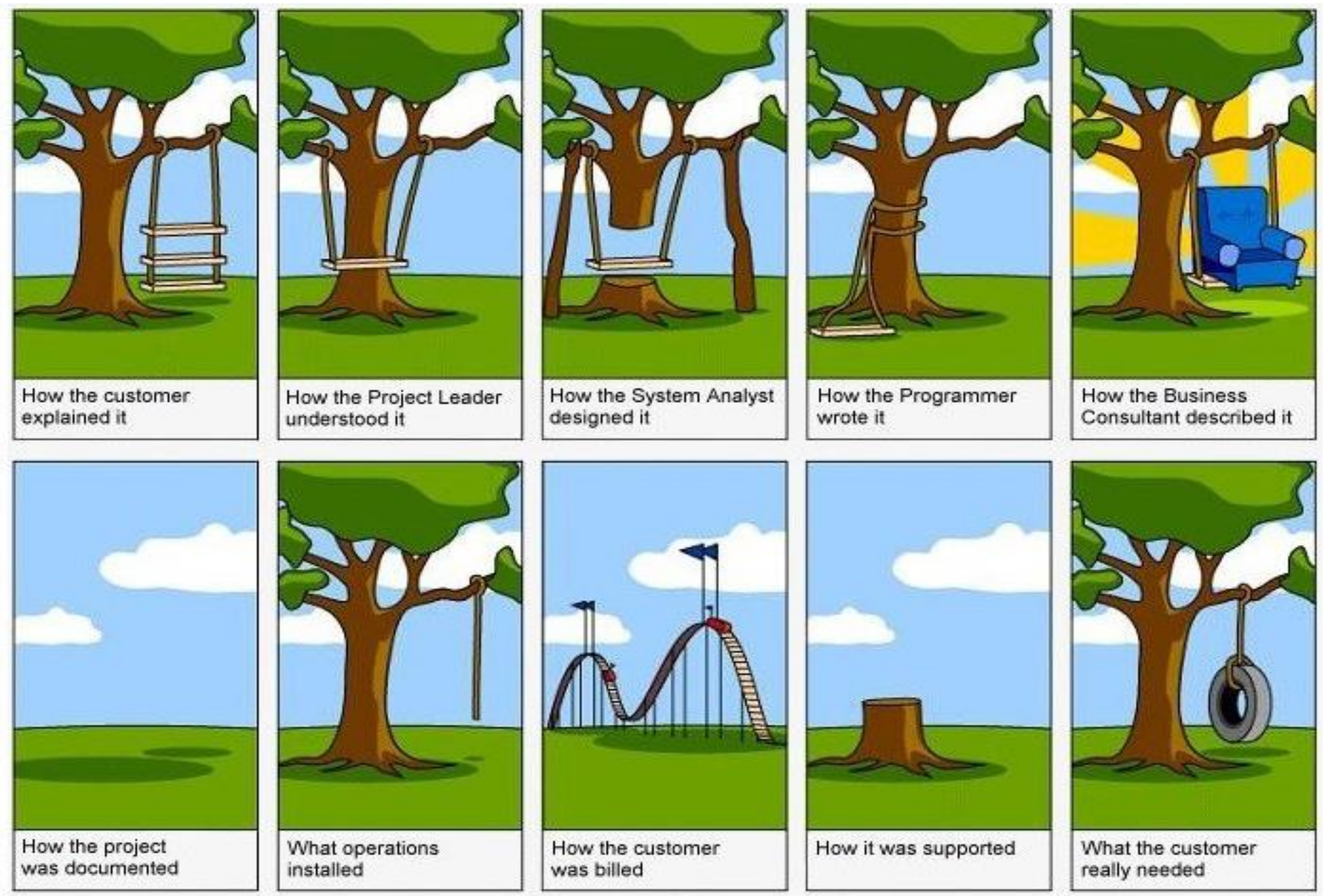
Requirements Analysis and Specification

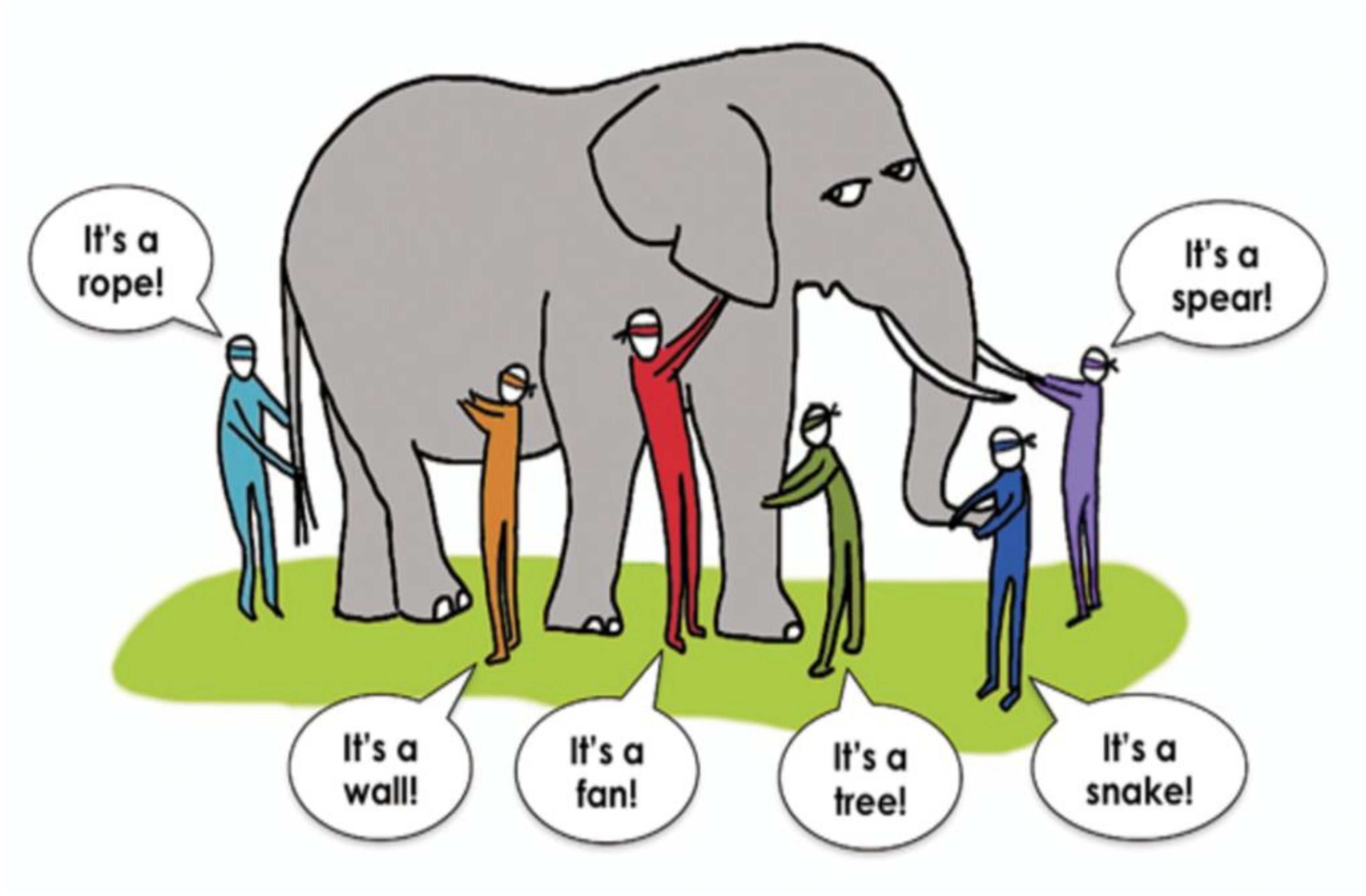
Requirements Analysis and Specification

- Many projects fail:
  - Because they direct 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 to develop successful software product.

The Fact about Requirements Engineering


How the customer explained it
How the Project Leader understood it
How the System Analyst designed it
How the Programmer wrote it
How the Business Consultant described it
How the project was documented
What operations installed
How the customer was billed
How it was supported
What the customer really needed

The Fact about Requirements Engineering



Requirement Engineering

- The process of establishing the services that the customer requires from a system and the constraints under which it operates and is developed.
- The requirements themselves are the descriptions of the system services and constraints that are generated during the requirements engineering process.

