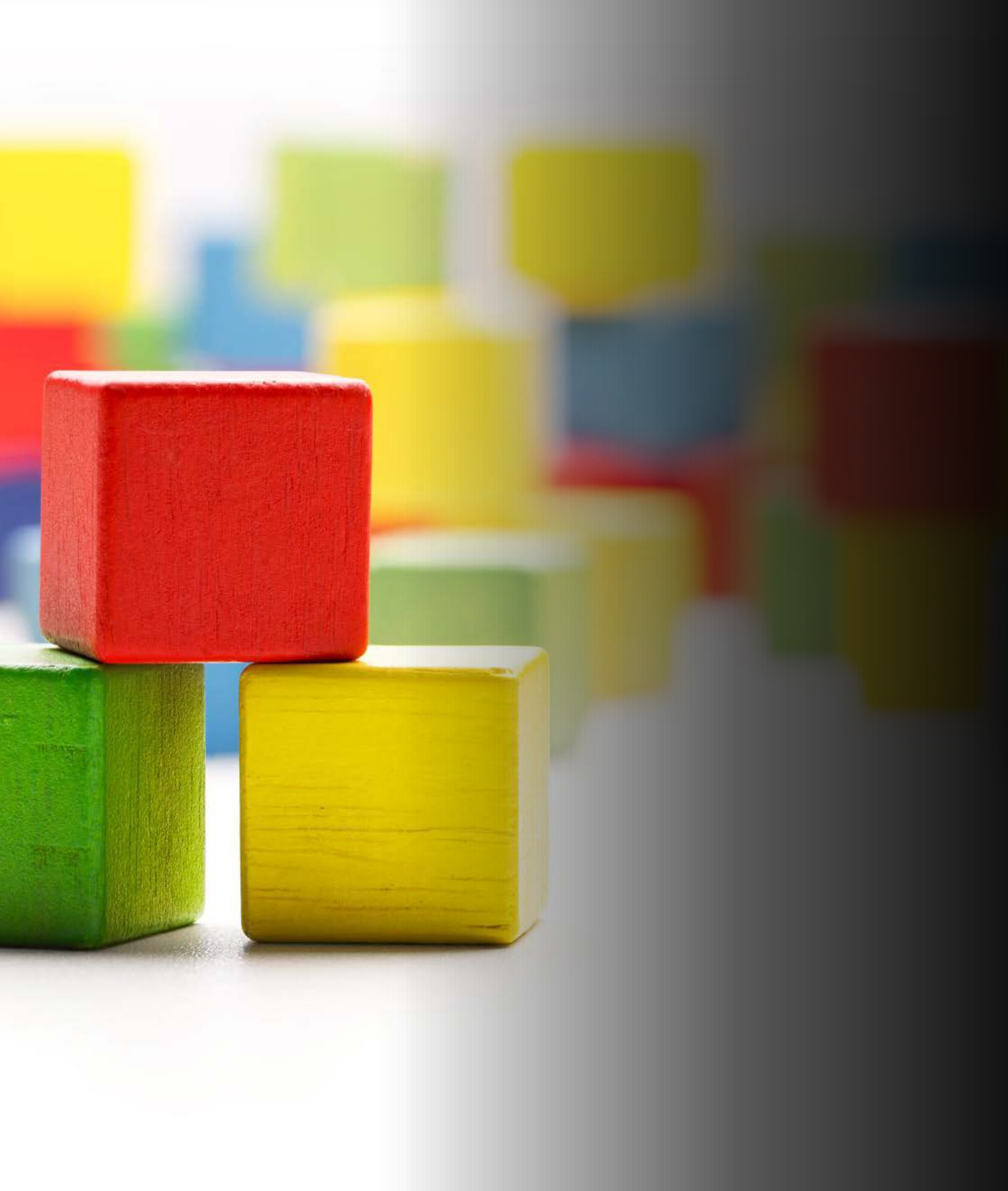


Software Engineering

Dr. Arijit Karati

Requirements Analysis and Specification

Recap...

