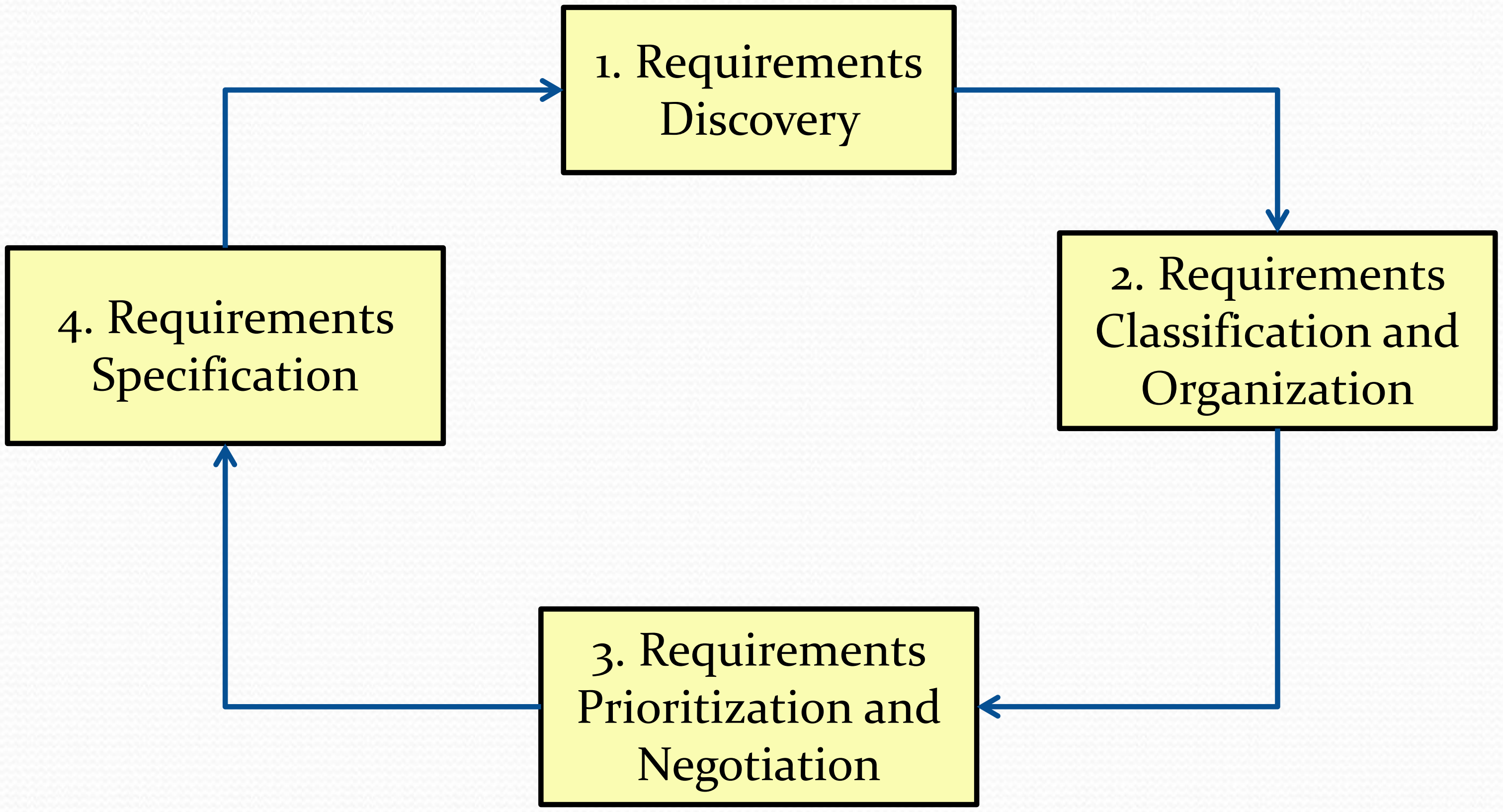
Feasibility study (cont.)

- Based on information assessment (what is required), information collection and report writing.
- Questions for people in the organisation
  - 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?