Types of Requirements

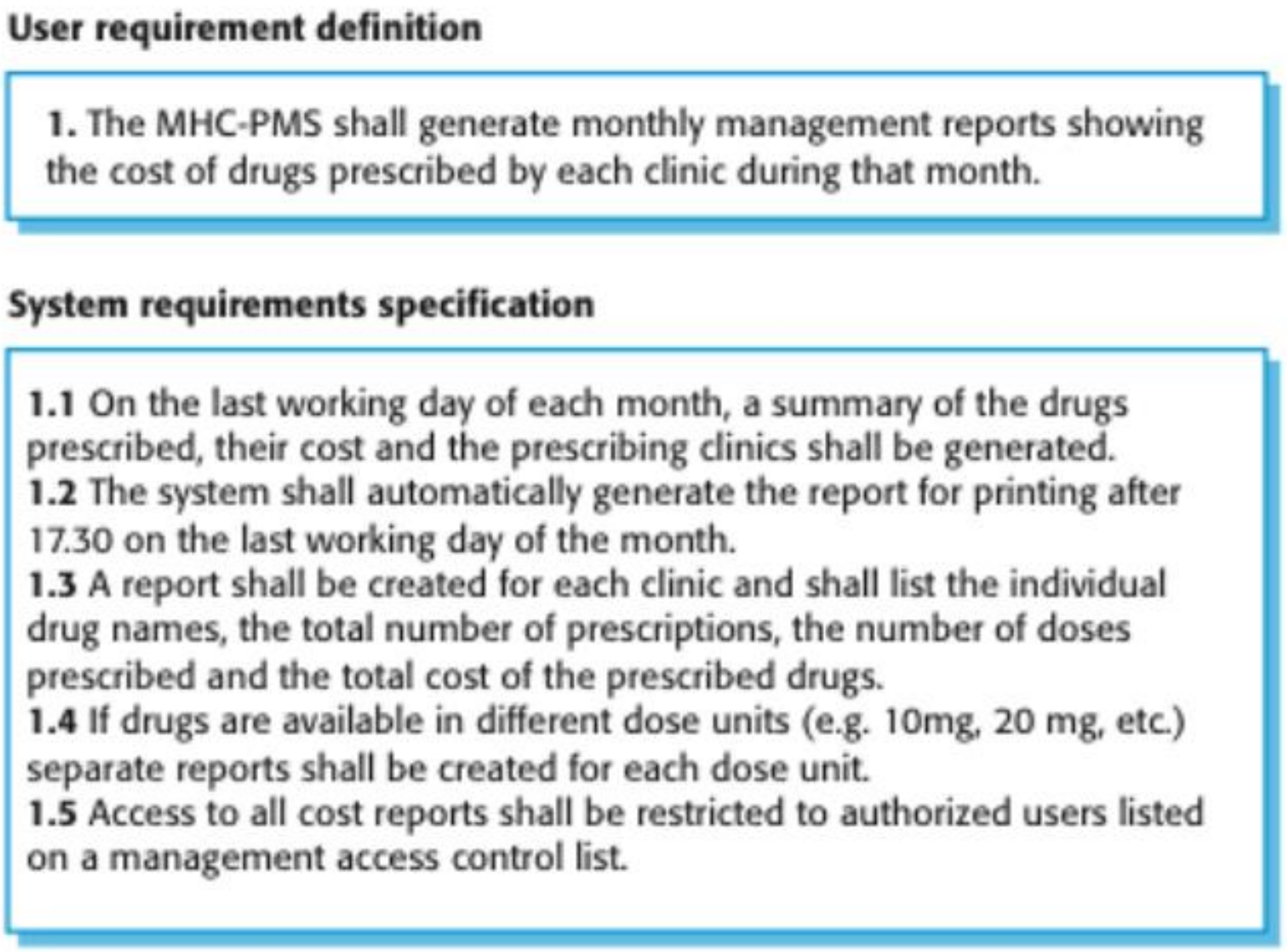
- User requirements

  – Statements in natural language plus diagrams of theservices the system provides and its operational constraints. Written for customers.

- System requirements

  – A structured document setting out detailed descriptions of the system's functions, services and operational constraints. Defines what should be implemented so may be part of a contract between client and contractor.

Types of Requirements



User requirement definition

1. The MHC-PMS shall generate monthly management reports showing the cost of drugs prescribed by each clinic during that month.

System requirements specification

1.1 On the last working day of each month, a summary of the drugs prescribed, their cost and the prescribing clinics shall be generated.
1.2 The system shall automatically generate the report for printing after 17.30 on the last working day of the month.
1.3 A report shall be created for each clinic and shall list the individual drug names, the total number of prescriptions, the number of doses prescribed and the total cost of the prescribed drugs.
1.4 If drugs are available in different dose units (e.g. 10mg, 20 mg, etc.) separate reports shall be created for each dose unit.
1.5 Access to all cost reports shall be restricted to authorized users listed on a management access control list.

