Unit 2

Requirements Analysis and Specification

(Book to be preferred: Fundamentals of Software Engineering- 4th Edition, Rajib Mall, PHI)

- No satisfactory solution without clear understanding and proper documentation of the requirements for the problem
- Good requirements document: Perform clear understanding of features required from software, and serves as basis for activities carried out during later life cycle phases
- Starts after feasibility study stage, ends with development and review of requirements specification (SRS) document
- Goal: To clearly understand customer requirements and systematically organize them into SRS document
- Done by System Analysts
- Two activities:
  - Requirements gathering and analysis
  - Requirements specification

Requirements Gathering and Analysis

- Complete requirements rarely obtainable from single customer
- Requirements to be gathered from several sources.
- Gathered requirements to be analyzed to remove problems frequently occurring in the requirements
- Divide activity into two tasks:
  - Requirements gathering
  - Requirements analysis

Requirements Gathering (Requirements elicitation)

- Objective: to collect requirements from the stakeholders
- Difficult and challenging if no working model
- Good analysts share their experience and expertise with customer and give suggestions to define functionalities more general and complete
- Important ways in which an experienced analyst gathers requirements:

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

Requirements Gathering (Requirements elicitation)

1. Studying existing documentation:
    - Studies all available documents
    - Customers provide statement of purpose (SoP) document
    - Discuss issues of context, purpose, features, etc.

2. Interview
    - Identify different user categories and determine requirements of each
    - Follow Delphi technique
    - Consolidate understood requirements into a document, circulate it for comments from users, then refine the document. Repeat till users agree on set of requirements.

Requirements Gathering (Requirements elicitation)

3. Task analysis
   - Black-box view of software to user, consider providing services
   - Services/ functionalities/ tasks
   - E.g.: issue a book service- different steps

1. Scenario analysis
   - Task with different scenarios under different situations
   - E.g.: scenario for the book issue task

2. Form analysis
   - Involves automating existing manual system
   - Analyze the existing forms and notification formats
   - Determine data input and output for the system

Requirements Analysis

- Purpose: To analyze gathered requirements to remove all ambiguities, incompleteness, inconsistencies
- Obtain clear understanding of software to be developed

- Three types of problems in the requirements:

1. Anomaly
   - Lead to incorrect system development
   - E.g.: Switch off heater when temperature becomes high

     If student scores low grade, then inform their parents

2. Inconsistency
   - One requirement contradicts the other
   - E.g.: Switch off furnace when temperature is above 200 degree Celsius

     Switch on water shower and keep furnace on when temperature is above 200 degree Celsius

Requirements Analysis

3. Incompleteness
   - Some requirements are overlooked
   - Experienced analyst detect missing features and suggest to customer for approval
   - E.g.: If student scores less than 6 GPA, then intimate regrettable performance to parents through postal letter and e-mail

Software Requirements Specification

- Organize requirements in SRS document

Users of SRS:

1. Users, customers, marketing personnel
2. Software developers
3. Test engineers
4. User documentation writers
5. Project managers
6. Maintenance engineers

Uses of well documented SRS:

1. Forms an agreement between customers and developers
2. Reduces future reworks
3. Provides a basis for estimating costs and schedules
4. Provides a baseline for validation and verification
5. Facilitates future extensions

Software Requirements Specification

- Organize requirements in SRS document(black-box specification)
- Describe system as black box, specify externally visible behavior

Characteristics(Qualities) of a Good SRS Document:

1. Concise
2. Implementation-independent
3. Traceable
4. Modifiable
5. Identification of response to undesired events
6. Verifiable

Attributes of Bad SRS Document(Problems):

1. Over-specification
2. Forward references
3. Wishful thinking
4. Noise

Software Requirements Specification

Categories of Customer/ User Requirements:

- SRS document should clearly document following aspects of a software:
  - Functional requirements
  - Non-functional requirements
    - Design and implementation constraints
    - External interfaces required
    - Other non-functional requirements
  - Goals of implementation

Functional Requirements

- Describe **what** the system should do
- What inputs/outputs
- What data the system should store
- What computations the system should perform
- Concepts, functions, features, information, Behaviors. These are generally listed as **shall statements** starting with **"The system shall ..."**.