- Adoption of a life cycle model.
  - ✓ A fundamental necessity while developing any large software product:
- Adherence to a software life cycle model:
  - ✓ Helps to do various development activities in a systematic and disciplined manner.
  - ✓ Also makes it easier to manage a software development effort.
- Different SLCMs and their comparisons.

Contents...

Requirements Analysis and Specification

- Many projects fail:
  - ✓ Because they start implementing the system.
  - ✓ Without determining whether they are building what the customer really wants.
- It is important to learn:
  - ✓ Requirements analysis and specification techniques carefully.

Requirements Analysis and Specification (CONT.)

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

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 (CONT.)

- 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
- 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

- Also known as requirements elicitation (啟髮式).
- 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 analysis
4. Scenario analysis
5. 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.
- Some desirable attributes of a good system analyst:
  - ✓ Good interaction skills,
  - ✓ Imagination and creativity,
  - ✓ Experience.

Case Study: Automation of Office Work at CSE Dept.

- The academic, inventory, and financial information at the CSE department:
  - ✓ Being carried though manual processing by two office clerks, a store keeper, and two attendants.
- Considering the low budget he had at his
- Disposal:
  - ✓ The HoD entrusted the work to a team of student volunteers.

Case Study: Automation of Office Work at CSE Dept. (CONT).

- The team was first briefed by the HoD 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 CSE Dept. (CONT).

- 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.

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:
  - ✓ Possibly 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.
- It is quite difficult to obtain:
  - ✓ A clear, in-depth understanding of the problem:
    - ❑ Especially if there is no working model of the problem.
- 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.

- 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.
- 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

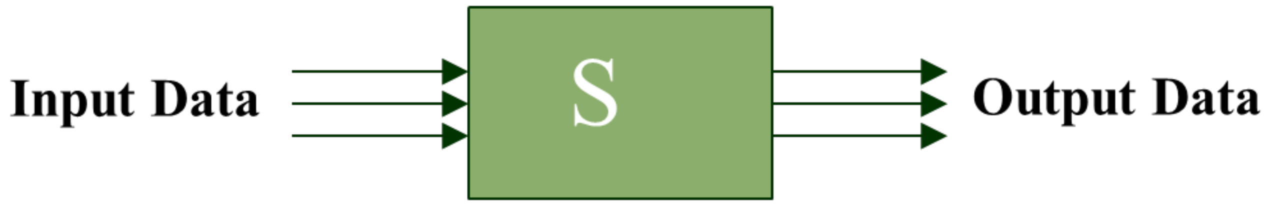
- Main aim of requirements specification:
  - ✓ Systematically organize the requirements arrived during requirements analysis.
  - ✓ Document requirements properly.
- The SRS document is useful in various contexts:
  - ✓ Statement of user needs
  - ✓ Contract document
  - ✓ Reference document
  - ✓ Definition 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.
- 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
- 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

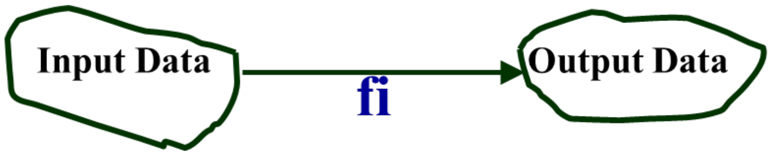
- It should be concise
  - ✓ and at the same time should not be ambiguous.
- It should specify what the system must do
  - ✓ and not say how to do it.
- Easy to change.,
  - ✓ i.e. it should be well-structured.
- It should be consistent.
- It should be complete.

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

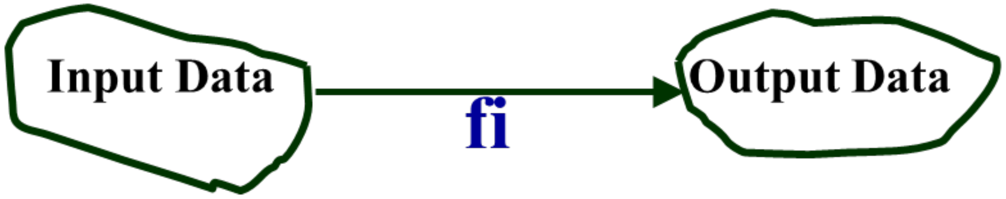
SRS Document (CONT.)

