ICT 3020 (2022/23)
Fundamentals of Software Engineering
Module 7: Requirements Engineering: Requirements Specification

Lighton Phiri <lighton.phiri@unza.zm>
Department of Library & Information Science
University of Zambia
https://bit.ly/3uI5u5V

Announcements—August 28, 2023 (1/2)

- **Research Proposal Presentations**
  - September 5, 2023
    - 12H00–15H30 GMT+2
    - Conference Room, Fifth Floor, School of Education Building
  - Presentation to be done by order of project team numbers
    - All team members required to actively participate

Announcements—August 28, 2023 (2/2)

- **Scheduled assessments**
  - August 28, 2023—Requirements Elicitation
    - Stakeholder Analysis | September 1, 2023
    - Elicitation Techniques | September 8, 2023
    - Gathering requirements | September 15, 2023

Lecture Outline

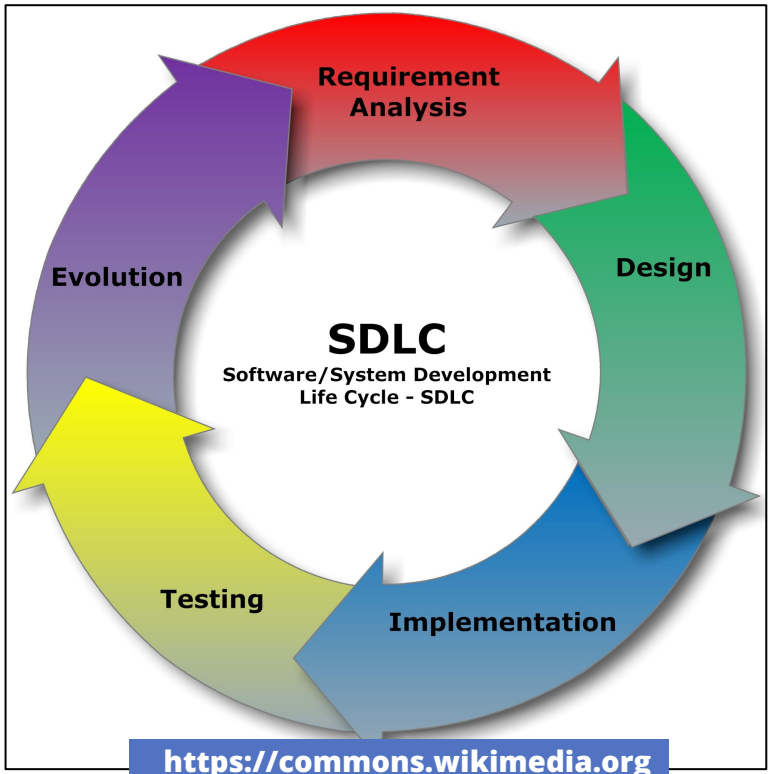
- Requirements Specification
- Requirements Specification Notations
- Software Requirements Specification Document
- Requirements Prioritisation
- Requirements Validation
- Requirements Change

Lecture Outline

- **Requirements Specification**
  - Introduction
  - Requirements and Design
  - Guidelines for Writing Requirements
  - Natural Language Specification
- **Requirements Specification Notations**
- **Software Requirements Specification Document**
- **Requirements Prioritisation**
- **Requirements Validation**
- **Requirements Change**

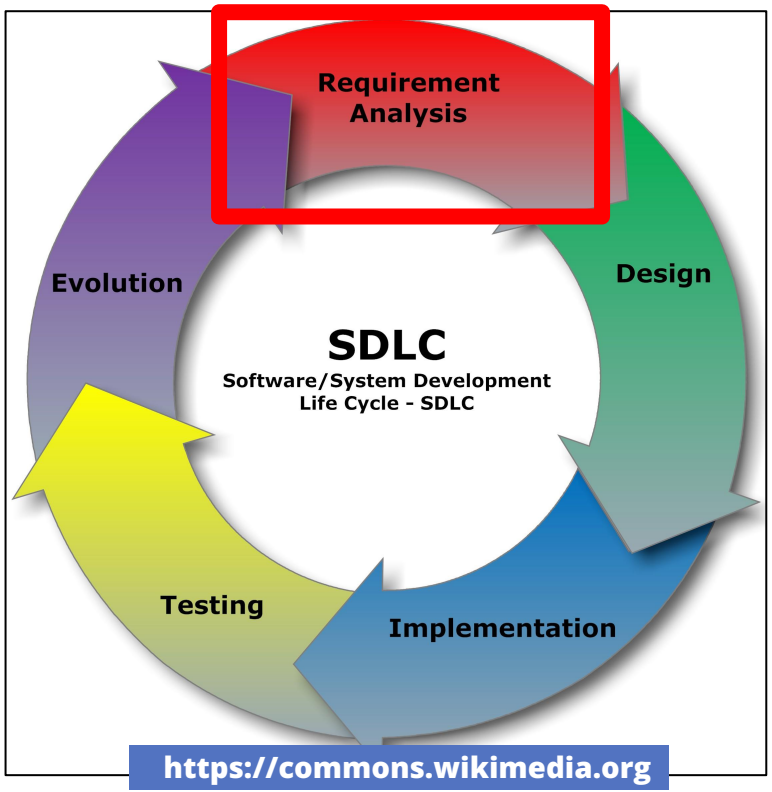
Requirements Specification (1/4)

- Real software processes are interleaved sequences of technical, collaborative and managerial activities with the overall goal of specifying, designing, implementing and testing a software system.



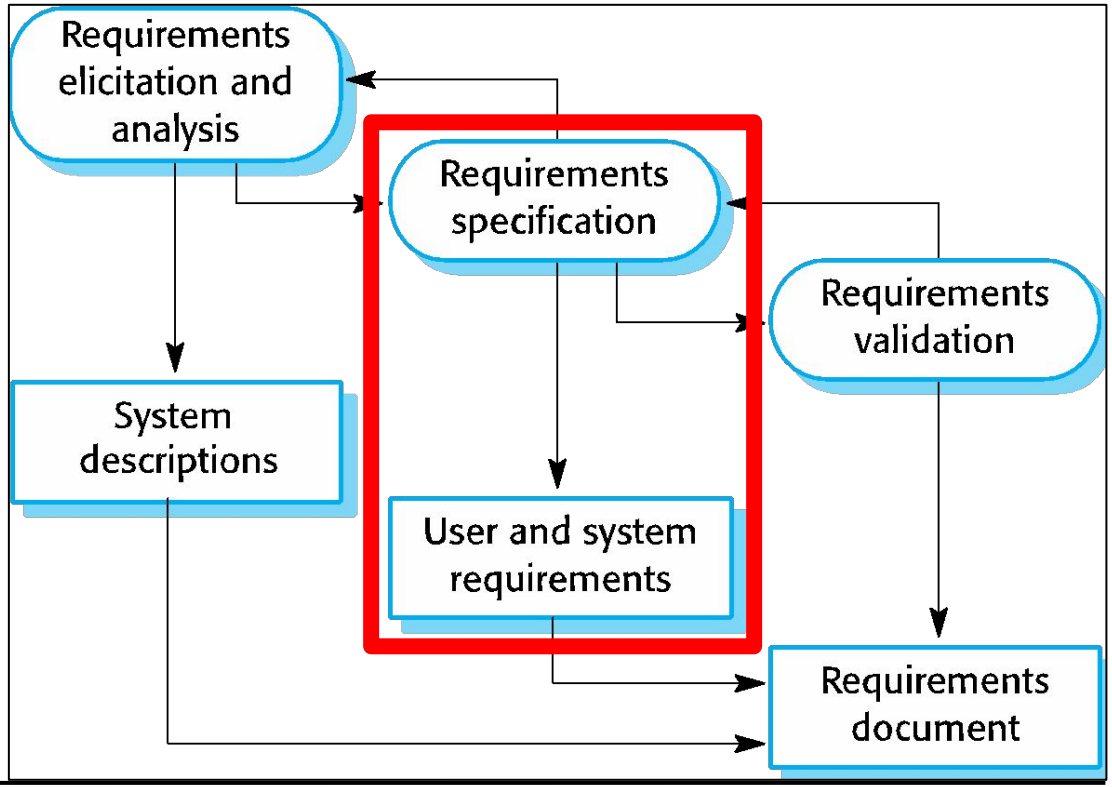
Requirements Specification (1/4)

- Real software processes are interleaved sequences of technical, collaborative and managerial activities with the overall goal of specifying, designing, implementing and testing a software system.



Requirements Specification (2/4)

- Specification involves the mapping of requirements gathered into a formal requirements document



Requirements Specification (3/4)

- Process of formally writing down 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.

