Requirements Analysis and Specification

Requirements Analysis and Specification

- Many projects fail:
  - because they start implementing the system without determining whether they are building what the customer really wants.

Requirements Analysis and Specification

- Goals of requirements analysis and specification phase:
  - fully understand the user requirements
  - remove inconsistencies, anomalies, etc. from requirements
  - document requirements properly in an SRS document

Requirements Analysis and Specification

- Consists of two distinct activities:
  - Requirements Gathering and Analysis
  - Specification

Requirements Analysis and Specification

- Systems analyst who undertakes requirements analysis and specification:
  - collects data pertaining to the product
  - analyzes collected data:
    - to understand what exactly needs to be done.
  - writes the Software Requirements Specification (SRS) document.

Requirements Analysis and Specification

- Final output of this phase:
  - Software Requirements Specification (SRS) Document.
  - The SRS document is reviewed by the customer.
  - reviewed SRS document forms the basis of all future development activities.

Requirements Analysis

- Requirements analysis consists of two main activities:
  - Requirements gathering
  - Analysis of the gathered requirements

Requirements Analysis

- Analyst gathers requirements through:
  - observation of existing systems
  - studying existing procedures
  - discussion with the customer and end-users
  - analysis of what needs to be done, etc.

Requirements Gathering

- If the project is to automate some existing procedures
  - e.g., automating existing manual accounting activities,
  - the task of the system analyst is a little easier
  - analyst can immediately obtain:
    - input and output formats
    - accurate details of the operational procedures

Requirements Gathering (CONT.)

- In the absence of a working system,
  - lot of imagination and creativity are required.
- Interacting with the customer to gather relevant data:
  - requires a lot of experience.

Requirements Gathering (CONT.)

- Some desirable attributes of a good system analyst:
  - Good interaction skills
  - imagination and creativity
  - experience

Analysis of the Gathered Requirements

- After gathering all the requirements:
  - analyze it:
    - Clearly understand the user requirements
    - Detect inconsistencies, ambiguities and incompleteness.
- Incompleteness and inconsistencies:
  - resolved through further discussions with the end-users and the customers.

Inconsistent requirement

- Some part of the requirement:
  - contradicts with some other part.
- Example:
  - One customer says turn off heater and open water shower when temperature > 100 C
  - Another customer says turn off heater and turn ON cooler when temperature > 100 C

Incomplete requirement

- Some requirements have been omitted:
  - due to oversight.
- Example:
  - The analyst has not recorded: when temperature falls below 90 C
    - heater should be turned ON
    - water shower turned OFF.

Analysis of the Gathered Requirements

(CONT.)

- Requirements analysis involves:
  - obtaining a clear, in-depth understanding of the product to be developed
  - remove all ambiguities and inconsistencies from the initial customer perception of the problem.

Analysis of the Gathered Requirements

(CONT.)

- It is quite difficult to obtain:
  - a clear, in-depth understanding of the problem:
    - especially if there is no working model of the problem.

Analysis of the Gathered Requirements
(CONT.)

- Experienced analysts take considerable time:
  - to understand the exact requirements the customer has in his mind.

Analysis of the Gathered Requirements(CONT.)

- Several things about the project should be clearly understood by the analyst:
  - What is the problem?
  - Why is it important to solve the problem?
  - What are the possible solutions to the problem?
  - What complexities might arise while solving the problem?

Software Requirements Specification

- Main aim of requirements specification:
  - systematically organize the requirements arrived during requirements analysis
  - document requirements properly.

Software Requirements Specification

- The SRS document is useful in various contexts:
  - statement of user needs
  - contract document
  - reference document
  - definition for implementation

Software Requirements Specification: A Contract Document

- Requirements document is a reference document.
- SRS document is a contract between the development team and the customer.
- Once the SRS document is approved by the customer,
  - development team starts to develop the product according to the requirements recorded in the SRS document.

SRS Document (CONT.)

- The SRS document is known as <u>black-box specification:</u>
  - the system is considered as a black box whose internal details are not known.
  - only its visible external (i.e. input/output) behavior is documented.

SRS Document (CONT.)

- SRS document concentrates on:
  - what needs to be done
  - carefully avoids the solution ("how to do") aspects.

Properties of a good SRS document

- It should be concise
  - and at the same time should not be ambiguous.
- It should specify what the system must do
  - and not say how to do it.
- Easy to change
  - i.e. it should be well-structured.
- It should be consistent.
- It should be complete.

Properties of a good SRS document (cont...)