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



Functional Requirement

- 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.



High-Level Function

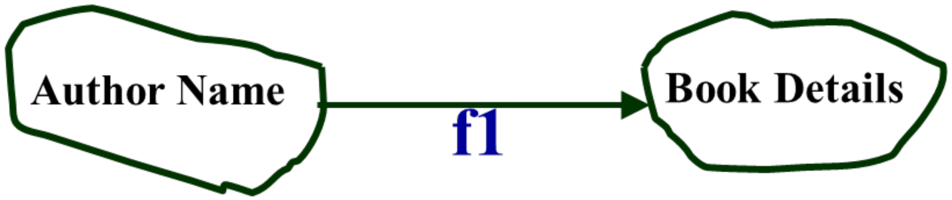
- A high-level function is
  - ✓ one Using which the user can get some useful piece of work done
  - ✓ Usually involves a series of interactions between the system and one or more users.
- Can the receipt printing work during withdrawal of money from an ATM:
  - ✓ Be called a useful piece of work?

High-Level Function (CONT.)

- 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.
- 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.

Functional Requirement (CONT.) [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.



Functional Requirement (CONT.) [Example]

- List all functional requirements
  - ✓ with proper numbering.

Req. 1:

- ✓ 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.

Functional Requirement (CONT.) [Example]

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.

Functional Requirement (CONT.) [Example]

Extend 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

Functional Requirement (CONT.) [Example]

Extend 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:
    - ❑ 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.

Functional Requirement (CONT.) [Example]

Extend 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 in the borrower list.

Functional Requirements (CONT.)

- 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
    - ❑ might consist of a set of identifiable functions
- 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.

Non-functional Requirements

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

Non-functional 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 (CONT.)

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

Organization of the SRS Document

- Introduction.
- Functional Requirements
- Non-functional Requirements
  - ✓ External interface requirements
  - ✓ Performance requirements
- Goals of implementation: 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

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 (CONT.)

- Noise:
  - ✓ Presence of text containing information irrelevant to the problem.
- Silence:
  - ✓ aspects important to proper solution of the problem are omitted.
- 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.

Examples of Bad SRS Documents
(CONT.)

- 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"
- 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.

Representation of complex processing logic:

- Decision trees
- Decision tables

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.

