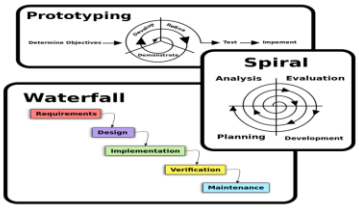
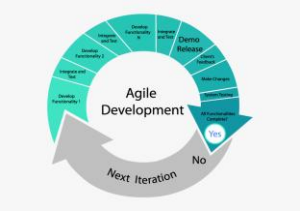
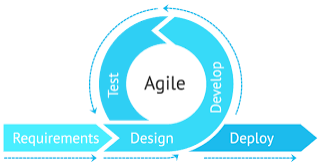


Software Engineering And Testing

Requirements Analysis & Specification





Outline

- Requirements Gathering and Analysis
  - Requirement Gathering
  - Requirement Analysis
- Software Requirements Specification (SRS)
  - Characteristics of a Good SRS Document
  - Attributes of Bad SRS Documents
  - Important Categories of Customer Requirements
  - Functional Requirements
  - How to Identify the Functional Requirements
  - How to Document the Functional Requirements
- Formal System Specification.

Introduction

- Understanding the requirements of a problem is among the most difficult tasks that face a software engineer.
- When you first think about it, developing a clear understanding of requirements doesn't seem that hard. As we think - the customer knows what is required, he has a good understanding of the features and functions that will provide benefit
- And even if customers and end-users are explicit in their needs, those needs will change throughout the project.
- A customer walks into your office, sits down, looks you straight in the eye, and says,
- "I know you think you understand what I said, but what you don't understand is what I said is not what I meant."

3

Requirement Analysis is Hard

"The **hardest** single **part** of building a software system is **deciding what to build**. No part of the work so cripples the resulting system if done wrong."

"Fred" Brooks Jr. is an American computer architect, software engineer, and computer scientist, best known for managing the development of IBM's System/360 family of computers

"The **seeds** of major software **disasters** are usually **sown** in the **first three months** of commencing the software project."

Capers Jones is an American specialist in software engineering methodologies

4

Requirement Analysis is Hard

What is Requirement Engineering?

Tasks and **techniques** that lead to an **understanding** of **requirements** is called **requirement engineering**.

5

Requirement Engineering

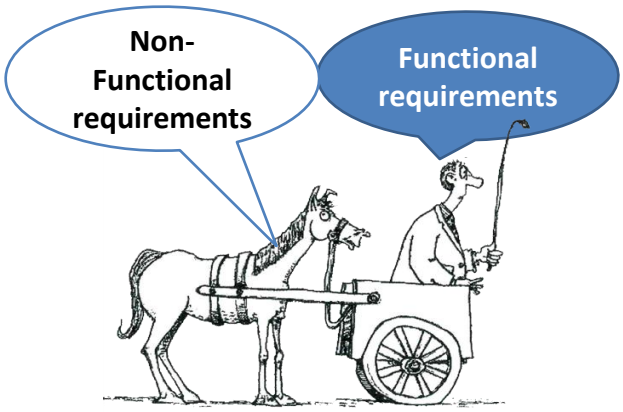
- Requirements engineering builds a bridge to design and construction.
- **A Requirement is:**

  A capability or condition required from the system.
- Requirements engineering provides the appropriate mechanism for understanding
  - what the customer wants,
  - analyzing need,
  - assessing feasibility,
  - negotiating a reasonable solution,
  - specifying the solution unambiguously,
  - validating the specification, and
  - managing the requirements as they are transformed into an operational system

6

Requirement Falls in two types

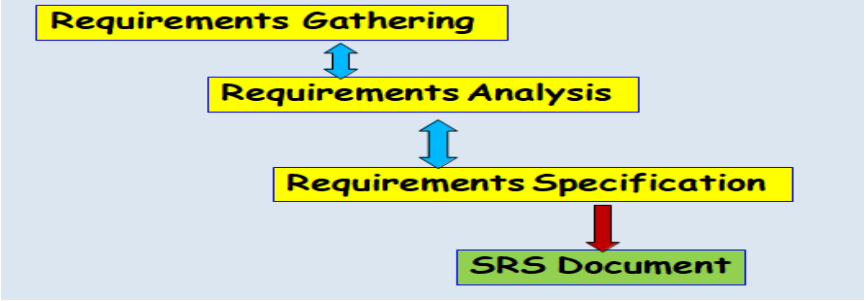
Functional requirements	Non-Functional requirements



Don't put what you want to do - before how you need to do it

7

Activities in Requirements Analysis and Specification



- **Requirements Gathering:** Fully understand the user requirements.
- **Requirements Analysis:** Remove inconsistencies, anomalies, etc. from requirements.
- **Requirements Specification:** Document requirements properly in an SRS document.