Functional and non-functional requirements

- **Functional requirements**
  - Statements of services the system should provide, how the system should react to particular inputs and how the system should behave in particular situations.
  - May state what the system should not do.
- **Non-functional requirements**
  - Constraints on the services or functions offered by the system such as timing constraints, constraints on the development process, standards, etc.
  - Often apply to the system as a whole rather than individual features or services.
- **Domain requirements**
  - Constraints on the system from the domain of operation

Requirements imprecision

- Problems arise when requirements are not precisely stated.
- Ambiguous requirements may be interpreted in different ways by developers and users.
- Consider the term ‘search’ in requirement 1
  - User intention – search for a patient name across all appointments in all clinics;
  - Developer interpretation – search for a patient name in an individual clinic. User chooses clinic then search.

Requirements imprecision

- In principle, requirements should be both complete and consistent.
- Complete
  - They should include descriptions of all facilities required.
- Consistent
  - There should be no conflicts or contradictions in the descriptions of the system facilities.
- In practice, it is impossible to produce a complete and consistent requirements document.

Non-functional requirements

- These define system properties and constraints e.g. reliability, response time and storage requirements.
- Constraints are I/O device capability, system representations, etc.
- Process requirements may also be specified mandating a particular IDE, programming language or development method.
- Non-functional requirements may be more critical than functional requirements.
- If these are not met, the system may be useless.