Requirements Specification (4/4)

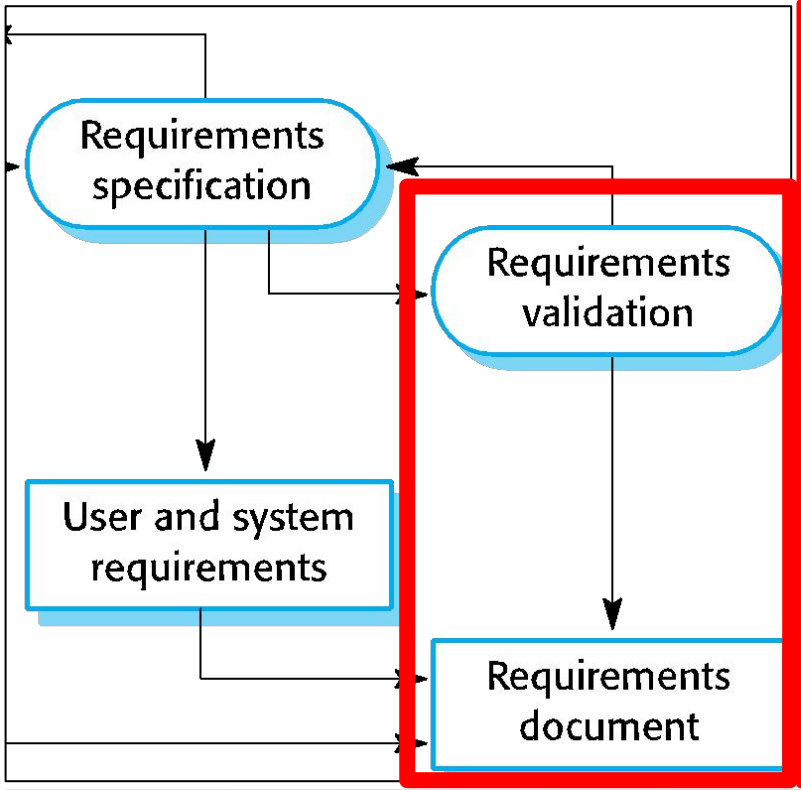
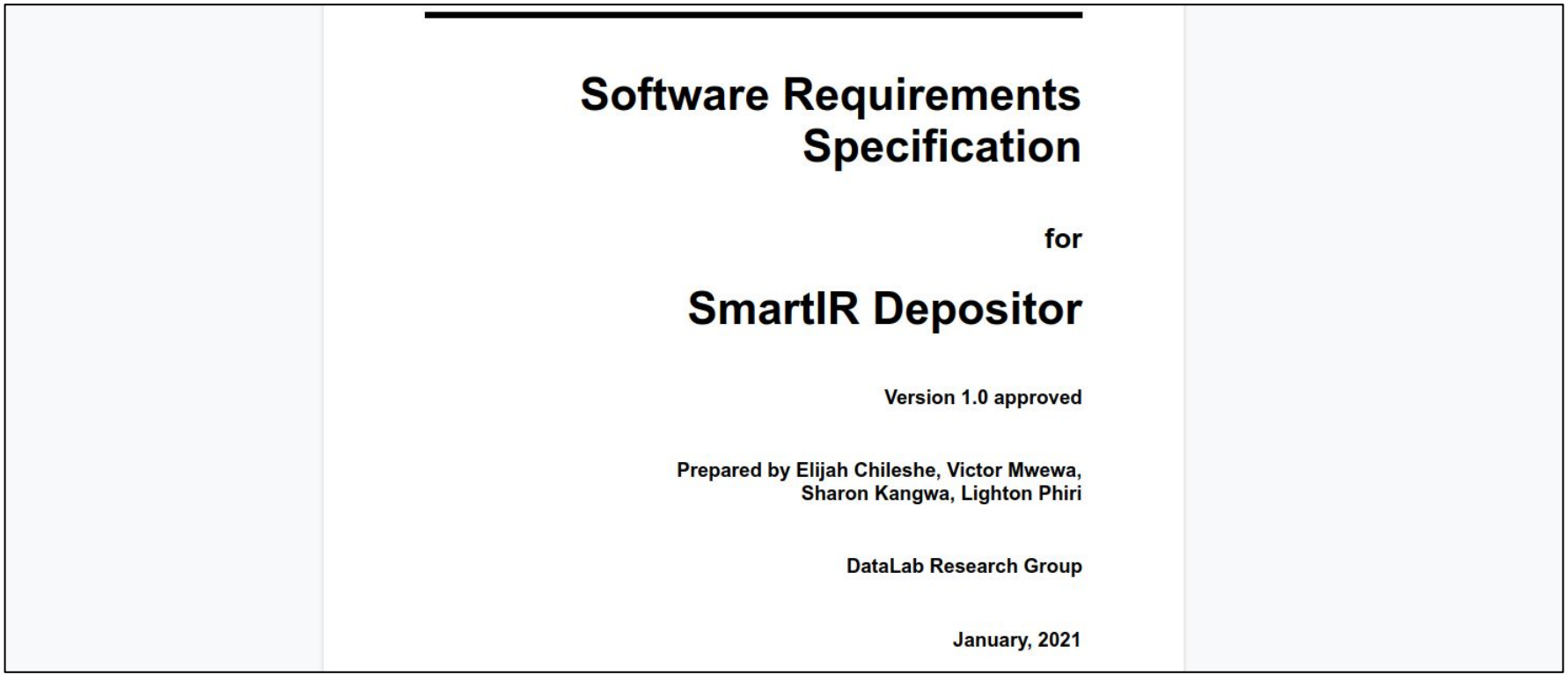


Table of Contents

Announcements—July 21, 2021 (3/3)



Software Requirements Specification

for

SmartIR Depositor

Version 1.0 approved

Prepared by Elijah Chileshe, Victor Mwewa, Sharon Kangwa, Lighton Phiri

DataLab Research Group

January, 2021

Requirements and Design

- **In principle, requirements should state what the system should do and the design should describe how it does this**
- **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

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

Natural Language Specification (1/2)

- 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

Natural Language Specification (2/2)

- **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

Lecture Outline

- **Requirements Specification**
- **Requirements Specification Notations**
  - Introduction
  - Structured Notation
  - Form-Based Notation
  - Tabular Notation
  - Use Case Notation
- **Software Requirements Specification Document**
- **Requirements Prioritisation**
- **Requirements Validation**
- **Requirements Change**

Requirements Specification Notation (1/2)

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.

Requirements Specification Notation (2/2)

Notation	Description
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

Natural Language Specification: Structured Specification

- 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.

Natural Language Specification: Form-Based Specification

- 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.

Natural Language Specification: Tabular Specification (1/2)

- 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.

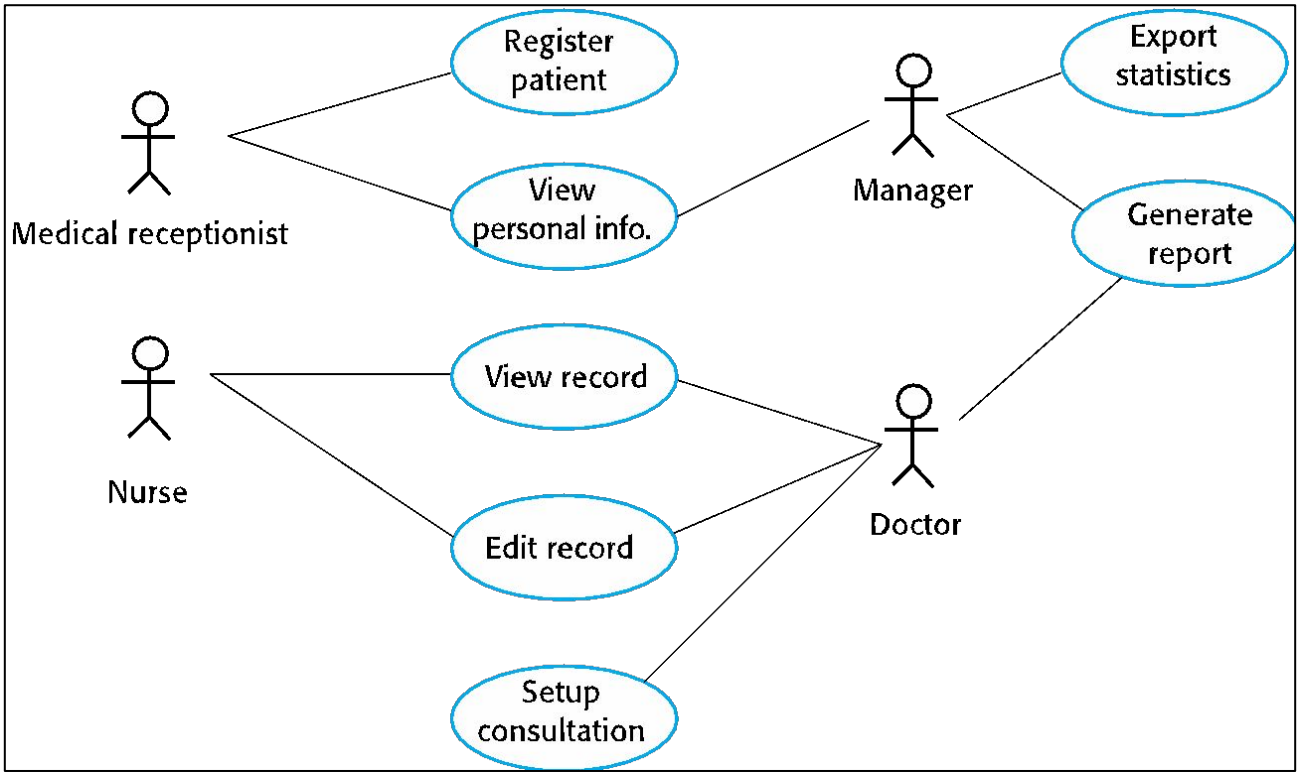
Natural Language Specification: Tabular Specification (2/2)

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

Natural Language Specification: Use Cases (1/2)

- Usecases are a kind of scenario that are included in the UML.
- Usecases 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.
- UML sequence diagrams may be used to add detail to usecases by showing the sequence of event processing in the system.