Elicitation and analysis

- It is about elicitation and 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. These are called *stakeholders*.

Elicitation Process


1. Requirements Discovery
2. Requirements Classification and Organization
3. Requirements Prioritization and Negotiation
4. Requirements Specification

Elicitation process

- Requirement discovery
  - Working with stakeholders to discover their requirements
- Requirement classification and organisation
  - Grouping related requirements into cluster (the basis for sub systems identification)
- Requirement prioritization and negotiation
  - Prioritize and resolve conflict requirements
- Requirement specification
  - Requirements are documented

Requirements discovery

- The process of gathering information about the proposed and existing systems and distilling the user and system requirements from this information.
- Sources of information include documentation, system stakeholders and the specifications of similar systems.

Traditional methods of requirement discovery

- Interviewing customers and domain experts
- Questionnaires
- Observation
- Study of business documents and existing software systems

Modern methods of requirement elicitation

- Prototyping
- Brainstorming
- Rapid application development

Prototyping

- Prototypes are useful for requirements elicitation because:
  - The customer may be more likely to view the prototype and react to it than to read the SRS and react to it. Thus, the prototype provides quick feedback.
  - The prototype displays unanticipated aspects of the systems behaviour. Thus, it produces not only answers but also new questions. This helps reach closure on the SRS.
  - An SRS based on a prototype tends to undergo less change during development, thus shortening development time.

(IEEE, SRS)

Brainstorming

- "A conference technique to form new ideas or find a solution to a specific problem by putting aside judgment, social inhibitions and rules"
- It is useful when there is difficulty of reaching consensus of stakeholders on requirements

Requirements specification

- Requirements documenting, converting to standard formats
- Invent/reuse a standard format and use it for all requirements.
- Use language in a consistent way. Use <u>shall</u> for mandatory requirements, <u>should</u> for desirable requirements.
- Use text highlighting to identify key parts of the requirement.
- Avoid the use of computer jargon.

Issues with using a natural language System Specification

- 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 specification

- Structured natural language:
  - Filling standard forms with predefined fields
- Graphical notations
  - E.g. UML Use cases diagram, sequence diagram, class diagram, activity diagram, state transition diagram
- Formal specification
  - Mathematical notations

Use cases for requirement specification

- Use case model illustrates the system's intended functions (use cases), its surroundings (actors), and the relationship between use cases and actors
- A scenario describes an example of system use in term of a series of interaction between the actor/user and the system
- A use case is an abstraction that describes a class of scenarios
- An actor represents an entity that interacts with the system, or need to exchange information with the system, it can be human or a system

Identifying actors

- Need to answer the questions for identifying **actors**
  - Which user group are supported by the system to perform their work ?
  - Which user groups execute the system’s main functions ?
  - Which user groups perform secondary functions, such as maintenance and administration ?
  - With what external hardware or software system will the system interact ?

Identifying scenarios

- Need to answer the questions for identifying **scenarios**
  - What are the tasks that the actor wants the system to perform ?
  - What information does the actor access ? Who creates that data ? Can it be modified or removed ? By whom?
  - Which external changes does the actor need to inform the system about ? How often ? When ?
  - Which events does the system need to inform the actor about ? With what latency ?