Non- Functional Requirements

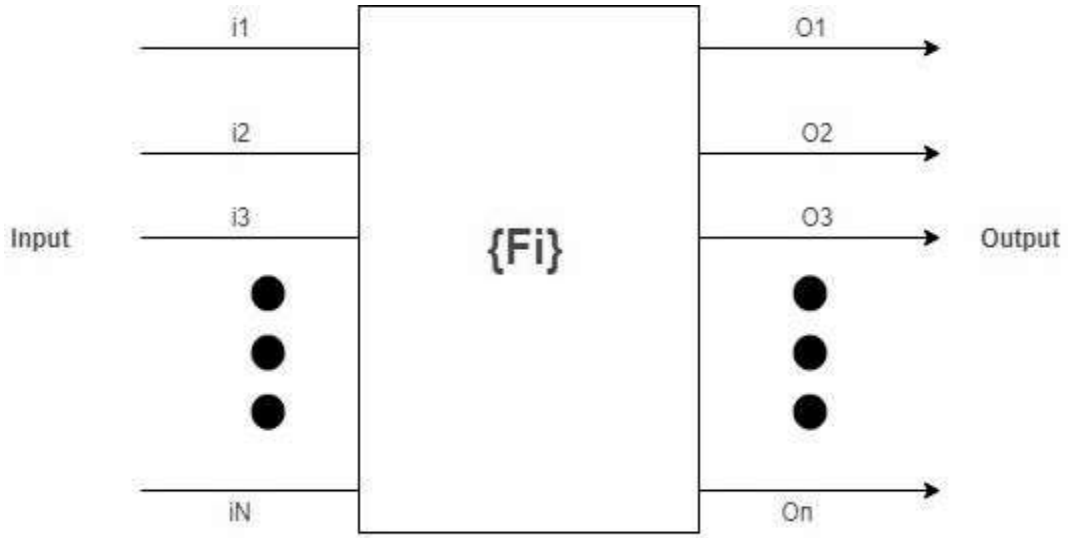
- Constraints that must be accomplished or adhered to ...
  - *Response time*
  - *Throughput*
  - *Resource usage*
  - *Reliability*
  - *Availability*
  - *Security*
  - *Recovery from failure*
  - *Cost*
  - *Technology to be used*

Other Requirements

- Enviromental Requirements
- Schedulling Requirements

1. Functional requirements:

- Capture functionalities required by users from the system
- Mathematical function f: I → O
- Function f_i read set of data i_i , transform it into set of data o_i
- Clearly describe each functionality with corresponding input and output data set



View of a system performing a set of functions

2. Non-functional requirements:

- Requirements that cannot be expressed as functions
- E.g.: accept input, produce output
- Address aspects of external interfaces, user interfaces, maintainability, portability, usability, max. no. of concurrent users, timing, throughput, etc.

2.1 Design and implementation constraints:

- Describe issues or items that will limit the options available
- E.g.: regulatory policies, h/w limitations, interfaces with other applications, technologies, tools, databases, communication protocols, security considerations, design conventions, programming standards, etc.

2.2 External interfaces required:

- E.g.: h/w, s/w and communication interfaces, user interfaces, report formats, etc.
- User interfaces- sample screen images, GUI standards, screen layout constraints, std. buttons/ functions appear on screen, keyboard shortcuts, error message display standards, etc.