Natural Language Specification: Use Cases (2/2)



Lecture Outline

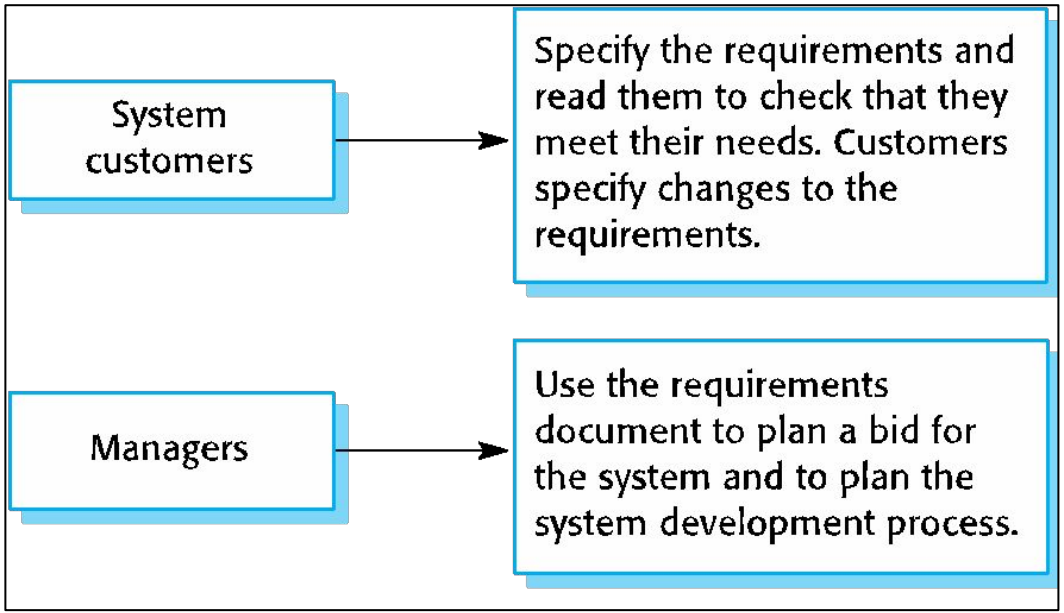
- **Requirements Specification**
- **Requirements Specification Notations**
- **Software Requirements Specification Document**
  - Introduction
  - Users of the SRS Document
  - SRS Document Structure
  - Case Example: smartIRDeposit
- **Requirements Prioritisation**
- **Requirements Validation**
- **Requirements Change**

Software Requirements Specification Document (1/3)

- The software 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 of WHAT the system should do rather than HOW it should do it.

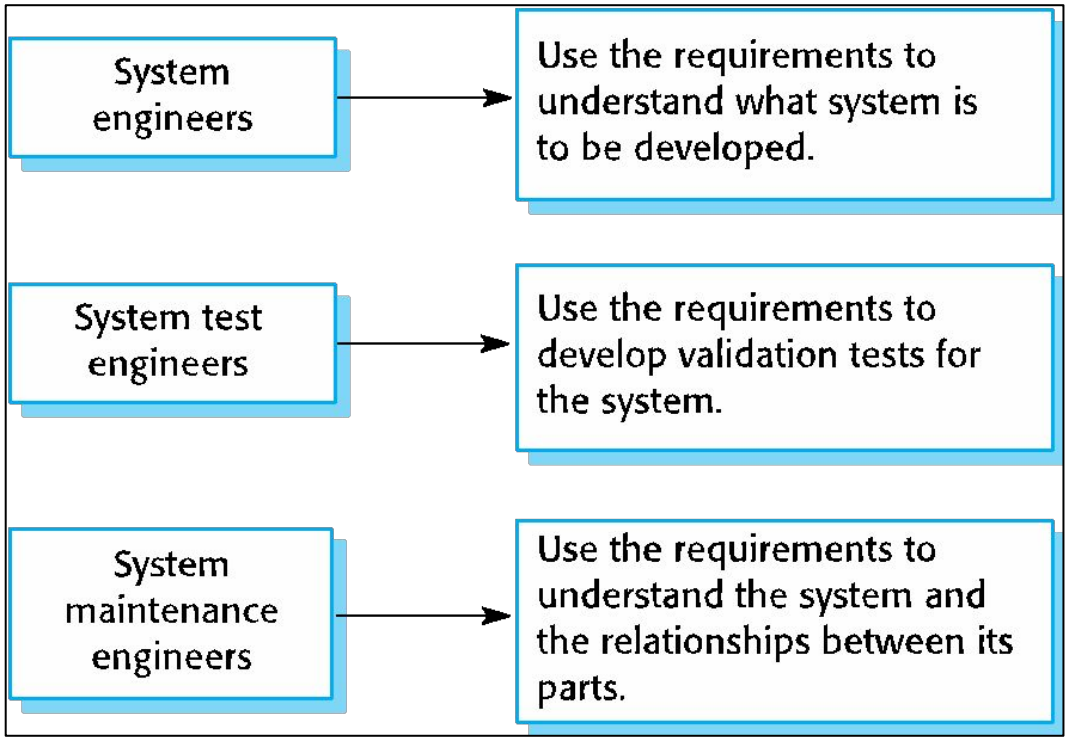
Software Requirements Specification Document (2/3)

- **The diverse set of potential users of the SRS document should be taken into account**
  - This is even more important in instances were expertise is outsourced



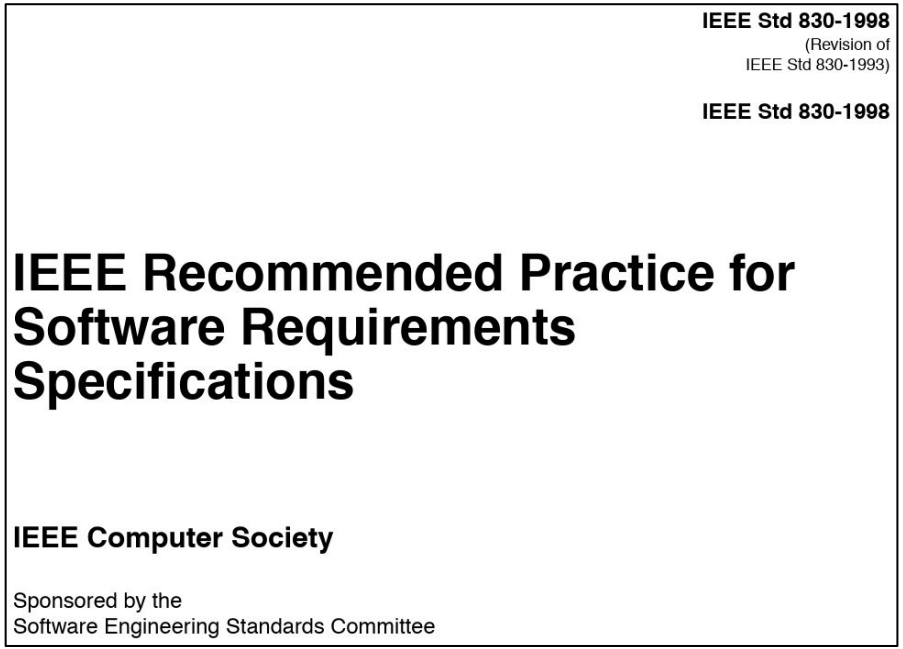
Software Requirements Specification Document (3/3)

- **The diverse set of potential users of the SRS document should be taken into account**
  - This is even more important in instances were expertise is outsourced



Software Requirements Specification Document Structure (1/2)

- **Invent or adopt a standard SRS template**
  - There are a number of SRS templates available
  - We shall use a modified version of the IEEE Std 830-1998 template with section 3 organised by feature


IEEE Std 830-1998
(Revision of
IEEE Std 830-1993)