8

Requirement Gathering

- Requirements gathering activity is also popularly known as *requirements elicitation.*
- *The* primary objective of the requirements gathering task is to collect the requirements from the *stakeholders.*
- Theimportant ways in which an experienced analyst gathers requirements:

1. **Studying existing documentation**
2. **Interview**
3. **Task analysis**
4. **Scenario analysis**
5. **Form analysis :** In form analysis, the exiting forms and the formats of the notifications produced are analysed to determine the data input to the system and the data that are output from the system

9

Requirement Gathering

- **Scenario analysis :** task can have many scenarios of operation. The different scenarios of a task may take place when the task is invoked under different situations.
- For different scenarios of a task, the behaviour of the software can be different.
- For example, the possible scenarios for the book issue task of a library automation software may be:
  - Book is issued successfully to the member and the book issue slip is printed.
  - The book is reserved, and hence cannot be issued to the member.
  - The maximum number of books that can be issued to the member is already reached, and no more books can be issued to the member.

10

Requirement Analysis

- The main purpose of the requirements analysis activity is to analyse the gathered requirements to remove all ambiguities, incompleteness, and inconsistencies from the gathered customer requirements and to obtain a clear understanding of the software to be developed.

11

Requirement Analysis

- The following basic questions pertaining to the project should be clearly understood by the analyst before carrying out analysis:
  - What is the problem?
  - Why is it important to solve the problem?
  - What exactly are the data input to the system and what exactly are the data output by the system?
  - What are the possible procedures that need to be followed to solve the problem?
  - What are the likely complexities that might arise while solving the problem?
  - If there are external software or hardware with which the developed software has to interface, then what should be the data interchange formats with the external systems?

12

Requirement Analysis

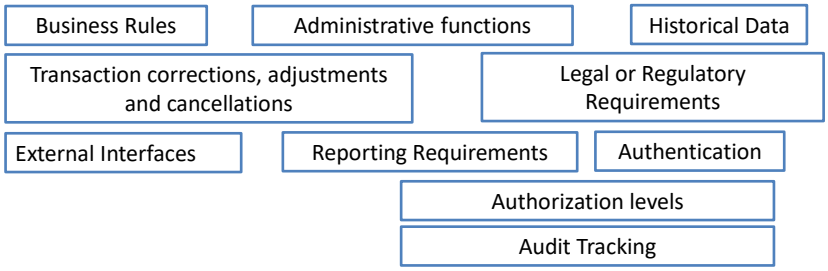
- During requirements analysis, the analyst needs to identify and resolve three main types of problems in the requirements:
  - **Anomaly** : When the temperature becomes high, the heater should be switched off.
  - **Inconsistency** : The furnace should be switched-off when the temperature of the furnace rises above 500°C.
  - When the temperature of the furnace rises above 500°C, the water shower should be switched- on and the furnace should remain on.
  - **Incompleteness** : In a chemical plant automation software, suppose one of the requirements is that if the internal temperature of the reactor exceeds 200°C then an alarm bell must be sounded. However, on an examination of all requirements, it was found that there is no provision for resetting the alarm bell after the temperature has been brought down in any of the requirements.

13

Functional Requirement

- Any requirement which specifies **what the system Should do**.
- A functional requirement will describe a particular behaviour of function of the system when certain conditions are met, for example: “Send email when a new customer signs up” or “Open a new account”.

Typical functional requirements



14

Non-Functional Requirement

- Any requirement which specifies **how the system performs a certain function**.
- A non-functional requirement will describe how a system should behave and what limits there are on its functionality.

Typical non-functional requirements

Response time	Availability	Regulatory
Throughput	Reliability	Manageability
Utilization	Recoverability	Environmental
Static volumetric	Maintainability	Data Integrity
Scalability	Serviceability	Usability
Capacity	Security	Interoperability

15

Library Management System

- **Functional Requirement:**
  - **Add Article:** New entries must be entered in database
  - **Update Article:** Any changes in articles should be updated in case of update
  - **Delete Article:** Wrong entry must be removed from system
  - **Inquiry Members:** Inquiry all current enrolled members to view their details
  - **Inquiry Issuance:** Inquiry all database articles
  - **Check out Article:** To issue any article must be checked out
  - **Check In article:** After receiving any article system will reenter article by Checking
  - **Inquiry waiting for approvals:** Librarian will generates all newly application which is in waiting list
  - **Reserve Article:** This use case is used to reserve any book with the name of librarian, it can be pledged

16

Library Management System

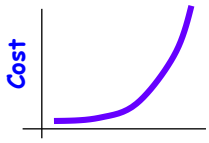
- **Non-Functional Requirement:**
  - **Safety Requirements:** The database may get crashed at any certain time due to virus or operating system failure. So, it is required to take the database backup.
  - **Security Requirements:** We are going to develop a secured database for the university. There are different categories of users namely teaching staff, administrator, library staff, students etc., Depending upon the category of user the access rights are decided.
  - **Software Constraints:** The development of the system will be constrained by the availability of required software such as database and development tools. The availability of these tools will be governed by.