2.3 Other non-functional requirements:

- Contain description of requirements other than design constraints and external interfaces
- E.g.: performance requirement- no. of transactions completed per unit time, reliability issues, result accuracy, security issues

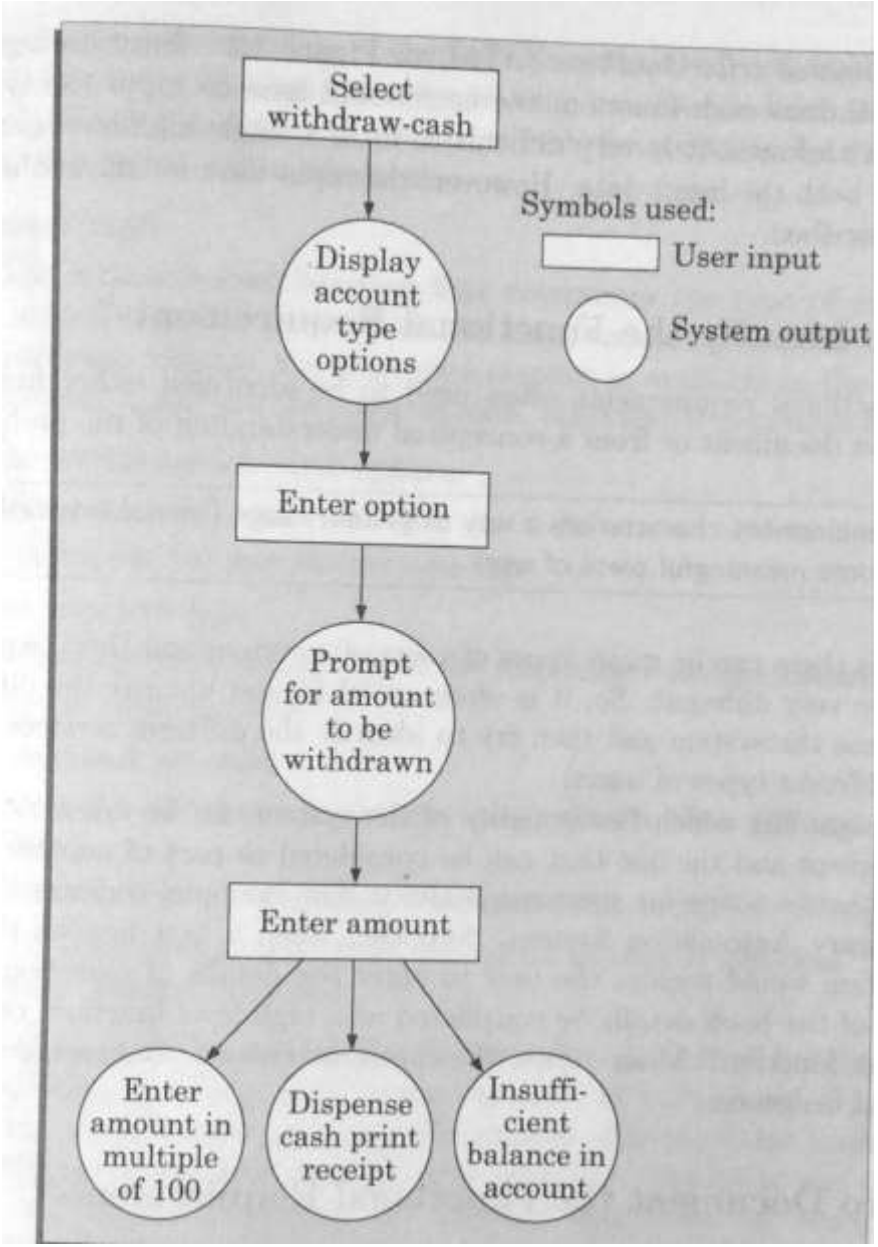
3. Goals of implementation:

- Offer general suggestions like choosing among different design solutions

Software Requirements Specification

Functional Requirements:

- Identify high-level functions from informal documentation and gathered requirements
- Split functions into smaller subrequirements
- Involve accepting data through user interface, transform it into required response, display response in proper format
- E.g.: search-book
- High-level function involves series of interactions between system and users
- For a function, there can be different interaction sequences or scenarios due to users selecting different options or entering different data items



User and system interactions in high-level functional requirements

Software Requirements Specification

Identify Functional Requirements:

- Identify from informal problem description document or from conceptual understanding of problem
- Identify user types and services expected from the software
- Decision regarding which functionality as high-level and which as part of another function
- E.g.: issue-book function

Document Functional Requirements:

- Identify state at which data is to be input , its input data domain, output data domain, types of processing on input data to obtain output data
- E.g.: Document withdraw-cash function of ATM system

- E.g.: Document functional requirements for withdraw-cash function of ATM system

R.1: Withdraw cash

R.1.1: Select withdraw amount option

Input:

Output:

R.1.2: Select account type

Input:

Output:

R.1.3: Get required amount

Input:

Output:

Processing:

- E.g.: Document functional requirements to search book and renew book for Library system

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 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. 1:

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

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 him are displayed.
  - The user can renew any of the books:
    - By clicking in the corresponding renew box.

Req. 2:

- R2.1:select renew option
  - Input: “renew” option selected,
  - Output: user prompted to enter his membership number and password.
- R2.2:login
  - 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.

Req. 2:

- R2.3:renew selected books
  - 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 them in the borrower list.

Organization of SRS Document:

- Introduction
  - Purpose:
  - Project scope:
  - Environmental characteristics:
- Overall description of organization of SRS document
  - Product perspective:
  - Product features:
  - User classes:
  - Operating environment:
  - Design and implementation constraints:
  - User documentation:
- Functional requirements for organization of SRS document

1. User class 1
   (a) Functional requirement 1.1
   (b) Functional requirement 1.2
2. User class 2
   (a) Functional requirement 2.1

(b) Functional requirement 2.2

- External interface requirements
  - User interfaces:
  - Hardware interfaces:
  - Software interfaces:
  - Communications interfaces:
- Other non-functional requirements for organization of SRS document
  - Performance requirements:
  - Safety requirements:
  - Security requirements:
- Functional requirements
  1. Operation mode 1
     (a) Functional requirement 1.1
     (b) Functional requirement 1.2
  2. Operation mode 2
     (a) Functional requirement 2.1
     (b) Functional requirement 2.2

Personal Library Software Example

Functional requirements

1.Manage own books

- 1.1 register book
- 1.2 Issue book
  - 1.2.1Display outstanding books
  - 1.2.2 confirm issue book
- 1.3 query outstanding books
- 1.4 query book
- 1.5 Return book

2.Manage friends details

2.1 register friend

2.2 Update friend details

2.2.1 display current details

2.2.2 update friend details

2.2.3 delete friend record

3. Manage borrowed books

3.1 register borrowed books

3.2 deregister borrowed books

3.3 Display borrowed books

4. Manage statistics

4.1 display book count

4.2 Display amount invested

4.3 display number of transactions

Non functional requirements

1. Databases
2. Platform
3. Web support

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.