IEEE Std 830-1998

IEEE Recommended Practice for Software Requirements Specifications

IEEE Computer Society

Sponsored by the
Software Engineering Standards Committee

Software Requirements Specification Document Structure (2/2)

Table of Contents

Software Requirements Specification Document Structure: Introduction (1/2)

- 1. Introduction
  - Presents an overview to help the reader understand how the SRS is organized and how to use it
- 1.1 Purpose
  - Identifies the product or application whose requirements are specified in this document
- 1.2 Document conventions
  - Standards and conventions used

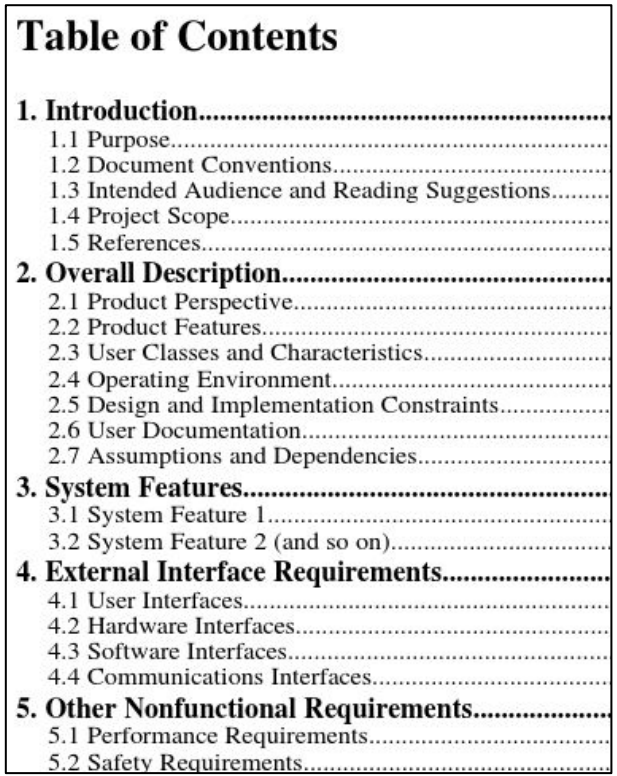


Table of Contents

Software Requirements Specification Document Structure: Introduction (2/2)

- **1.3 Project scope**
  - Description of the software being specified and its purpose
- **1.4 References**
  - List any documents or other resources to which this SRS refers.

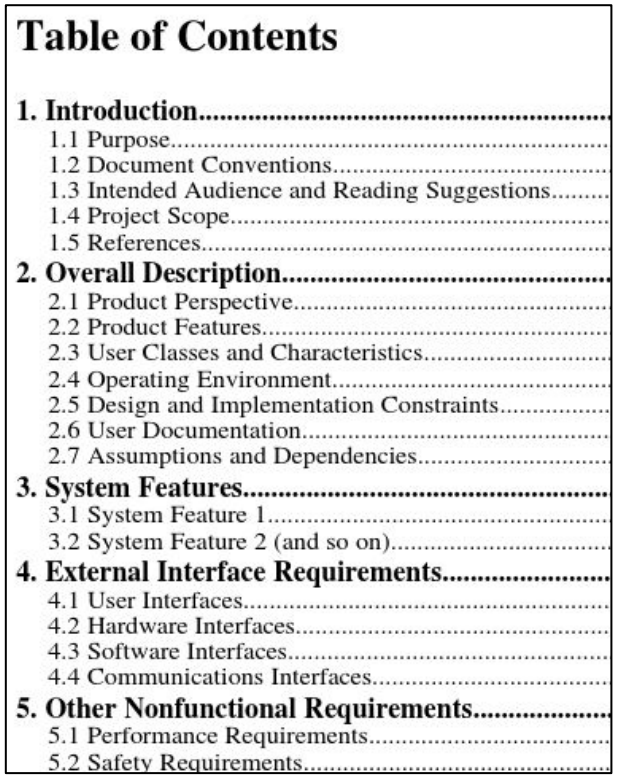


Table of Contents

Software Requirements Specification Document Structure: Description (1/3)

- **2. Overall description**
  - Overview of the product and the environment in which it will be used
- **2.1 Product perspective**
  - Describe the product's context and origin
- **2.2 User classes and characteristics**
  - Identify the various user classes that you anticipate will use this product, and describe their pertinent characteristics

Table of Contents

Software Requirements Specification Document Structure: Description (2/3)

- **2.3 Operating environment**
  - Environment in which the software will operate, including the hardware platform; operating systems and versions
- **2.4 Design and implementation constraints**
  - Programming language must be used, a particular code library that has already had time invested to develop it needs to be used

Table of Contents

Software Requirements Specification Document Structure: Description (3/3)

- **2.5 Assumptions and dependencies**
  - Identify any dependencies the project or system being built has on external factors or components outside its control
  - system functionality; business-related assumptions appear in the vision and scope document

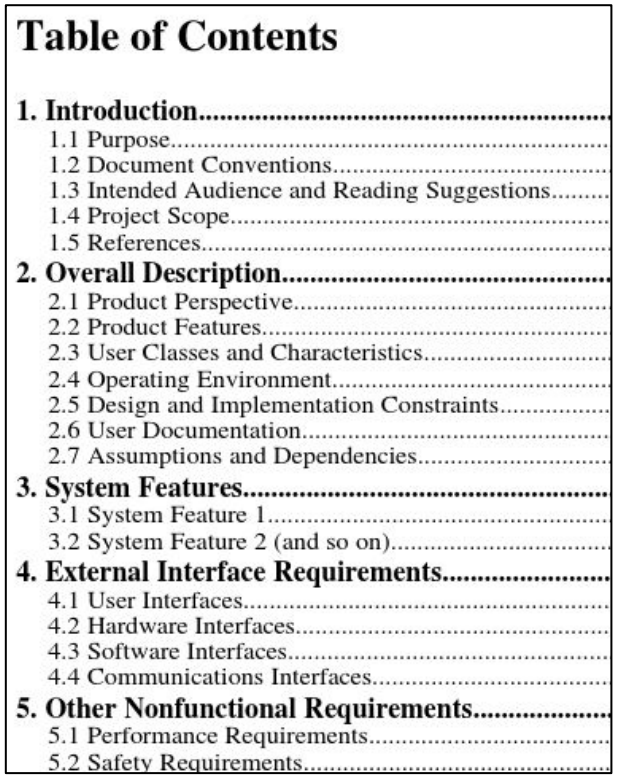


Table of Contents

Software Requirements Specification Document Structure: Features (1/2)

- **3. System features**
  - State the name of the feature in just a few words, such as “3.1 Spell Check.” Repeat section 3.x with its subsections 3.x.1 and 3.x.2 for each system feature.
- **3.x.1 Description**
  - Short description of the feature and indicate whether it is of high, medium, or low priority

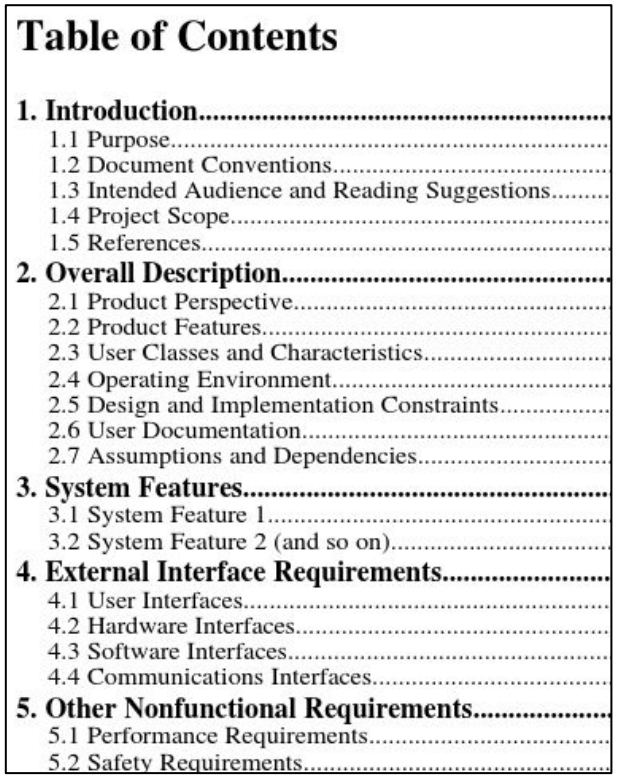


Table of Contents

Software Requirements Specification Document Structure: Features (2/2)

- **3.x.2 Functional requirements**
  - Itemize the specific functional requirements associated with this feature

Table of Contents

Software Requirements Specification Document Structure: Data Req. (1/3)

- **4. Data requirements**
  - Data that the system will consume as inputs, process in some fashion, or create as outputs
- **4.1 Logical data model**
  - Numerous notations exist for data modeling, including entity-relationship diagrams and UML class diagrams.
- **4.2 Data dictionary**
  - Data structures and the meaning, data type, length and formats allowed