17

Requirements Analysis and Specification

- **Good SRS reduces development cost:**
  - Req. errors are expensive to fix later
  - Req. changes cost a lot (typically 40% of requirements change later)
  - Good SRS can minimize changes and errors
  - Substantial savings — effort spent during req. saves multiple times that effort
- **An Example:**
  - Cost of fixing errors in req., design, coding, acceptance testing and operation increases exponentially

Need for SRS...



18

SRS (Software Requirements Specification)

Software Requirement Specification (SRS) is a document that completely describes what the proposed software should do without describing how software will do it.

- SRS is also helping the clients to understand their own needs.

SRS Contains

- A complete information description
- A detailed functional description
- A representation of system behaviour
- An indication of performance requirements and design constraints
- Appropriate validation criteria
- Other information suitable to requirements

19

What are the Uses of an SRS Document?

- Establishes the basis for agreement between the customers and the suppliers
- Forms the starting point for development.
- Provide a basis for estimating costs and schedules.
- Provide a basis for validation and verification.
- Provide a basis for user manual preparation.
- Serves as a basis for later enhancements.
- **SRS document concentrates on:**
  - What needs to be done in terms of input-output behaviour
  - Carefully avoids the solution ("how to do") aspects.

20

Characteristics of Good SRS

- SRS should be **accurate, complete, efficient, and of high quality,** so that it does not affect the entire project plan.
- An SRS is **said to be of high quality when** the developer and user **easily understand** the prepared document.
- Characteristics of a Good SRS:

Correct

SRS is correct when all user requirements are stated in the requirements document.

Note that there is no specified tool or procedure to assure the correctness of SRS.

Unambiguous

SRS is unambiguous when every stated requirement has only one interpretation.

Complete

SRS is complete when the requirements clearly define what the software is required to do.

21

Characteristics of Good SRS

Ranked for Importance/Stability

All requirements are not equally important, hence each requirement is identified to make differences among other requirements.

Stability implies the probability of changes in the requirement in future.

Modifiable

The requirements of the user can change, hence requirements document should be created in such a manner that those changes can be modified easily.

Traceable

SRS is traceable when the source of each requirement is clear and facilitates the reference of each requirement in future.

Verifiable

SRS is verifiable when the specified requirements can be verified with a cost-effective process to check whether the final software meets those requirements.

Consistent

SRS is consistent when the subsets of individual requirements defined do not conflict with each other.

22

Problems without SRS

- Without developing the SRS document, the system would not be properly implemented according to customer needs.
- Software developers would not know whether what they are developing is what exactly is required by the customer.
- Without SRS, it will be very difficult for the maintenance engineers to understand the functionality of the system.
- It will be very difficult for user document writers to write the users' manuals properly without understanding the SRS.

23

Standard Template for writing SRS

1. Introduction
- 1.1 Purpose
- 1.2 Document Conventions
- 1.3 Intended Audience and Reading Suggestions
- 1.4 Project Scope
- 1.5 References

2. Overall Description
- 2.1 Product Perspective
- 2.2 Product Features
- 2.3 User Classes and Characteristics
- 2.4 Operating Environment
- 2.5 Design and Implementation Constraints
- 2.6 User Documentation
- 2.7 Assumptions and Dependencies

3. System Features
- 3.1 System Feature 1
- 3.2 System Feature 2 (and so on)

4. External Interface Requirements
- 4.1 User Interfaces
- 4.2 Hardware Interfaces
- 4.3 Software Interfaces
- 4.4 Communications Interfaces

5. Other Nonfunctional Requirements
- 5.1 Performance Requirements
- 5.2 Safety Requirements
- 5.3 Security Requirements
- 5.4 Software Quality Attributes

6. Other Requirements

Appendix A: Glossary | Appendix B: Analysis Models | Appendix C: Issues List

24

SRS

- **Four important parts:**
  - Functional requirements,
  - External Interfaces
  - Non-functional requirements,
  - Constraints