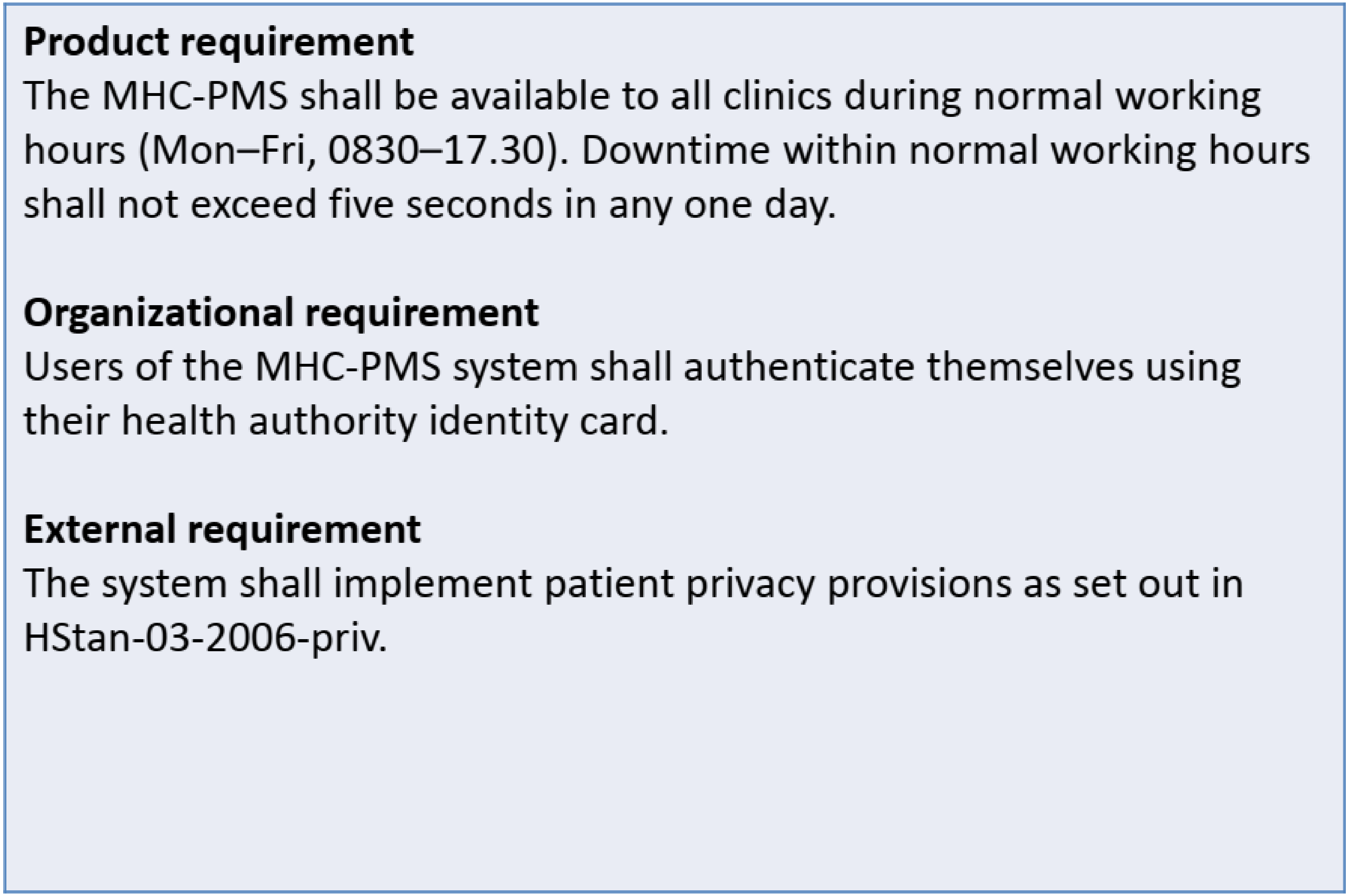
Non-functional requirements

- Non-functional requirements may affect the overall architecture of a system rather than the individual components.
  - For example, to ensure that performance requirements are met, you may have to organize the system to minimize communications between components.
- A single non-functional requirement, such as a security requirement, may generate a number of related functional requirements that define system services that are required.
  - It may also generate requirements that restrict existing requirements.

Non-functional requirements

- Product requirements
  - Requirements which specify that the delivered product must behave in a particular way e.g. execution speed, reliability, etc.
- Organisational requirements
  - Requirements which are a consequence of organisational policies and procedures e.g. process standards used, implementation requirements, etc.
- External requirements
  - Requirements which arise from factors which are external to the system and its development process e.g. interoperability requirements, legislative requirements, etc.

Non-functional requirements



Product requirement
The MHC-PMS shall be available to all clinics during normal working hours (Mon–Fri, 0830–17.30). Downtime within normal working hours shall not exceed five seconds in any one day.

Organizational requirement
Users of the MHC-PMS system shall authenticate themselves using their health authority identity card.

External requirement
The system shall implement patient privacy provisions as set out in HStan-03-2006-priv.

Specifying nonfunctional requirements

Property	Measure
Speed	Processed transactions/second User/event response time Screen refresh time
Size	Mbytes Number of ROM chips
Ease of use	Training time Number of help frames
Reliability	Mean time to failure Probability of unavailability Rate of failure occurrence Availability
Robustness	Time to restart after failure Percentage of events causing failure Probability of data corruption on failure
Portability	Percentage of target dependent statements Number of target systems

Domain requirements

- The system's operational domain imposes requirements on the system.
  - For example, a train control system has to take into account the braking characteristics in different weather conditions.
- Domain requirements be new functional requirements, constraints on existing requirements or define specific computations.
- If domain requirements are not satisfied, the system may be unworkable.

Domain requirements

- It is difficult for a non-specialist to understand the implications of this and how it interacts with other requirements.
- Understandability
  - Requirements are expressed in the language of the application domain;
  - This is often not understood by software engineers developing the system.
- Implicitness
  - Domain specialists understand the area so well that they do not think of making the domain requirements explicit.

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.
- 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

- 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

- 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.
- 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 Activities

Studying the existing documentation:

- The analyst studies all the available documents regarding the system to be developed before visiting the customer site.
- Customers usually provide statement of purpose (SoP) document to the developers.
- These documents might discuss issues such as the context in which the software is required, the basic purpose, the stakeholders, features of any similar software developed elsewhere, etc.

Requirements Gathering Activities

Interview:

- There are many different categories of users of a software.
- Each category of users requires a different set of features from the software.
- Thus, it is important for the analyst to first identify the different categories of users and then determine the requirements of each.
- For example, the different categories of users of a library automation software could be the *library members*, *the librarians*, and *the accountants*.

Requirements Gathering Activities

- The library members would like to use the software to query availability of books and issue and return books.
- The librarians might like to use the software to determine books that are overdue, create member accounts, delete member accounts, etc.
- To systematize this method of requirements gathering, the Delphi technique can be followed.
- The analyst consolidates the requirements as understood by him in a document and circulates it for the comments of the various categories of users.
- Based on their feedback, document is refined.
- This procedure is repeated till the different users agree on the set of requirements.