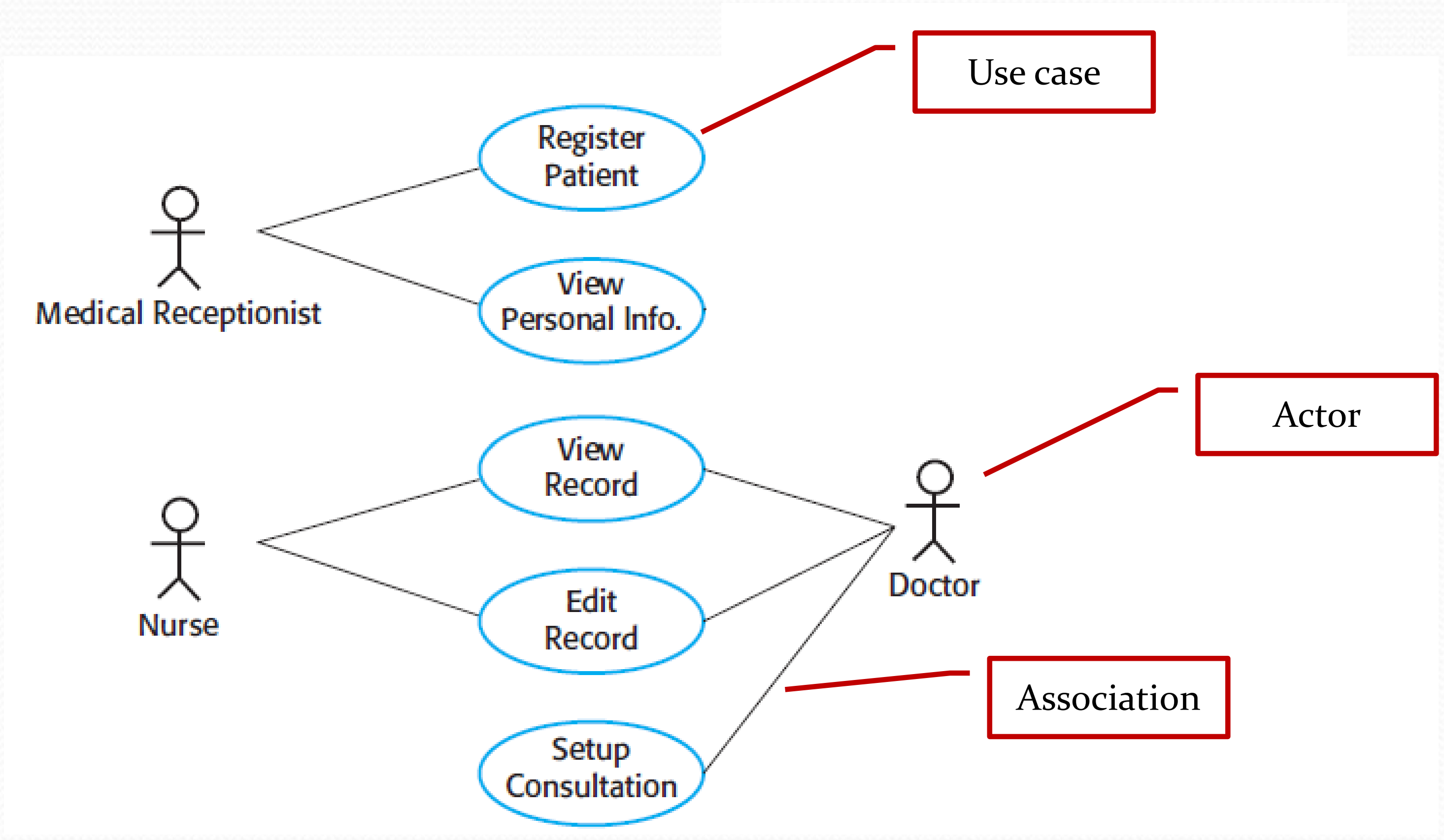
Identifying Use cases

- A use case represents a unit of functionality of value to an actor.
- A use case is an abstraction that describes a class of scenarios.
- Use case is often named with a verb to describe an action.

Example- Heathcare Patient MS Scenarios

- Scenario for collecting medical history
  - When a new patient attends a clinic, <u>a new record is created</u> by a medical receptionist and <u>personal information is added to it</u> (name, age, etc.).
  - A nurse then interviews the patient and <u>collects medical history</u>.
  - The patient then has an initial <u>consultation with a doctor</u> who makes diagnosis and if appropriate, recommends a course of treatment.

Use cases examples



Example - structured specification

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

Example - Tabular specification

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

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

Requirements checking – Good writing

- Validity/Correctness. Does the system provide the functions which best support the customer's needs?
- Consistency. Are there any requirements conflicts?
    - if it is impossible to satisfy both simultaneously, e.g:
- Completeness. Are all functions required by the customer included?
- Realism/feasible. Can the requirements be implemented given available budget and technology ?
    - E.g: request for an inexpensive system that analyzes huge amount of data and outputs the analysis resutls within seconds
- Verifiability. Can the requirements be checked?