25

Functional Requirements

- **Specifies all the functionality that the system should support**
  - Heart of the SRS document:
  - Forms the bulk of the Document
- **Outputs for the given inputs and the relationship between them**
- **Must specify behavior for invalid inputs too!**

26

Functional Requirements Document

- **Overview :** describe purpose of the function and the approaches and techniques employed
- **Inputs and Outputs :**
  - sources of inputs and destination of outputs
  - quantities, units of measure, ranges of valid inputs and outputs
  - Timing
- **Processing :**
  - validation of input data
  - exact sequence of operations
  - responses to abnormal situations
  - any methods (eg. equations, algorithms) to be used to transform inputs to outputs

27

Nonfunctional Requirements

- **Characteristics of the system which can not be expressed as functions:**
  - Maintainability,
  - Portability,
  - Usability,
  - Security,
  - Safety, etc.
- **Reliability issues**
- **Performance issues:**
  - **Example:** How fast can the system produce results?
    - At a rate that does not overload another system to which it supplies data, etc.
    - Response time should be less than 1sec 90% of the time
    - Needs to be measurable (verifiability)

28

Constraints

- Hardware to be used,
- Operating system
  - or DBMS to be used
- Capabilities of I/O devices
- Standards compliance
- Data representations by the interfaced system

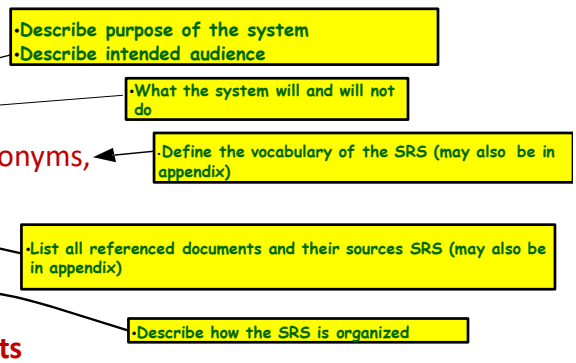
External Interface Requirement

- User interfaces
- Hardware interfaces
- Software interfaces
- Communications interfaces with other systems
- File export formats

29

IEEE 830-1998 Standard for SRS

- Title
- Table of Contents
- 1. Introduction
  - 1.1 Purpose
  - 1.2 Scope
  - 1.3 Definitions. Acronyms, and Abbreviations
  - 1.4 References
  - 1.5 Overview
- 2. Overall Description
- 3. Specific Requirements
- Appendices
- Index



30

IEEE 830-1998 Standard – Section 2 of SRS

- Title
- Table of Contents
- 1. Introduction
- 2. Overall Description
  - 2.1 Product Perspective
  - 2.2 Product Functions
  - 2.3 User Characteristics
  - 2.4 Constraints
- 3. Specific Requirements
- 4. Appendices
- 5. Index

- Present the business case and operational concept of the system
- Describe external interfaces: system, user, hardware, software, communication
- Describe constraints: memory, operational, site adaptation

- Summarize the major functional capabilities

- Describe technical skills of each user class

- Describe other constraints that will limit developer's options; e.g., regulatory policies; target platform, database, network, development standards requirements

31

IEEE 830-1998 Standard – Section 3 of SRS (1)

- ....
- 1. Introduction
- 2. Overall Description
- 3. Specific Requirements
  - 3.1 External Interfaces
  - 3.2 Functions
  - 3.3 Performance Requirements
  - 3.4 Logical Database Requirements
  - 3.5 Design Constraints
  - 3.6 Software System Quality Attributes
  - 3.7 Object Oriented Models
- 4. Appendices
- 5. Index

Specify software requirements in sufficient detail so that designers can design the system and testers can verify whether requirements met.

State requirements that are externally perceivable by users, operators, or externally connected systems

Requirements should include, at the least, a description of every input (stimulus) into the system, every output (response) from the system, and all functions performed by the system in response to an input

32

IEEE 830-1998 Standard – Templates

- **Section 3 (Specific Requirements)can be organized in several different ways based on**
  - Modes
  - User classes
  - Concepts (object/class)
  - Features
  - Stimuli

33

Example Section 3 of SRS of Academic Administration Software

- **SPECIFIC REQUIREMENTS**

1. Functional Requirements

1. Subject Registration

- The subject registration requirements are concerned with functions regarding subject registration which includes students selecting, adding, dropping, and changing a subject.

- **F-001:** The system shall allow a student to register a subject.
- **F-002:** It shall allow a student to drop a course.
- **F-003:** It shall support checking how many students have already registered for a course.