Requirements Gathering Activities

Task Analysis:

- The users usually have a black-box view of a software and consider the software as something that provides a set of services (functionalities).
- A service supported by a software is also called a *task*.
- A software performs various tasks of the users.
- In this context, the analyst tries to identify and understand the different tasks to be performed by the software.

Requirements Gathering Activities

- For each identified task, the analyst tries to formulate the different steps necessary to realize the required functionality in consultation with the users.

For example, for the issue book service, the steps may be authenticate user, check the number of books issued to the customer and determine if the maximum number of books that this member can borrow has been reached, check whether the book has been reserved, post the book issue details in the member's record, and finally print out a book issue slip that can be presented by the member at the security counter to take the book out of the library premises.

Requirements Gathering Activities

Form analysis:

- Form analysis activity is undertaken by the analyst, when the project involves automating an existing manual system.
- During the operation of a manual system, normally several forms are filled up by the stakeholders.
- These exiting forms are analyzed to determine the data input to the system and the data that are output from the system.
- For the different sets of data input to the system, how these input data would be used by the system to produce the corresponding output data is determined from the users.

The details about requirement elicitation techniques can be referred from the Zowghi & Coulin "*Requirements Elicitation: A Survey of Techniques, Approaches, and Tools*".

Requirements Gathering

- 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.

- 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.

- 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

- 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.
- 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.
- The SRS document is useful in various contexts:
  - Statement of user needs
  - Contract document
  - Reference document
  - Definition for implementation

Software Requirements Specification

- SRS is a document that completely describes what the proposed software should do without describing how software do it.
- It contains:
  - A complete information description
  - A detailed functional description
  - A representation of system behaviour
  - An indication of performance requirements and design constrains
  - Appropriate validation criteria
  - Other information suitable to requirements

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 and should be well-structured.
- It should be consistent and unambiguous
- It should be complete
- It should be correct
- It should be ranked for importance
- It should be traceable
- It should be verifiable

Examples of Bad SRS Documents

- <u>Unstructured Specifications:</u>
  - 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

- <u>Ambiguity:</u>
  - Literary expressions
  - Unquantifiable aspects, e.g. “good user interface”
- <u>Forward References:</u>
  - References to aspects of problem
    - defined only later on in the text.
- <u>Wishful Thinking:</u>
  - Descriptions of aspects
    - for which realistic solutions will be hard to find.

Template for SRS

Title page

- Software Requirements Specification
- Version <<Annotated Version>>
- <<date>>
- Prepared by <<System Analyst details>>

Table of Contents

Revision History

Template for SRS

- Introduction
  - Purpose
  - Scope of Project
  - Glossary
  - References
- Overall Description
  - Product perspective
  - Product features
  - Operating environment
  - Design and implementation constraints
  - Assumptions and Dependencies

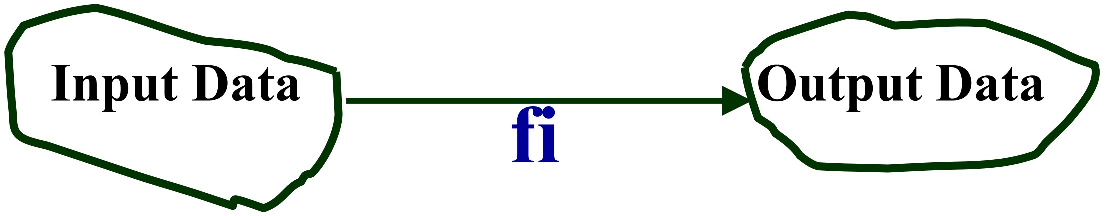
Template for SRS

System Description

- Functional Requirements Specification
  - Use cases
  - Use case description
- Non-Functional Requirements
  - Performance requirements
  - Safety requirements
  - Security requirements
  - Software quality attributes
- External Interface Requirements
  - User interfaces
  - Hardware interfaces
  - Software interfaces
  - Communication interfaces

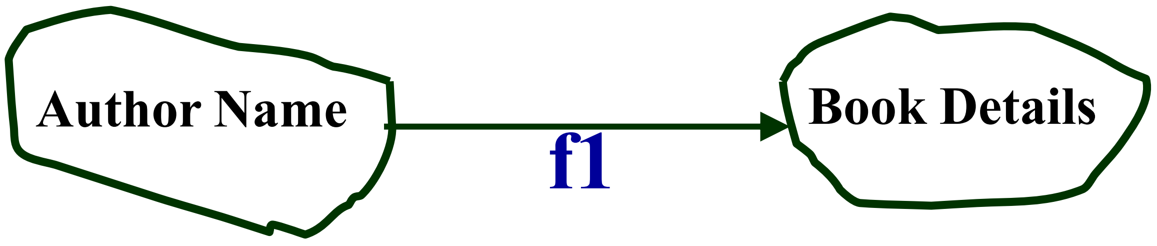
SRS Document

- 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

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:
  - 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

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:
  - 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.