Requirement checking (cont.)

- Conflicts resolving: Prioritize requirements
  - 1. Requirements that absolutely must be met (Essential)
  - 2. Requirements that are highly desirable but not necessary (Desirable)
  - 3. Requirements that are possible but could be eliminated (Optional)
- Requirement is verifiable/testable
  - Example: Water quality information must be accessible immediately

=> Water quality records must be retrieved within 5 seconds of a request

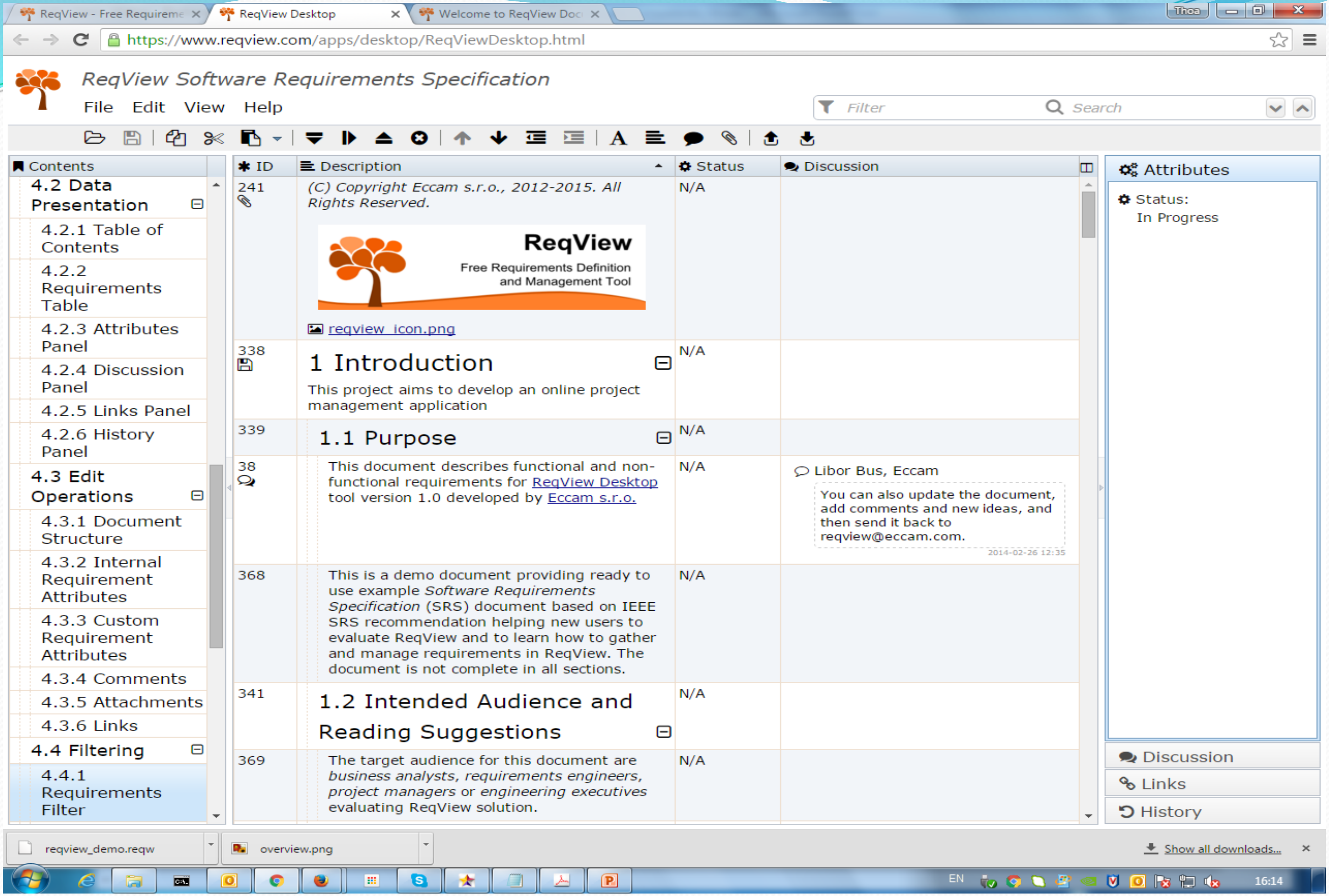
Requirements validation techniques

- Requirements reviews
  - Systematic manual analysis of the requirements (e.g. walkthrough, formal inspection)
- Prototyping
  - Using an executable model of the system to check requirements (e.g. Throw-away prototype).
- Functional test-case generation
  - Developing tests for requirements to check testability.
  - Designing these tests may reveal errors in the specification
  - Missing or ambiguous information in the requirements description may make it difficult to formulate tests

Requirements management

- Requirements management is a part of overall project management. It concerns
  - Discovering, classifying, organizing and documenting requirements
  - Changes to requirements
  - Requirement traceability

- Requirements have to be numbered following some identification scheme. Each requirement has a unique identifier for easy reference



ReqView Software Requirements Specification

Contents
4.2 Data Presentation
4.2.1 Table of Contents
4.2.2 Requirements Table
4.2.3 Attributes Panel
4.2.4 Discussion Panel
4.2.5 Links Panel
4.2.6 History Panel