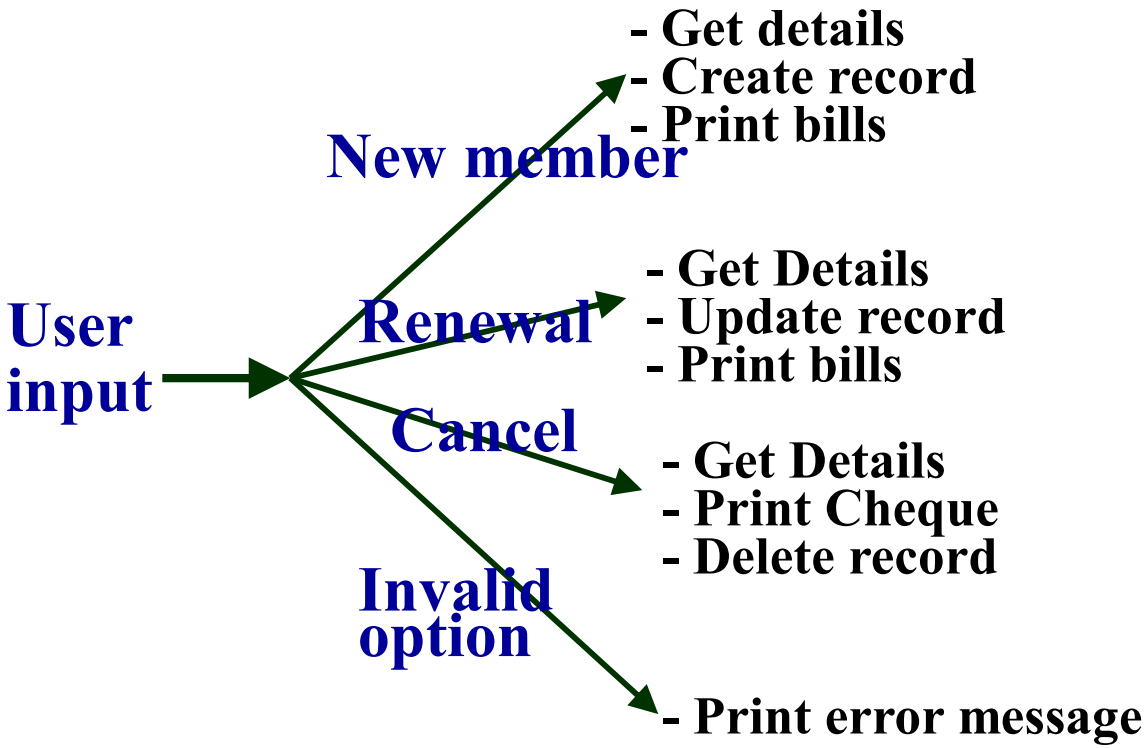
- When the **new member** option is selected,
  - The software asks details about the member:
    - name,
    - address,
    - phone number, etc.
  - If proper information is entered,
    - A membership record for the member is created
    - A bill is printed for the annual membership charge plus the security deposit payable.

Example: LMS

- 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:
  - Variables or conditions to be evaluated
- Lower rows specify:
  - Actions to be taken when 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: LMS

- 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 name etc.		X		
Build customer record		X		
Generate bill		X	X	
Ask membership details			X	X
Update expiry date			X	
Print cheque				X
Delete record				X

Formal System Specification

- 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

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

Formal Specification Language

- Consists of two sets:
  - *syn* – syntactic domain
  - *sem* – semantic domain
  - *sat* – satisfaction relation

For a given specification *syn*, and model of the system *sem*,

if *sat(syn*, *sem)*, then *syn* is said to be the specification of *sem*, and *sem* is said to be the *specificand of syn*.

Formal Specification Language

- Relations
  - *syn* – consists of alphabet of symbols & formation rules to construct well-formed formulas from alphabet
  - *sem* – formal techniques- algebras, theories & programs, programming languages, state sequences, event sequences, state transition, state m/c, trees etc.
  - *sat* – is determined by using homomorphism known as *semantic abstraction function*
    - Function
      - maps elements of semantic domain into equivalent classes
      - Preserve system's behavior and system's structure

Formal methods

- Model oriented method:
  - Suitable for system design
  - Defines system behavior
  - Construct mathematical model – tuples, relations, functions, sets, sequences, etc.
  - Operations :
    - *p* produce – S1 + p => S
    - *c* consume – S + c => S1
    - Examples: Z, CSP, CCS
  - Do not support augmentations, changes not possible

Formal methods

- Property oriented method:
  - Defines system behavior
  - Suitable for requirement specification
  - Producer-consumer system
  - Eg : Algebraic specification
  - Changes/Augmentation of specification possible by conjunction of axioms

Operational Semantics

- Way computations are represented
- Behavior of system
- *Single run* or grouping of runs of system
- Linear semantics –

  run of a system described by sequence of events & states
- Branching semantics –

  directed graph, nodes-states
- Maximally parallel semantics –

  concurrent actions at state, availability of resources
- Partial order semantics –

  structures of states, partial order relation among states, *precedence ordering*