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.

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.

Nonfunctional Requirements

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

Nonfunctional Requirements

- Non-functional 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

- **N.1: Database:** A data base management system that is available free of cost in the public domain should be used.
- **N.2: Platform:** Both Windows and Unix versions of the software need to be developed.
- **N.3: Web-support:** It should be possible to invoke the query book functionality from any place by using a web browser.

Representation of complex processing logic:

- Sometimes the conditions in functional requirements can be complex.
- Several alternative interaction and processing sequences may exist depending on the outcome of the corresponding condition checking.
- A simple text description in such cases can be difficult to comprehend and analyze.
- In such situations, we use
  - Decision trees
  - Decision tables

Decision Trees

- Decision trees:
  - A graphical representation
  - 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.
- 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.

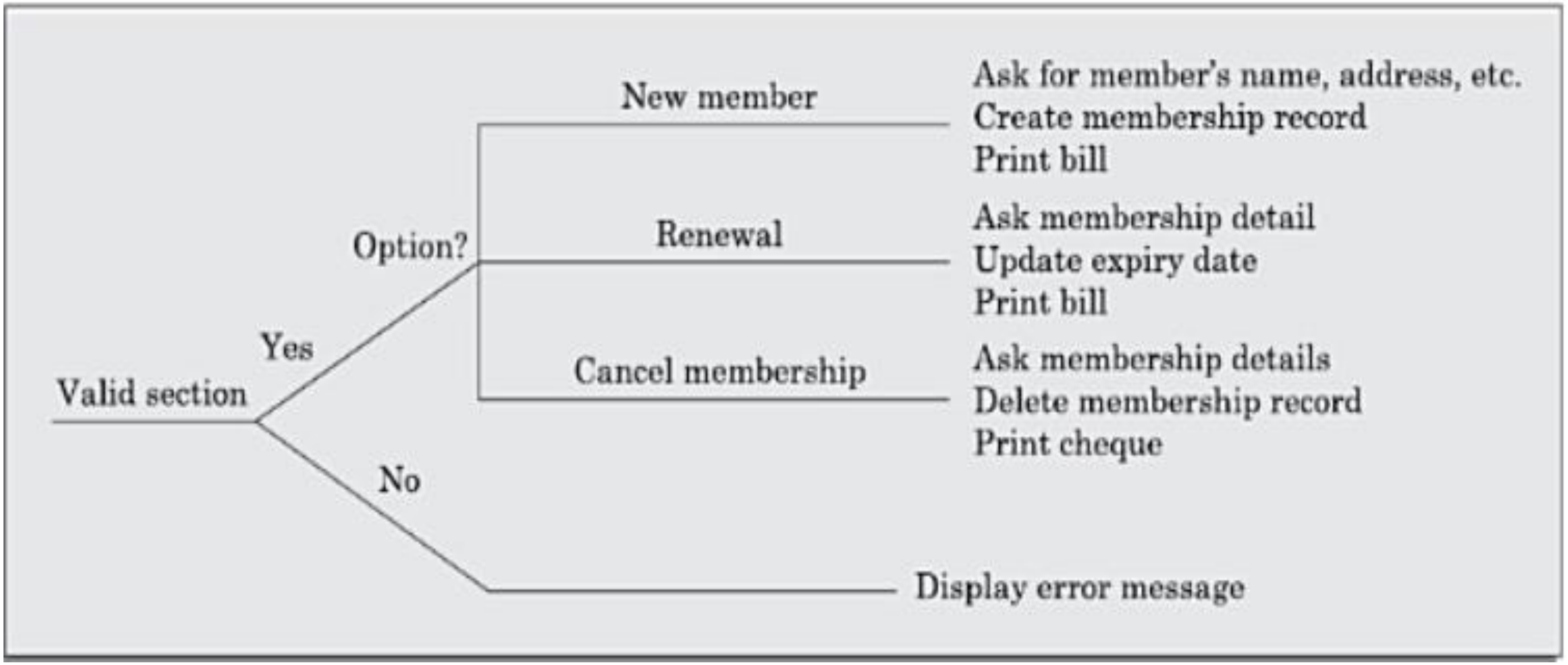
Example(cont.)