4.3 Edit Operations
4.3.1 Document Structure
4.3.2 Internal Requirement Attributes
4.3.3 Custom Requirement Attributes
4.3.4 Comments
4.3.5 Attachments
4.3.6 Links
4.4 Filtering
4.4.1 Requirements Filter

ID	Description	Status	Discussion
241	*(C) Copyright Eccam s.r.o., 2012-2015. All Rights Reserved.* ReqView Free Requirements Definition and Management Tool reqview_icon.png	N/A	
338	1 Introduction This project aims to develop an online project management application	N/A	
339	1.1 Purpose	N/A	
38	This document describes functional and non-functional requirements for ReqView Desktop tool version 1.0 developed by Eccam s.r.o.	N/A	Libor Bus, Eccam You can also update the document, add comments and new ideas, and then send it back to reqview@eccam.com. 2014-02-26 12:35
368	This is a demo document providing ready to use example *Software Requirements Specification* (SRS) document based on IEEE SRS recommendation helping new users to evaluate ReqView and to learn how to gather and manage requirements in ReqView. The document is not complete in all sections.	N/A	
341	1.2 Intended Audience and Reading Suggestions	N/A	
369	The target audience for this document are *business analysts, requirements engineers, project managers* or *engineering executives* evaluating ReqView solution.	N/A	

Attributes
Status:
In Progress

Requirements change

- A requirement may change, is removed or a new requirement is added.
- Requirements changes should be stored and tracked
- Requirement traceability is a part of management of requirement changes, taking into account requirements relationships.