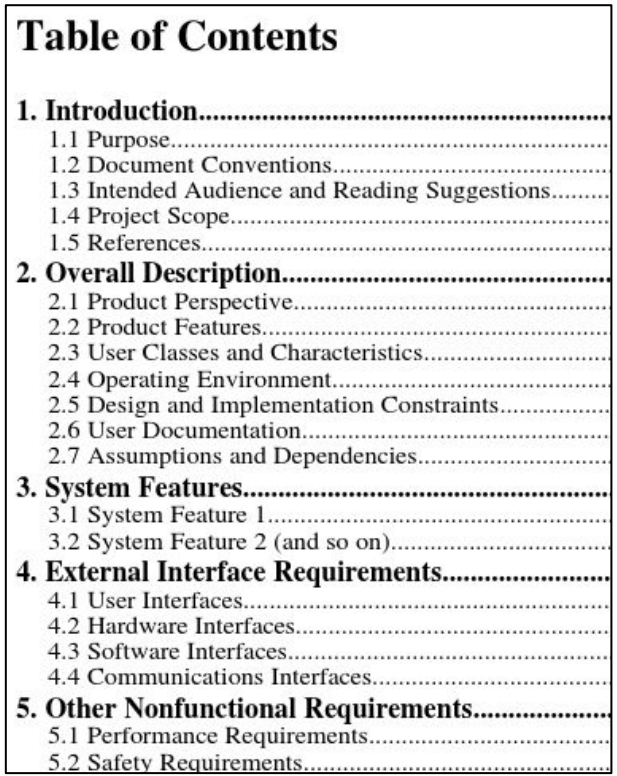


Table of Contents

Software Requirements Specification Document Structure: Data Req. (2/3)

- **4. Data requirements**
  - Data that the system will consume as inputs, process in some fashion, or create as outputs
- **4.1 Logical data model**
  - Numerous notations exist for data modeling, including entity-relationship diagrams and UML class diagrams.
- **4.2 Data dictionary**
  - Data structures and the meaning, data type, length and formats allowed

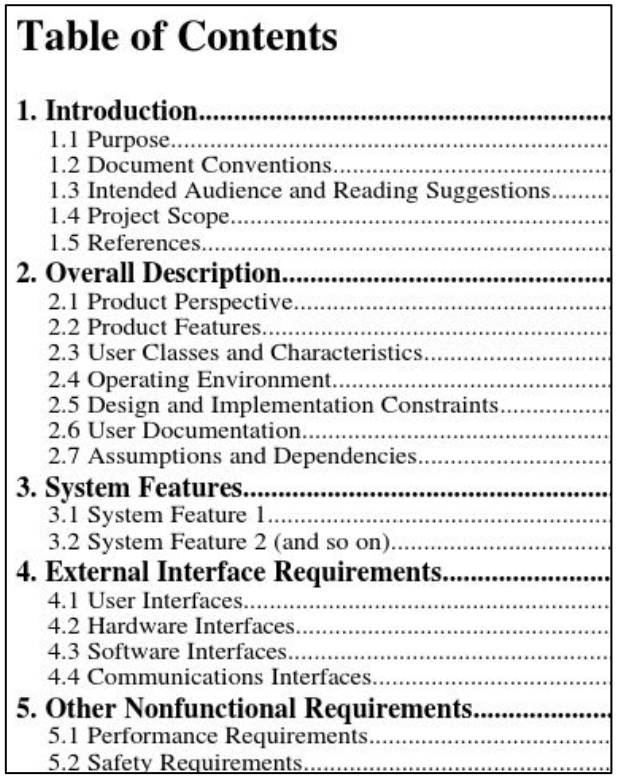


Table of Contents

Software Requirements Specification Document Structure: Data Req. (3/3)

- **4.3 Reports**
  - If your application will generate any reports, identify them here and describe their characteristics.
- **4.4 Data acquisition, integrity, retention, and disposal**
  - Describe how data is acquired and maintained.

Table of Contents

Software Requirements Specification Document Structure: Interfaces (1/3)

- **5. External interface requirements**
  - information to ensure that the system will communicate properly with users and with external hardware or software elements
- **5.1 User interfaces**
  - Describe the logical characteristics of each user interface that the system needs
- **5.2 Software interfaces**
  -

Table of Contents

Software Requirements Specification Document Structure: Interfaces (2/3)

- **5.2 Software interfaces**
  - Describe the connections between this product and other software components
- **5.3 Hardware interfaces**
  - Describe the characteristics of each interface between the software components and hardware components

Table of Contents

Software Requirements Specification Document Structure: Interfaces (3/3)

- **5.4 Communications interfaces**
  - State the requirements for any communication functions the product will use

Table of Contents

Software Requirements Specification Document Structure: Quality (1/3)

- **6. Quality attributes**
  - Specifies nonfunctional requirements other than constraints
- **6.1 Usability**
  - Usability requirements deal with ease of learning, ease of use, error avoidance and recovery, efficiency of interactions, and accessibility

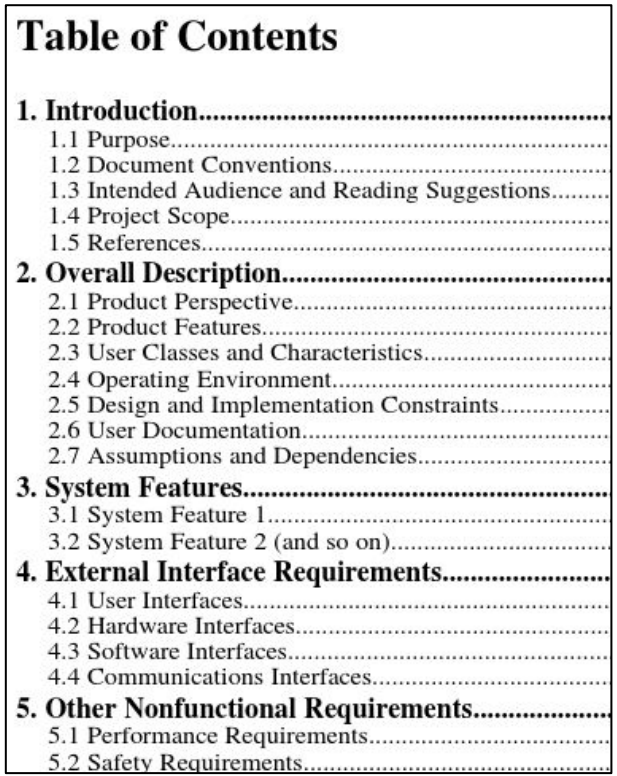


Table of Contents

Software Requirements Specification Document Structure: Quality (2/3)

- **6.2 Performance**
  - State specific performance requirements for various system operations.
- **6.3 Security**
  - Specify any requirements regarding security or privacy issues that restrict access to or use of the Product.

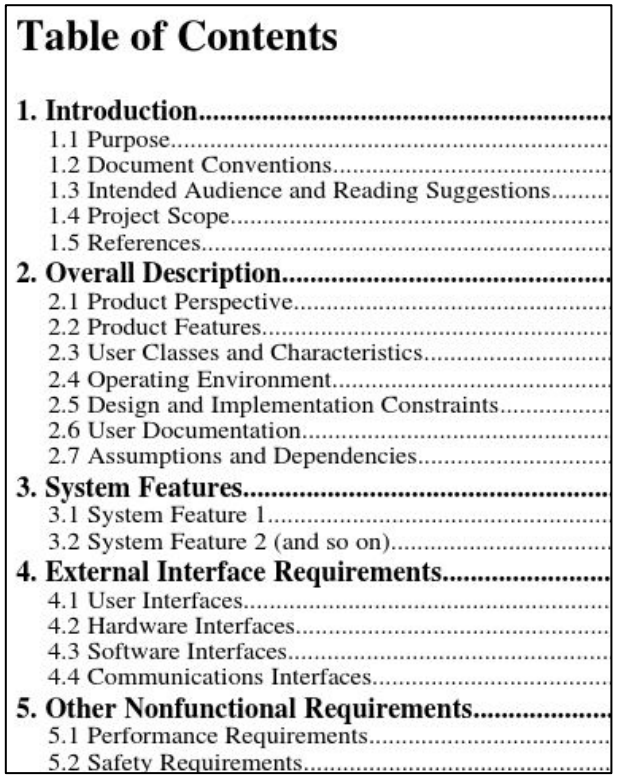


Table of Contents

Software Requirements Specification Document Structure: Quality (3/3)

- **6.4 Safety**
  - Specify requirements that are concerned with possible loss, damage, or harm that could result from use of the product.
- **6.x [Others]**
  - Additional product quality attribute to describe characteristics that will be important either to customers or to developers and maintainers.