- It should be traceable
  - you should be able to trace which part of the specification corresponds to which part of the design and code, etc and vice versa.
- It should be verifiable
  - e.g. “system should be user friendly” is not verifiable

SRS Document (CONT.)

- SRS document, normally contains three important parts:
  - functional requirements,
  - nonfunctional requirements,
  - constraints on the system.

Functional Requirements

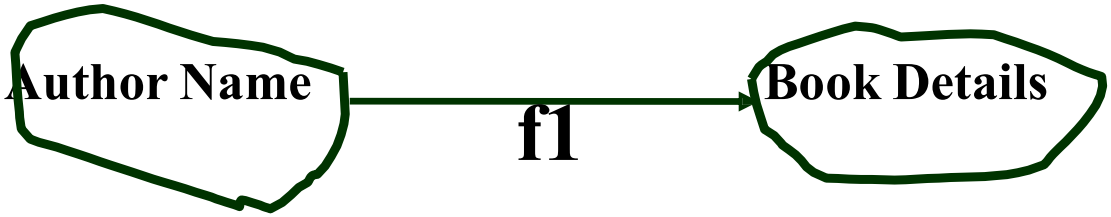
- Functional requirements describe:
  - Functionality required by the users from the system.
  - A set of **high-level** requirements
  - Each high-level requirement:
    - takes in some data from the user
    - outputs some data to the user
  - Each high-level requirement:
    - might consist of a set of identifiable functions

Functional Requirements

- For each high-level requirement:
  - every 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

Example: Functional Requirement

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



Functional Requirements

- How to identify the functional requirements?
  - Need to be identified either from an informal problem description document or from a conceptual understanding of the problem.
  - Services expected from the users.
  - To decide whether is it high level or not?
    - Decide the scope.

Example Functional Requirements (LMS)

- **Req. 1:Search Book**
  - Once the 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.

Example Functional Requirements

- **Req. 2:Renew Book**
  - 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.

- **How to document functional requirement along with its sub-Requirements?**

Functional Requirement R1: Search Book

- R1.1: Select Search Option
  - Input: "search" option,
  - Output: user prompted to enter the key words.
- R1.2:Input Search Details
  - Input: key words
  - Output: Details of all books whose title or author name matches any of the key words.
    - Details include: Title, Author Name, Publisher name, Year of Publication, ISBN Number, Catalog Number, Location in the Library.
  - Processing: Search the book list for the keywords

Functional Req. 2: Renew Book

- R2.1: Select option with username and password
  - Input: “renew” option selected,
  - Output: user prompted to enter his membership number and password.
- R2.2::Validate Username and Password
  - Input: membership number and password
  - Output:
    - list of the books borrowed by user are displayed. User prompted to enter books to be renewed or
    - user informed about bad password
  - Processing: Password validation, search books issued to the user from borrower list and display[3.4]

Functional Req. 2 (Cont.)

- R2.3:Renew based on user choice
  - Input: user choice for renewal of the books issued to him through mouse clicks in the corresponding renew box.
  - Output: Confirmation of the books renewed
  - Processing: Renew the books selected by the in the borrower list.

Example: Withdraw Cash from ATM

R1: withdraw cash

Description:

The withdraw cash function first determines the type of account that the user has and the account number from which the user wishes to withdraw cash. It checks the balance to determine whether the requested amount is available in the account. If enough balance is available, it outputs the required cash, otherwise it generates an error message.

Example: Withdraw Cash from ATM

R1.1 select withdraw amount option

Input: “withdraw amount” option

Output: user prompted to enter the account type

R1.2: select account type

Input: user option

Output: prompt to enter amount

Example: Withdraw Cash from ATM

R1.3: get required amount

Input: amount to be withdrawn in integer values greater than 100 and less than10,000 in multiples of 100.

Output: The requested cash and printed transaction statement.

Processing: the amount is debited from the user's account if sufficient balance is available, otherwise an error message displayed.

Nonfunctional Requirements

- Characteristics of the system which **can not be expressed as functions**:
  - maintainability
  - portability
  - Usability
  - Maximum number of concurrent users, etc.

Nonfunctional Requirements

- Nonfunctional requirements include:
  - performance issues
  - human-computer interface issues
  - Interface with other external systems
  - security
  - Timing
  - Throughput, etc.

Nonfunctional Requirements

- response time
- transaction rates
- A website should be capable enough to handle 20 million users with affecting its performance
- The software should be portable. So moving from one OS to other OS does not create any problem.