Requirements Documentation

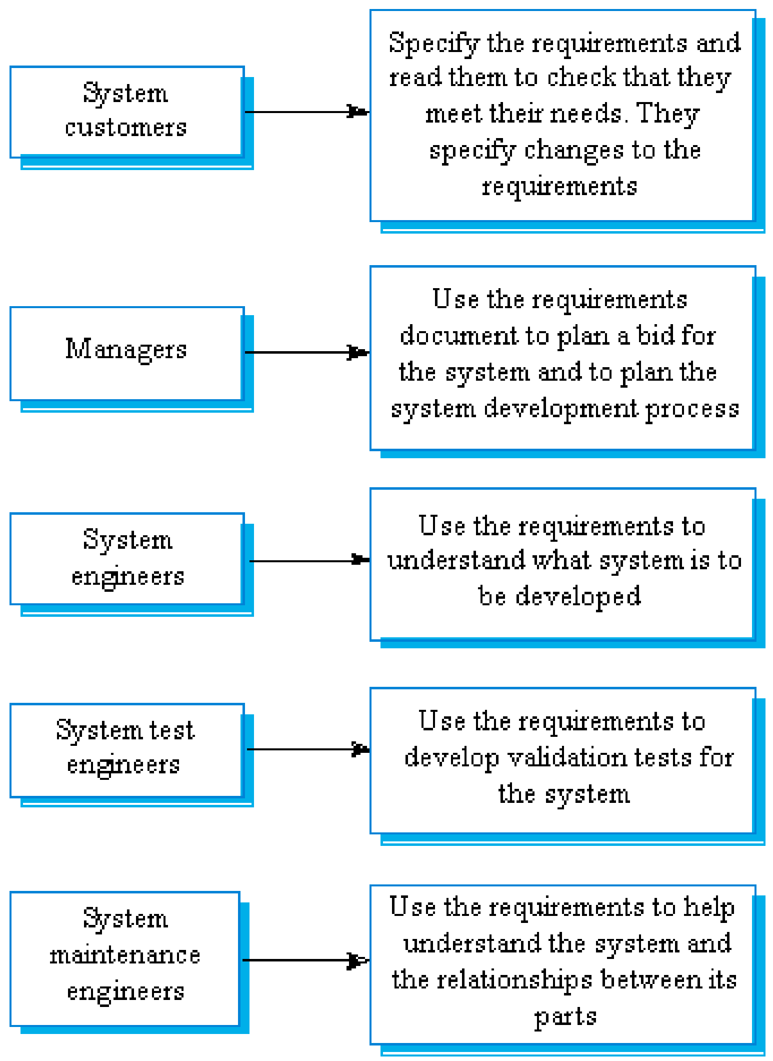
The requirements document

- The requirements document is the 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 far as possible, it should set out WHAT the system should do rather than HOW it should do it

Users of a requirements document


System customers
Specify the requirements and read them to check that they meet their needs. They specify changes to the requirements
Managers
Use the requirements document to plan a bid for the system and to plan the system development process
System engineers
Use the requirements to understand what system is to be developed
System test engineers
Use the requirements to develop validation tests for the system
System maintenance engineers
Use the requirements to help understand the system and the relationships between its parts

Standards for requirements documents

- A number of large organisations, like IEEE and US Department of Defence, have defined standards for requirements documents.
- The most widely known standard is IEEE/ANSI 830-1998.
- Defines a generic structure for a requirements document that must be instantiated for each specific system.

IEEE requirements standard

- The IEEE standard suggests the following structure for requirements documents:
    - Introduction
    - General description
    - Specific requirements
    - Appendices
    - Index

IEEE requirements standard

- Introduction
  - Purpose of the requirements document
  - Scope of the product
  - Definitions, acronyms and abbreviations
  - References
  - Overview of the remainder of the document

