Requirements Analysis and Specification

Acknowledgement: Fundamentals of Software Engineering (**Prof. Rajib Mall**)

Organization of this Lecture

- Introduction
- Requirements analysis
- Requirements specification
- SRS document
- Decision table
- Decision tree
- Summary

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

- It is important to learn:
  - Requirements analysis and specification techniques carefully.

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.

Who Carries Out Requirements Analysis and Specification?

- The person who undertakes requirements analysis and specification:
  - Known as **systems analyst**:
  - **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 Activities

1. Studying the existing documentation
2. Interview
3. Task/Scenario analysis
4. Form analysis

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.

Case Study: Automation of Office Work at SES

- The academic, inventory, and financial information at the SES department:
  - Being carried though manual processing by two office clerks, and two attendants.

- Considering the budget he had at his disposal:
  - The official entrusted the work to a team of student volunteers.

Case Study: Automation of Office Work at SES

- The team was first briefed by the office in-charge about the specific activities to be automated.

- The analyst first discussed with the two clerks:
  - Regarding their specific responsibilities (tasks) that were to be automated.

- The analyst also interviewed student and faculty representatives who would also use the software.

Case Study: Automation of Office Work at SES

- For each task, they asked:
  - About the steps through which these are performed.
  - They also discussed various scenarios that might arise for each task.
  - The analyst collected all types of forms that were being used.

Analysis of the Gathered Requirements

- Main purpose of requirements analysis:
  - Clearly understand the user requirements,
  - Detect inconsistencies, ambiguities, and incompleteness.

- Incompleteness and inconsistencies:
  - Resolved through further discussions with the end-users and the customers.

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.)

- Experienced systems analysts know - often as a result of painful experiences ---
  - Without a clear understanding of the problem, it is impossible to develop a **satisfactory** system.

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?

Analysis of the Gathered Requirements(CONT.)

- Some anomalies and inconsistencies can be very subtle:
  - Escape even most experienced eyes.
  - If a formal model of the system is constructed,
    - Many of the subtle anomalies and inconsistencies get detected.

Analysis of the Gathered Requirements(CONT.)

- After collecting all data regarding the system to be developed,
  - Remove all inconsistencies and anomalies from the requirements,
  - Systematically organize requirements into a Software Requirements Specification (SRS) document.

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,
    - Any subsequent controversies are settled by referring the SRS document.

Software Requirements Specification: A Contract Document

- Once customer agrees to the SRS document:
  - Development team starts to develop the product according to the requirements recorded in the SRS document.

- The final product will be acceptable to the customer:
  - As long as it satisfies **all** the requirements recorded in the SRS document.

SRS Document (CONT.)

- The SRS document is known as black-box specification:
  - 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.
- The SRS document serves as a contract
  - Between development team and the customer.
  - Should be carefully written

SRS Document (CONT.)

- The requirements at this stage:
  - Written using end-user terminology.

- If necessary:
  - Later a **formal requirement specification** may be developed from it.

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, code, etc and vice versa.
- It should be verifiable
  - e.g. “system should be user friendly” is not verifiable

SRS Document (CONT.)

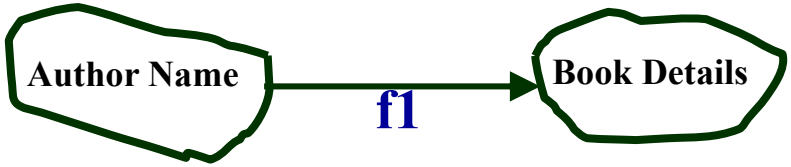
- SRS document, normally contains three important parts:
  - **Functional** requirements,
  - **Non-functional** requirements,
  - Goals of Implementation.

SRS Document (CONT.)

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

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

- 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 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.

Nonfunctional Requirements

- Characteristics of the system which can not be expressed as functions:
  - Maintainability,
  - Portability,
  - Usability, etc.

Nonfunctional Requirements

- Nonfunctional requirements include:
  - Reliability issues,
  - Performance issues:
    - Example: How fast the system can produce results
      - so that it does not overload another system to which it supplies data, etc.
  - Human-computer interface issues,
  - Interface with other external systems,
  - Security, maintainability, etc.

Non-Functional Requirements

- Hardware to be used,
- Operating system
  - or DBMS to be used
- Capabilities of I/O devices
- Standards compliance
- …….

Goals of Implementation

- Goals describe things that are desirable of the system:
  - But, would not be checked for compliance.
  - For example,
    - Reusability issues
    - Functionalities to be developed in future

Organization of the SRS Document

- Introduction.
- Functional Requirements
- Nonfunctional Requirements
  - External interface requirements
  - Performance requirements
- Goals of implementation

Functional Requirements

- A high-level function is one:
  - Using which the user can get some useful piece of work done.

- 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.

High-Level Function

- A high-level function:
  - Usually involves a series of interactions between the system and one or more users.
- Even for the same high-level function,
  - There can be different interaction sequences (or scenarios)
  - Due to users selecting different options or entering different data items.

Example Functional Requirements

- List all functional requirements
  - with **proper numbering**.
- Req. 1:
  - Once the user selects the "**search**" option,
    - he is asked to enter the keywords.
  - 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:
  - 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 the user are displayed.
  - The user can renew any of the books:
    - By clicking in the corresponding renew box.

Req. 1:

- R.1.1:
  - Input: “search” option,
  - Output: user prompted to enter the key words.
- R1.2:
  - 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

Req. 2:

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

Req. 2:

- R2.3:
  - 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 user in the borrower list.