Table of Contents

Software Requirements Specification Document Structure: Internationalisation

- **7. Internationalization and localization requirements**
  - Internationalization and localization requirements ensure that the product will be suitable for use in nations, cultures, and geographic locations other than those in which it was created

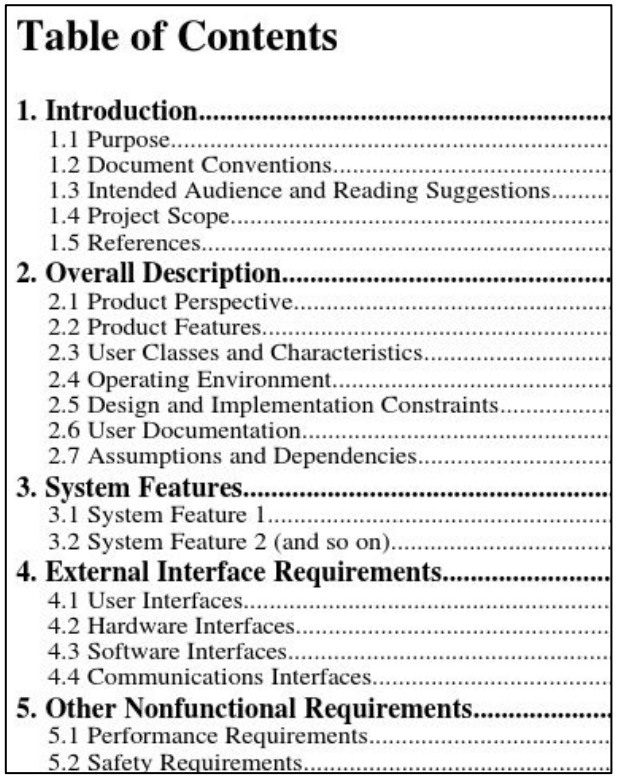


Table of Contents

Software Requirements Specification Document Structure: Additional Req.

- 8. [Other requirements]
  - Define any other requirements that are not covered elsewhere in the SRS.

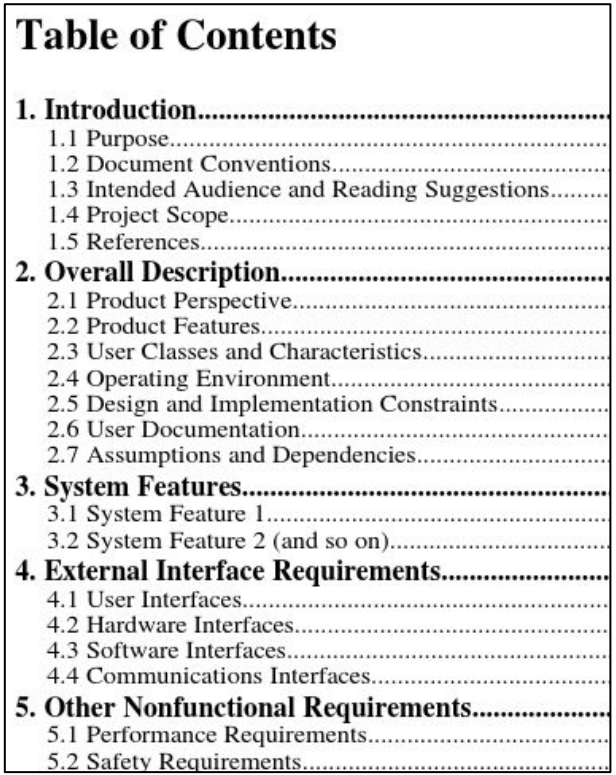


Table of Contents

Software Requirements Specification Document Structure: Appendices

- **Appendix A: Glossary**
  - Define any specialized terms that a reader needs to know to understand the SRS, including acronyms and abbreviations
- **Appendix B: Analysis models**
  - Optional section includes or points to pertinent analysis models such as data flow diagrams, feature trees, state-transition diagrams, or entity-relationship diagrams

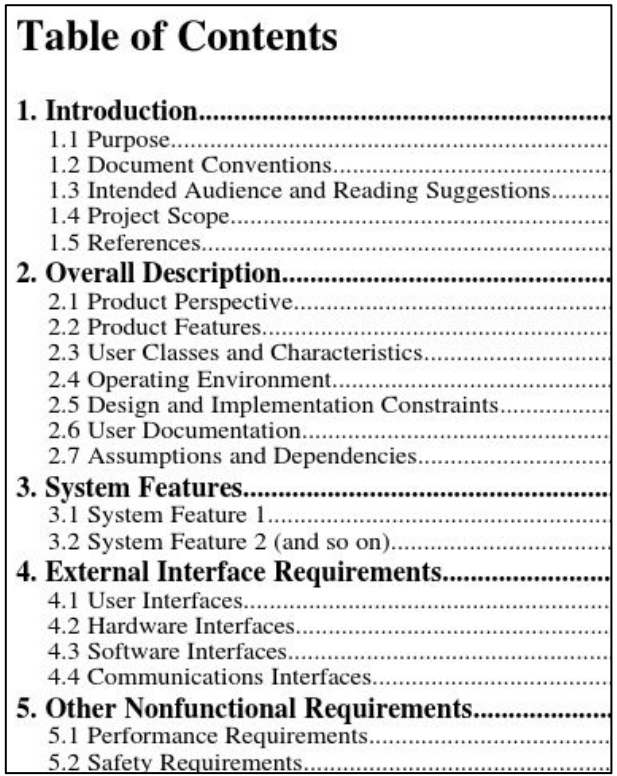


Table of Contents

Case Example: sIRd Platform (1/2)

Software Requirements Specification

for

Smart IR Deposit (sIRd) System

Version 1.0 approved

Prepared by Lighton Phiri <lighton.phiri@unza.zm>

DataLab Research Group, The University of Zambia

August 1, 2020

Table of Contents

Case Example: sIRd Platform (2/2)

Revision History

Name	Date	Reason For Changes	Version
Initial version	01/08/20	Initial version of SRS document	1.0

- Some portions of the SRS document that require design artefacts as input can be versioned

Software Requirements Specification for **Smart IR Deposit (sIRd) System**

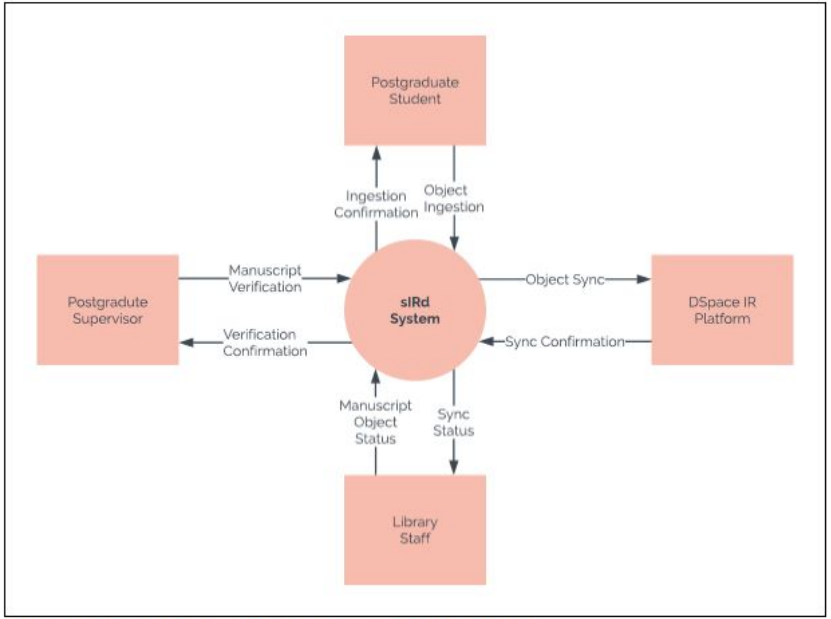


Figure 1: Context Diagram for sIRd System Version 1.0

2.2. Product Features

Lecture Outline

- **Requirements Specification**
- **Requirements Specification Notations**
- **Software Requirements Specification Document**
- **Requirements Prioritisation**
  - Revisiting the SmartIR Deposit System
  - Requirements Prioritisation Techniques
- **Requirements Validation**
- **Requirements Change**

Problem Statement: smartIR Deposit Case Example (1/2)

- **The University of Zambia has an Institutional Repository (IR) that is used to store and make available scholarly research out—digital objects**
  - The ingestion of digital objects into the IR is performed by a lean group of staff—two individuals presently, resulting in a backlog of digital objects
  - In order to address the problem resulting from a backlog of digital objects, there is a proposal to implement a third-party application for efficiently and effectively ingesting digital objects into the IR

Problem Statement: smartIR Deposit Case Example (2/2)

#	Feature/Requirement
1	FR 1—Automatic classification of object collection
2	FR 2—Automatic classification of object subjects
3	FR 3—Automatic classification of ETD-ms elements
4	FR 4—CRUD operations implementations
5	FR 5—Authentication and authorisation via single signon
6	FR 6—Synchronisation with IR
7	FR 7—User management