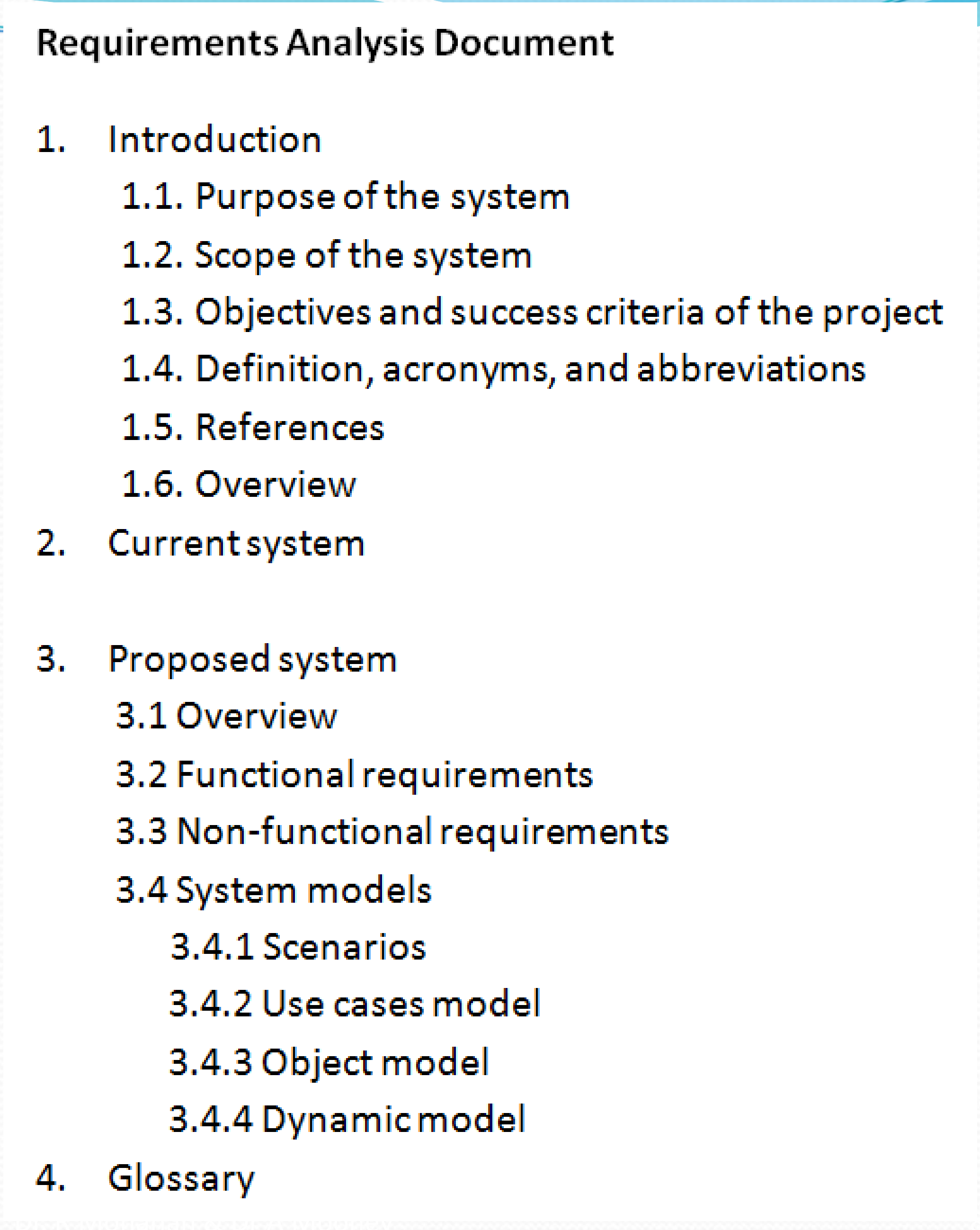
IEEE requirements standard

- General description
  - Product perspective
  - Product functions
  - User characteristics
  - General constraints
  - Assumptions and dependencies

IEEE requirements standard

- Specific requirements
  - These cover functional, non-functional and interface requirements.
  - Most substantial part of the document, but because organisations are so different, it is not appropriate to define a standard structure for this section.
  - The requirements may document external interfaces, describe system functionality and performance, specify logical database requirements, design constraints and quality characteristics.

Requirements Document Possible Structure



Requirements Analysis Document

1. Introduction
   - 1.1. Purpose of the system
   - 1.2. Scope of the system
   - 1.3. Objectives and success criteria of the project
   - 1.4. Definition, acronyms, and abbreviations
   - 1.5. References
   - 1.6. Overview
2. Current system

3. Proposed system
   - 3.1 Overview
   - 3.2 Functional requirements
   - 3.3 Non-functional requirements
   - 3.4 System models
     - 3.4.1 Scenarios
     - 3.4.2 Use cases model
     - 3.4.3 Object model
     - 3.4.4 Dynamic model
4. Glossary

Summary - Key points

- Requirements set out what the system should do and define constraints on its operation and implementation.
- Functional requirements set out services the system should provide.
- Non-functional requirements constrain the system being developed or the development process.
- Software requirements document (SRD) is an agreed statement of the system requirements. Both system customers and software developer are users of SRD

Summary - Key points

- Requirement engineering process includes feasibility study, requirements elicitation and analysis, requirement specification, requirement validation, and requirements management
- Apart from natural language, some graphic models are used in requirement analysis (some UML diagrams), which is the basis for system design
- The IEEE standard is a useful starting point for defining more detailed specific requirements standards.