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.

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.

- Contradictions:
  - Contradictions might arise
    - if the same thing described at several places in different ways.

Examples of Bad SRS Documents

- Ambiguity:
  - Unquantifiable aspects, e.g. “good user interface”
- Forward References:
  - References to aspects of problem
    - defined only later on in the text.

Representation of complex processing logic:

- Decision trees
- Decision tables

Decision Trees

- Decision trees:
  - **Edges** of a decision tree represent **conditions**
  - **Leaf nodes** represent **actions** to be performed.

- A decision tree gives a graphic view of:
  - Logic involved in decision making
  - Corresponding actions taken.

Example: LMS

- A Library Membership automation Software (LMS) should support the following three options:
  - New member,
  - Renewal,
  - Cancel membership.

Example: LMS

- When the new member option is selected,
  - The software asks details about the member:
    - name,
    - address,
    - phone number, etc.

Example(cont.)

- If proper information is entered,
  - A membership record for the member is created
  - A bill is printed for the annual membership charge plus the security deposit payable.

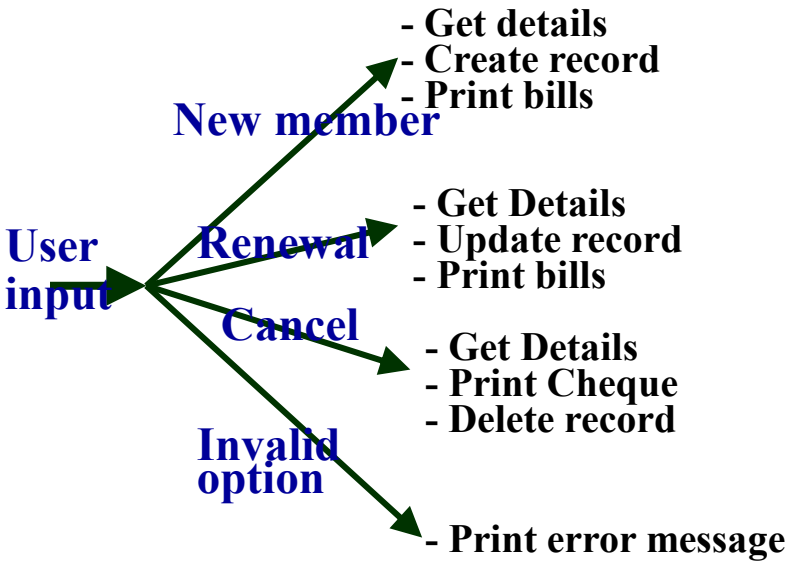
Example(cont.)

- If the renewal option is chosen,
  - LMS asks the member's name and his membership number
    - checks whether he is a valid member.
  - If the name represents a valid member,
    - the membership expiry date is updated and the annual membership bill is printed,
    - otherwise an error message is displayed.

Example(cont.)

- If the cancel membership option is selected and the name of a valid member is entered,
  - The membership is cancelled,
  - A cheque for the balance amount due to the member is printed
  - The membership record is deleted.

Decision Tree



Decision Table

- A decision table shows in a tabular form:
  - Processing logic and corresponding actions
- Upper rows of the table specify:
  - The conditions to be evaluated
- Lower rows specify:
  - The actions to be taken when the corresponding conditions are satisfied.

Decision Table

- In technical terminology,
  - a column of the table is called a rule:
  - A rule implies:
    - if a condition is true, then execute the corresponding action.

Example:

Conditions				
Valid selection	No	Yes	Yes	Yes
New member	-	Yes	No	No
Renewal	-	No	Yes	No
Cancellation	-	No	No	Yes
Actions				
Display error message	x	-	-	-
Ask member's details	-	x	-	-
Build customer record	-	x	-	-
Generate bill	-	x	x	-
Ask member's name & membership number	-	-	x	x
Update expiry date	-	-	x	-
Print cheque	-	-	-	x
Delete record	-	-	-	x

Comparison

- Both decision tables and decision trees
  - Can represent complex logic.
- Decision trees are easier to read and understand
  - When the number of conditions are small.
- Decision tables help to look at every possible combination of conditions.

Formal Specification

- A formal specification technique is a mathematical method to:
  - Accurately specify a system.
  - Verify that implementation satisfies specification.
  - Prove properties of the specification.

Formal Specification

- Advantages:
  - Well-defined semantics, **no scope for ambiguity**
  - **Automated** tools can check properties of specifications

Formal Specification

- Disadvantages of formal specification techniques:
  - Difficult to learn and use
  - Not able to handle complex systems

Formal Specification

- Mathematical techniques used include:
  - Logic-based
  - set theoretic
  - algebraic specification
  - finite state machines, etc.

Summary

- **Requirements analysis and specification**
  - An important phase of software development:
  - Any error in this phase would affect all subsequent phases of development.

- **Consists of two different activities**:
  - Requirements gathering and analysis
  - Requirements specification

Summary

- The aims of requirements analysis:
  - Gather all user requirements
  - Clearly understand exact user requirements
  - Remove inconsistencies and incompleteness.
- The goal of specification:
  - Systematically organize requirements
  - Document the requirements in an SRS document.

Summary

- Main components of SRS document:
  - Functional requirements
  - Non-functional requirements
  - Goals of implementation/ constraints
- Techniques to express complex logic:
  - Decision tree
  - Decision table