Constraints

- Constraints describe things that the system should or should not do.
  - For example,
    - standards compliance
    - how fast the system can produce results
      - so that it does not overload another system to which it supplies data, etc.

Constraints

- The system will work on our existing technical infrastructure - no new technologies will be introduced.

Organization of the SRS Document

- Introduction
- Goals of Implementation
- Functional Requirements
- Nonfunctional Requirements
- Behavioural Description

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.

Examples of Bad SRS Documents

- Noise:
  - Presence of text containing information irrelevant to the problem.
- Silence:
  - aspects important to proper solution of the problem are omitted.
- Ambiguity:
  - Literary expressions
  - Unquantifiable aspects, e.g. “good user interface”

Examples of Bad SRS Documents

- Over-specification:
  - Addressing "how to" aspects
    - For example, "Library member names should be stored in a sorted descending order"
  - Over-specification restricts the solution space for the designer.
- Contradictions:
  - Contradictions might arise
    - if the same thing described at several places in different ways.

Requirement Engineering

- Inception—ask a set of questions that establish …
  - basic understanding of the problem
  - the people who want a solution
  - the nature of the solution that is desired, and
  - the effectiveness of preliminary communication and collaboration between the customer and the developer
- Elicitation—elicit requirements from all stakeholders
- Elaboration—create an analysis model that identifies data, function and behavioral requirements
- Negotiation—agree on a deliverable system that is realistic for developers and customers

Requirement Engineering

- Specification—can be any one (or more) of the following:
  - A written document
  - A set of models
  - A formal mathematical
  - A collection of user scenarios (use-cases)
  - A prototype
- Validation—a review mechanism that looks for
  - errors in content or interpretation
  - areas where clarification may be required
  - missing information
  - inconsistencies (a major problem when large products or systems are engineered)
  - conflicting or unrealistic (unachievable) requirements.
- Requirements management

Requirements Engineering

- **Inception:** ask a set of questions that establish.
  - basic understanding of the problem
  - the people who want a solution
  - the nature of the solution that is desired
  - the effectiveness of preliminary communication and collaboration between the customer and the developer

Requirements Engineering

- **The questions**
  - Who is behind the request for this work?
  - Who will use the solution?
  - What will be the economic benefit of a successful solution?
  - Is there another source for the solution that you need?

Requirements Engineering

- **Elicitation:** elicit requirements from all stakeholders.
- The goal is:
  - to identify the problem
  - propose elements of the solution
  - negotiate different approaches
  - specify a preliminary set of solution requirements

Requirements Engineering

- **Elaboration:** create an **analysis model** that identifies data, function and behavioral requirements
  - **Scenario-based elements**
    - Functional: processing narratives for software functions
    - Use-case: descriptions of the interaction between an "actor" and the system
  - **Class-based elements**
    - Implied by scenarios

Requirements Engineering

- **Behavioral elements**
  - State diagram
- **Flow-oriented elements**
  - Data flow diagram
- **Negotiation:** agree on a deliverable system that is realistic for developers and customers

Requirements Engineering

- **Negotiation:**
- Identify the key stakeholders!
- These are the people who will be involved in the negotiation!
- Determine each of the stakeholders “win
- conditions”!
- Win conditions are not always obvious!
- Negotiate!
- Work toward a set of requirements that lead to “winwin

Requirements Engineering

- **Specification:** can be any one (or more) of the following:
  - A written document
  - A set of models
  - A formal mathematical
  - A collection of user scenarios (use-cases)
  - A prototype

Requirements Engineering

- **Validation:** a review mechanism that looks for
  - errors in content or interpretation
  - areas where clarification may be required
  - missing information
  - inconsistencies (a major problem when large products or systems are engineered)
  - conflicting or unrealistic (unachievable) requirements.
- **Requirements management**

Validating Requirements

- **Is each requirement consistent** with the overall objective for the system/product?
- Have all requirements been specified at the **proper level of abstraction**? That is, do some requirements provide a level of technical detail that is inappropriate at this stage.
- Is the requirement **really necessary or does it represent an add-on feature** that may not be essential to the objective of the system?

Validating Requirements

- Is each requirement bounded and unambiguous?
- Do any requirements conflict with other requirements?
- Is each requirement achievable in the **technical environment**?
- Is each requirement **testable**, once implemented?

User Story

- A **user story** is just a simple way of expressing requirements in a consistent format.

  As a_____, (stake holder)

  I want to_____, (task or function)

  So that_____. (Why?)

  Ex:

  As a customer,

  I want to be able to select a "pay now" option when I view my bill,

  So that I can pay the bill immediately.