- **Feature set for SmartIR Deposit system**
  - This excludes non-functional requirements

Requirements Prioritisation (1/3)

3.1. System Feature 1

<Don't really say "System Feature 1." State the feature name in just a few words.>

3.1.1. Description and Priority

<Provide a short description of the feature and indicate whether it is of High, Medium, or Low priority. You could also include specific priority component ratings, such as benefit, penalty, cost, and risk (each rated on a relative scale from a low of 1 to a high of 9).>

3.1.2. Stimulus/Response Sequences

- **It is not always possible to implement the system requirements in a single release**
  - Requirements need to be prioritised in order to identify the most important requirements

Requirements Prioritisation (1/4)

3.1. System Feature 1

<Don't really say "System Feature 1." State the feature name in just a few words.>

3.1.1. Description and Priority

<Provide a short description of the feature and indicate whether it is of High, Medium, or Low priority. You could also include specific priority component ratings, such as benefit, penalty, cost, and risk (each rated on a relative scale from a low of 1 to a high of 9).>

3.1.2. Stimulus/Response Sequences

- **It is not always possible to implement the system requirements in a single release**
  - Requirements/features need to be prioritised in order to identify the most important requirements

Requirements Prioritisation (2/4)

- **There are a number of techniques that are used to prioritise requirements**
  - MOSCOW technique classifies requirements based on urgency
    - MUST—Mandatory
    - SHOULD—Of high priority
    - COULD—Preferred but not necessary
    - WOULD—Desired in future version of software
  - Using the feature set, you assigned the phrases MUST, SHOULD, COULD and WOULD

Problem Statement: smartIR Deposit Case Example (2/2)

#	Feature/Requirement	Prioritisation
1	FR 1—Automatic classification of object collection	SHOULD
2	FR 2—Automatic classification of object subjects	COULD
3	FR 3—Automatic classification of ETD-ms elements	COULD
4	FR 4—CRUD operations implementations	MUST
5	FR 5—Authentication & authorisation via single signon	WOULD
6	FR 6—Synchronisation with IR	MUST
7	FR 7—User management	MUST

- 1-Low-WOULD, 2-Medium-COULD, 3-High-SHOULD, or 4-Urgent-MUST

Requirements Prioritisation (3/4)

- **There are a number of techniques that are used to prioritise requirements**
  - A simple ranking technique can be used by enumerating the features
    - Analytical Hierarchy Process (AHP) is a commonly used approach for determining priority using pairwise comparison
    - Simple subjective scales can also be used
      - High Priority (1), Medium Priority (2) and Low Priority (3)
  - The highly ranked features are implemented first

Problem Statement: smartIR Deposit Case Example | Simple Ranking

#	Feature/Requirement	Prioritisation
1	FR 1—Automatic classification of object collection	4
2	FR 2—Automatic classification of object subjects	5
3	FR 3—Automatic classification of ETD-ms elements	6
4	FR 4—CRUD operations implementations	1
5	FR 5—Authentication & authorisation via single signon	7
6	FR 6—Synchronisation with IR	2
7	FR 7—User management	3

- Simple ranking mechanism can be used

Problem Statement: smartIR Deposit Case Example | Pairwise Comparison

	FR 1	FR 2	FR 3	FR 4	FR 5	FR 6	FR 7
FR 1	—	1	1	0	1	0	0
FR 2	0	—	1	0	1	0	0
FR 3	0	0	—	0	1	0	0
FR 4	1	1	1	—	1	0	1
FR 5	0	0	0	0	—	0	0
FR 6	1	1	1	1	1	—	1
FR 7	1	1	1	0	1	0	—

- Each of the requirements compared in pairs to determine the most important ones

Problem Statement: smartIR Deposit Case Example | Pairwise Comparison

	FR 1	FR 2	FR 3	FR 4	FR 5	FR 6	FR 7	
FR 1	—	1	1	0	1	0	0	3
FR 2	0	—	1	0	1	0	0	2
FR 3	0	0	—	0	1	0	0	1
FR 4	1	1	1	—	1	0	1	5
FR 5	0	0	0	0	—	0	0	0
FR 6	1	1	1	1	1	—	1	6
FR 7	1	1	1	0	1	0	—	4

- Each of the requirements compared in pairs to determine the most important ones

Problem Statement: smartIR Deposit Case Example | Pairwise Comparison

#	Feature/Requirement	Prioritisation
1	FR 1—Automatic classification of object collection	3
2	FR 2—Automatic classification of object subjects	2
3	FR 3—Automatic classification of ETD-ms elements	1
4	FR 4—CRUD operations implementations	5
5	FR 5—Authentication & authorisation via single signon	0
6	FR 6—Synchronisation with IR	6
7	FR 7—User management	4

- Simple ranking mechanism can be used

Requirements Prioritisation (4/4)

- **There are a number of techniques that are used to prioritise requirements**
  - KANO analysis focuses on the satisfaction of stakeholders
    - The requirements are viewed from the perspective of the customers
      - Basic factors—Features that will result in customers not using the system
      - Performance factors—Features that will draw users
      - Excitement factors—customers are excited to use the system because of features

Problem Statement: smartIR Deposit Case Example | KANO Analysis

#	Feature/Requirement	Prioritisation
1	FR 1—Automatic classification of object collection	PERFORM
2	FR 2—Automatic classification of object subjects	PERFORM
3	FR 3—Automatic classification of ETD-ms elements	PERFORM
4	FR 4—CRUD operations implementations	BASIC
5	FR 5—Authentication & authorisation via single signon	EXCITEMENT
6	FR 6—Synchronisation with IR	BASIC
7	FR 7—User management	BASIC

- 1-Low-WOULD, 2-Medium-COULD, 3-High-SHOULD, or 4-Urgent-MUST

Lecture Outline

- **Requirements Specification**
- **Requirements Specification Notations**
- **Software Requirements Specification Document**
- **Requirements Prioritisation**
- **Requirements Validation**
  - Requirements Validation and Checking
  - Requirements Review
- **Requirements Change**

Requirements Validation and Checking (1/3)

- **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 Validation and Checking (2/3)

- 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 and Checking (3/3)

- **There are a number of techniques that can be used to validate and check requirements**
  - Requirements reviews—Systematic manual analysis of the requirements.
  - Prototyping—Using an executable model of the system to check requirements. Covered in Chapter 2.
  - Test-case generation—Developing tests for requirements to check testability.

Requirements Review (1/2)

- 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 Review (2/2)

- 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?

(Wiegers and Beatty) Example Review Defect Checklist—Completeness (1/2)

- Do the requirements address all known customer or system needs?
- Is any needed information missing? If so, is it identified as TBD?
- Have algorithms intrinsic to the functional requirements been defined?
- Are all external hardware, software, and communication interfaces defined?
- Is the expected behaviour documented for all anticipated error conditions?

(Wiegers and Beatty) Example Review Defect Checklist—Completeness (2/2)

- Do the requirements provide an adequate basis for design and test?
- Is the implementation priority of each requirement included?
- Is each requirement in scope for the project, release, oor iteration?

(Wiegers and Beatty) Example Review Defect Checklist—Correctness

- Do any requirements conflict with other requirements?
- Is each requirement written in clear, concise, unambiguous, grammatically correct language?
- Is each requirement verifiable by testing, demonstration review, or analysis?
- Are any specified error messsages clear and meaningful?
- Are all requirements actually requirements, not solutions or constraints?
- Are the requirements technically feasible and implementable within known constraints?

(Wiegers and Beatty) Example Review Defect Checklist—Quality Attributes

- Are all usability, performance, security and safety objectives properly specified?
- Are other quality attributes documented, with the acceptable trade-offs specified?
- Are the time-critical function identified and timing criteria specified for them?
- Have internationalisation and localisation issues been adequately addressed?
- Are all the quality requirements measurable?

(Wiegers and Beatty) Example Review Defect Checklist—Organisation and Traceability

- Are the requirements organised in a logical and accessible way?
- Are all cross-references to other requirements and documents correct?
- Are all requirements written at a consistent and appropriate level of detail?
- Is each requirement uniquely and correctly labeled?
- Is each functional requirement traced back to its origin?

(Wiegers and Beatty) Example Review Defect Checklist—Additional Issues

- Are any use cases or process flows missing?
- Are any alternative flows, exceptions, or other information missing from use cases?
- Are all of the business rules identified?
- Are there any missing visual models that would provide clarity or completeness?
- Are all necessary report specifications present and complete?