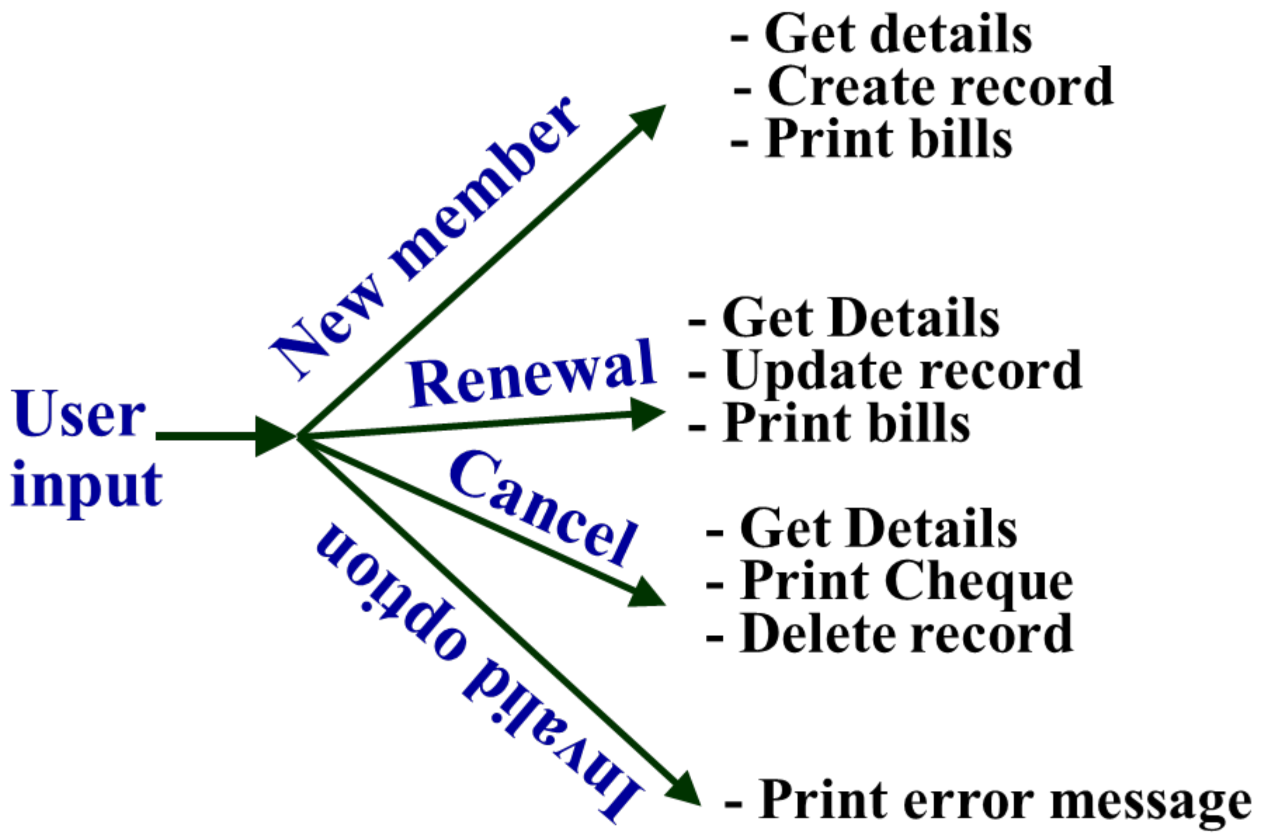
Example: LMS

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

Example: LMS (CONT.)

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

- Decision tables specify:
  - ✓ Which variables are to be tested
  - ✓ What actions are to be taken if the conditions are true,
  - ✓ The order in which decision making is performed.
- A decision table shows in a tabular form:
  - ✓ Processing logic and corresponding actions
- Upper rows of the table specify:
  - ✓ The variables or conditions to be evaluated

Decision Table (CONT.)

- Lower rows specify:
  - ✓ The actions to be taken when the corresponding conditions are satisfied.
- 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 -- -- --
Ask member's name etc.
Build customer record -- -- --
Generate bill -- --
Ask membership details --
Update expiry date -- -- --
Print cheque -- -- --
Delete record -- -- --

Comparison

- Both decision tables and decision trees
  - ✓ Can represent complex program 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

- Any failure of a safety-critical product
  - ✓ might result in loss of life and property
- How can we ensure that specification is trouble free? ---- 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.
- Advantages:
  - ✓ Well-defined semantics, no scope for ambiguity
  - ✓ Automated tools can check properties of specifications
  - ✓ Executable specification

Formal Specification (CONT.)

- Disadvantages of formal specification techniques:
  - ✓ Difficult to learn and use
  - ✓ Not able to handle complex systems
- Mathematical techniques used include:
  - ✓ Logic-based
  - ✓ set theoretic
  - ✓ algebraic specification
  - ✓ finite state machines, etc.

Semi-formal Specification

- Structured specification languages
  - ✓ SADT (Structured Analysis and Design Technique)
- PSL/PSA (Problem Statement Language/Problem Statement Analyser)
  - ✓ PSL is a semi-formal specification language
  - ✓ PSA can analyse the specifications expressed in PSL

Summary

- Requirements analysis
- Requirements specification
- SRS document
  - ✓ Organization: Functional, Non-Functional and Goal of Implementation
  - ✓ GOOD style and BAD Style
- complex processing logic
  - ✓ Decision table
  - ✓ Decision tree
- Formal and Semi-formal Specifications