34

Design Constraints (3.2)

- **3.2 Design Constraints**
- **C-001:** AAS shall provide user interface through standard web browsers.
- **C-002:** AAS shall use an open source RDBMS such as Postgres SQL.
- **C-003:** AAS shall be developed using the JAVA programming language

35

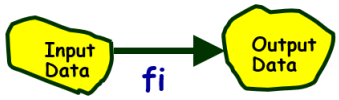
Non-functional Requirement

- **3.3 Non-Functional Requirements**
- **N-001:** AAS shall respond to query in less than 5 seconds.
- **N-002:** AAS shall operate with zero down time.
- **N-003:** AAS shall allow upto 100 users to remotely connect to the system.
- **N-004:** The system will be accompanied by a well-written user manual.

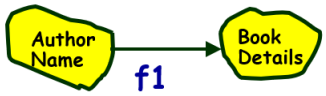
36

Functional Requirement

- **It is desirable to consider every system as:** Performing a set of functions {fi}.
- **Each function fi considered as:** Transforming a set of input data to corresponding output data.



- Example : **F1: Search Book**
  - Input:
    - an author's name:
  - Output:
    - details of the author's books and the locations of these books in the library.



37

Functional Requirement

- **Functional requirements describe:**
  - A set of high-level requirements
  - Each high-level requirement:
    - takes in some data from the user
    - outputs some data to the user
  - Each high-level requirement:
    - might consist of a set of identifiable sub-functions
- **For each high-level requirement:**
  - A function is described in terms of:
    - Input data set
    - Output data set
    - Processing required to obtain the output data set from the input data set.

38

Is it a Functional Requirement?

- **A high-level function is one:** Using which the user can get some useful piece of work done.
- **Can the receipt printing work during withdrawal of money from an ATM:** Be called a functional requirement?
- **A high-level requirement typically involves:**
  - Accepting some data from the user,
  - Transforming it to the required response, and then
  - Outputting the system response to the user.

39

Use Cases

- **A use case is a term in UML:** Represents a high level functional requirement.
- **Use case representation is more well-defined and has agreed documentation:**
  - Compared to a high-level functional requirement and its documentation
  - Therefore many organizations document the functional requirements in terms of use cases

40

Example of functional requirements

- **Req. 1:** Once user selects the "search" option,
  - he is asked to enter the key words.
  - The system should output details of all books
    - whose title or author name matches any of the key words entered.
    - Details include: Title, Author Name, Publisher name, Year of Publication, ISBN Number, Catalog Number, Location in the Library.
- **Req. 2:** When the "renew" option is selected,
  - The user is asked to enter his membership number and password.
  - After password validation,
    - The list of the books borrowed by him are displayed.
    - The user can renew any of the books: By clicking in the corresponding renew box.

41

Examples of Bad SRS Documents

- **Unstructured Specifications:**
  - Narrative essay — one of the worst types of specification document:
    - Difficult to change,
    - Difficult to be precise,
    - Difficult to be unambiguous,
    - Scope for contradictions, etc.
- **Noise:** Presence of text containing information irrelevant to the problem.
- **Silence:** Aspects important to proper solution of the problem are omitted.
- **Contradictions:** Contradictions might arise if the same thing described at several places in different ways.
- **Ambiguity:**
  - Literary expressions
  - Unquantifiable aspects, e.g. "good user interface"

42

Examples of Bad SRS Documents

- **Overspecification:** Addressing "how to" aspects
  - For example, "Library member names should be stored in a sorted descending order"
  - Overspecification restricts the solution space for the designer.
- **Forward References:** References to aspects of problem
  - defined only later on in the text.
- **Wishful Thinking:** Descriptions of aspects
  - for which realistic solutions will be hard to find.

43

Suggestions for Writing Good quality Requirements

- Keep sentences and paragraphs short.
- Use active voice.
- Use proper grammar, spelling, and punctuation.
- Use terms consistently and define them in a glossary.
- To see if a requirement statement is sufficiently well defined,
  - Read it from the developer's perspective
- Split a requirement into multiple sub-requirements:
  - Because each will require separate test cases and because each should be separately traceable.
  - If several requirements are strung together in a paragraph, it is easy to overlook one during construction or testing.

44

Formal Specifications

- **A formal specification technique is a mathematical method to:**
  - Accurately specify a system.
  - Verify that implementation satisfies specification.
  - Prove properties of the specification.
- **Mathematical techniques used include:**
  - Logic-based
  - set theoretic
  - algebraic specification
  - finite state machines, etc.
- **Advantages:**
  - Well-defined semantics, no scope for ambiguity
  - Automated tools can check properties of specifications
  - Executable specification
- **Disadvantages of formal specification techniques:**
  - Difficult to learn and use
  - Not able to handle complex systems

45

46