Lecture Outline

- Requirements Specification
- Requirements Specification Notations
- Software Requirements Specification Document
- Requirements Prioritisation
- Requirements Validation
- Requirements Change
  - Changing Requirements
  - Requirements Evolution
  - Requirements Management

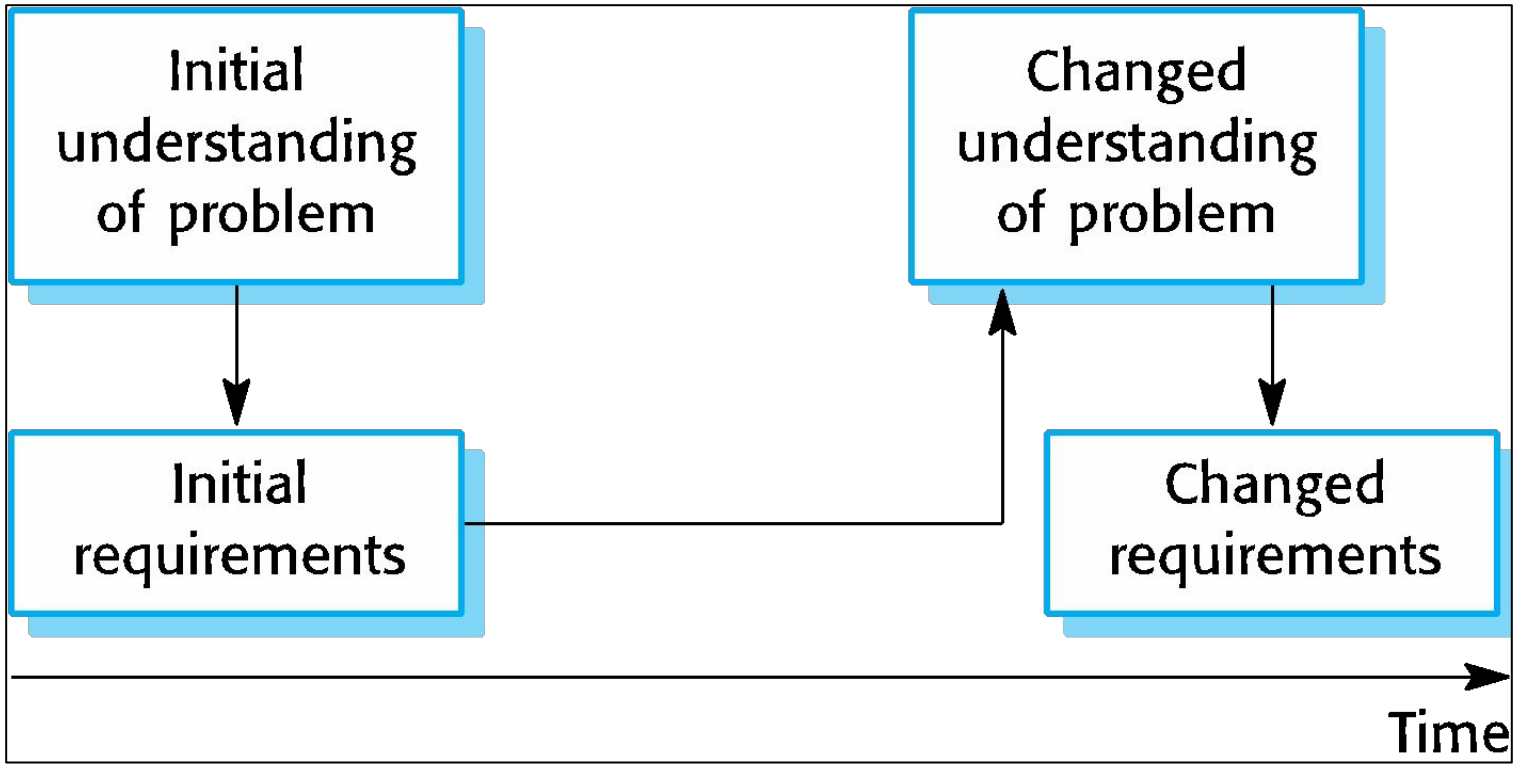
Changing Requirements (1/2)

- **Business and technical environment of the system always changes after installation.**
  - New hardware may be introduced, it may be necessary to interface the system with other systems, business priorities may change , and new legislation and regulations may be introduced.
- **People who pay for a system and the users of that system are rarely the same people.**
  - System customers impose requirements because of organizational and budgetary constraints. These may conflict with end-user requirements and, after delivery, new features may have to be added for user support if the system is to meet its goals.

Changing Requirements (2/2)

- **Large systems usually have a diverse user community, with many users having different requirements and priorities that may be conflicting or contradictory.**
  - The final system requirements are inevitably a compromise between them and, with experience, it is often discovered that the balance of support given to different users has to be changed.

Requirements Evolution



Requirements Management

- Requirements management is the process of managing changing requirements during the requirements engineering process and system development.
- New requirements emerge as a system is being developed and after it has gone into use.
- You need to keep track of individual requirements and maintain links between dependent requirements so that you can assess the impact of requirements changes. You need to establish a formal process for making change proposals and linking these to system requirements.

Requirements Management (1/4)

- **Establishes requirements management detail required.**
- **Requirements management decisions:**
  - Requirements identification Each requirement must be uniquely identified so that it can be cross-referenced with other requirements.
  - A change management process This is the set of activities that assess the impact and cost of changes. I discuss this process in more detail in the following section.
  - Traceability policies These policies define the relationships between each requirement and between the requirements and the system design that should be recorded.

Requirements Management (2/4)

- **Requirements management decisions:**
  - Tool support Tools that may be used range from specialist requirements management systems to spreadsheets and simple database systems.

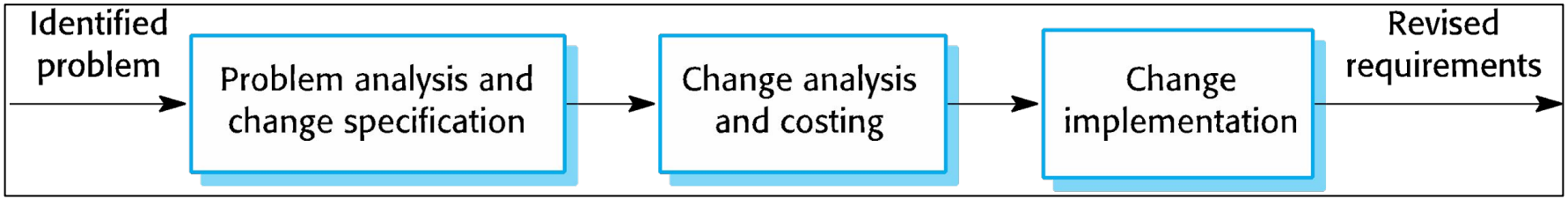
Requirements Management (3/4)

- **Deciding if a requirements change should be accepted**
  - Problem analysis and change specification
    - During this stage, the problem or the change proposal is analyzed to check that it is valid. This analysis is fed back to the change requestor who may respond with a more specific requirements change proposal, or decide to withdraw the request.
  - Change analysis and costing
    - The effect of the proposed change is assessed using traceability information and general knowledge of the system requirements. Once this analysis is completed, a decision is made whether or not to proceed with the requirements change.

Requirements Management (4/4)

- **Deciding if a requirements change should be accepted**
  - Change implementation
    - The requirements document and, where necessary, the system design and implementation, are modified. Ideally, the document should be organized so that changes can be easily implemented.

Requirements Change Management



Summary (1/5)

- 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 (2/5)

- The requirements engineering process is an iterative process that includes requirements elicitation, specification and validation.
- Requirements elicitation is an iterative process that can be represented as a spiral of activities – requirements discovery, requirements classification and organization, requirements negotiation and requirements documentation.
- You can use a range of techniques for requirements elicitation including interviews and ethnography. User stories and scenarios may be used to facilitate discussions.

Summary (3/5)

- Requirements specification is the process of formally documenting the user and system requirements and creating a software requirements document.
- 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.

Summary (4/5)

- Requirements validation is the process of checking the requirements for validity, consistency, completeness, realism and verifiability.
- Business, organizational and technical changes inevitably lead to changes to the requirements for a software system. Requirements management is the process of managing and controlling these changes.

Summary (5/5)

- Requirements Specification
- Requirements Specification Notations
- Software Requirements Specification Document
- Requirements Prioritisation
- Requirements Validation
- Requirements Change

Q & A Session

- Comments, concerns and complaints?

ict3020@unza.zm
https://bit.ly/3uI5u5V
http://bit.ly/2kK2ZkA

Bibliography

[1] Wiegers, K., & Beatty, J. (2013). Software requirements. Pearson Education.

[2] Sommerville, I. (2011). Software engineering 9th Edition. ISBN-10, 137035152.

[3] Kotonya, G., & Sommerville, I. (1998). Requirements engineering: processes and techniques. Wiley Publishing.

[4] Sommerville, I., & Sawyer, P. (1997). Requirements engineering: a good practice guide. John Wiley & Sons, Inc.

ICT 3020 (2022/23)
Fundamentals of Software Engineering
Lecture 6: Requirements Engineering: Requirements Specification

Lighton Phiri <lighton.phiri@unza.zm>
Department of Library & Information Science
University of Zambia
https://bit.ly/3uI5u5V