- 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

- 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.

Decision Table

- 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
- 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 <u>rule</u>:
  - A rule implies:
    - if a condition is true, then execute the corresponding action.

Example:

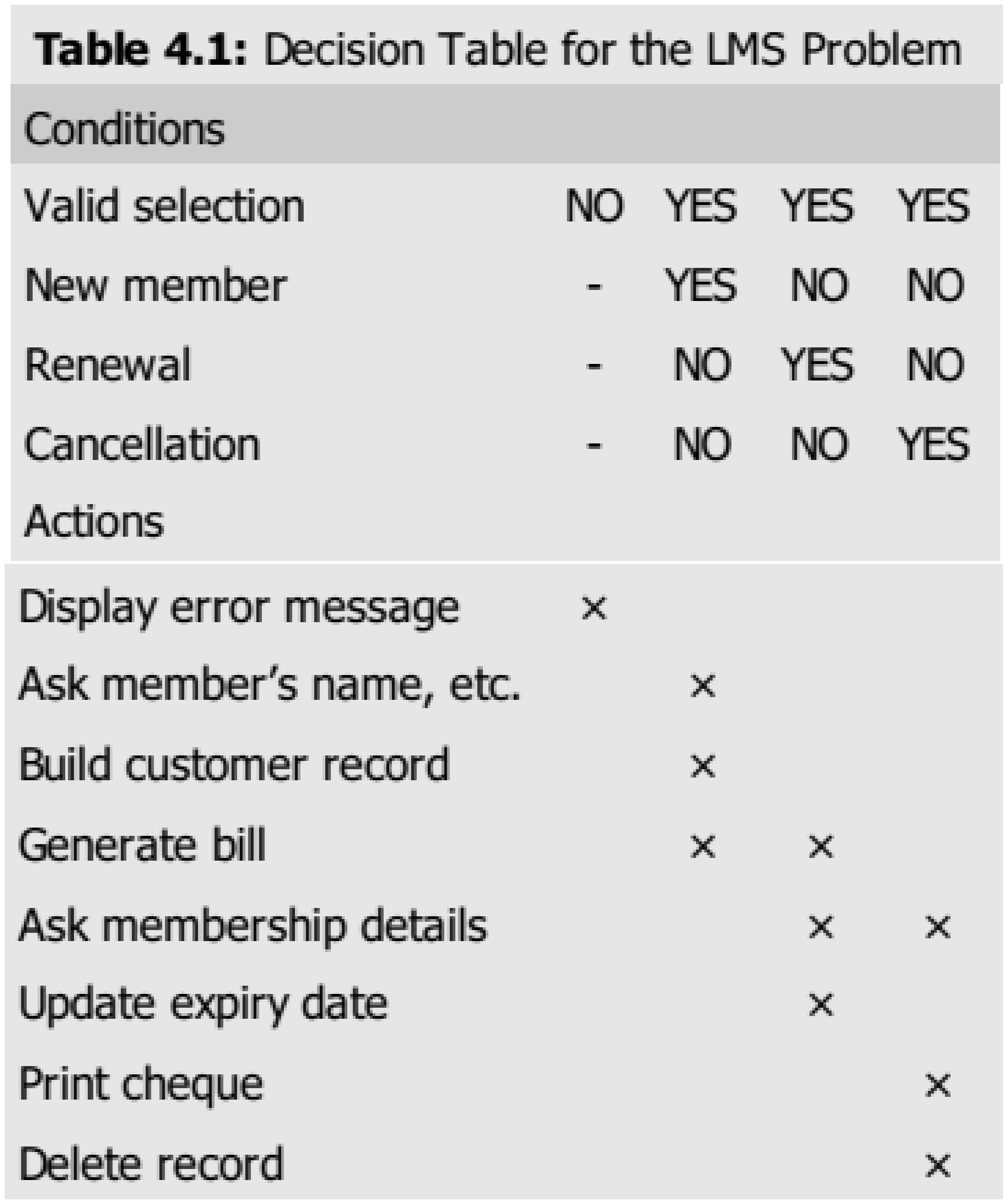


Table 4.1: Decision Table for the LMS Problem

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.

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
  - Constraints
- Techniques to express complex logic:
  - Decision tree
  - Decision table

Summary

- Formal requirements specifications have several advantages.
  - But the major shortcoming is that these are hard to use.