# Requirements Analysis and Specification

## 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.

- It is important to learn:
  - requirements analysis and specification techniques thoroughly.

## Requirements ?

- Being asked what you want out of a system is like being asked what you want out of life !
- Where do you start !
- What are the parameters !

## 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

## 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.
- **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

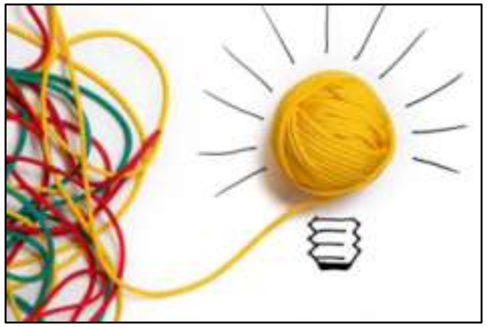
- If the project is to automate some existing procedures
  - e.g., automating existing manual accounting activities,
  - the task of the system analyst is a little easier
  - analyst can immediately obtain:
    - input and output formats
    - accurate details of the operational procedures

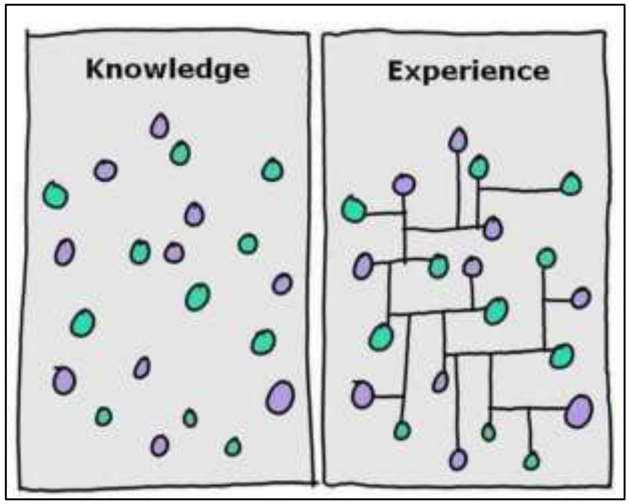
- In the absence of a working system,
  - lot of imagination and creativity are required.

- 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.





## Requirement Gathering techniques

- **Interviews** : Most common method
- **Questionnaires :** collect information from a target population which is very large and in remote locations
- **Observations:** Take notes while user is doing a process.
- **Facilitated Workshops**: bring a larger group on a common platform to discuss and reach agreements. They help to define cross-functional requirements for a product
- **JAD** :JAD (Joint Application Development) is a methodology that involves the client or end user in the design and development of an application, through a succession of collaborative workshops called JAD sessions.
- **Brainstorming :** It involves self-generated contribution of ideas by the group members around a specific issue, problem or requirement.
- **Prototyping**
- **Existing system/Documentation analysis**

# 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.
- <u>Example:</u>
  - 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.
- <u>Example:</u>
  - 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.
- 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?

- 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.

# Software Requirements Specification

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

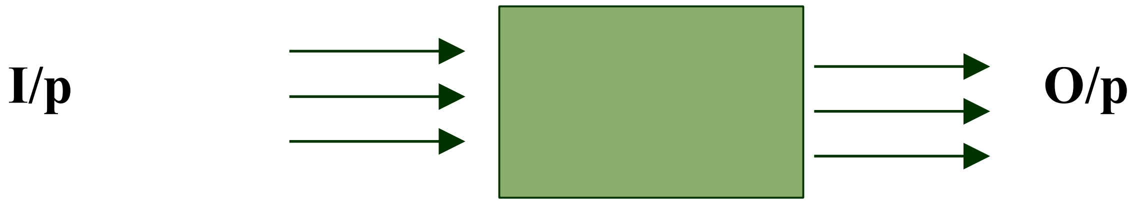
- The SRS document is useful in various contexts:
  - statement of user needs
  - contract document
  - reference document
  - definition for implementation

## 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 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

## Properties of a good SRS document

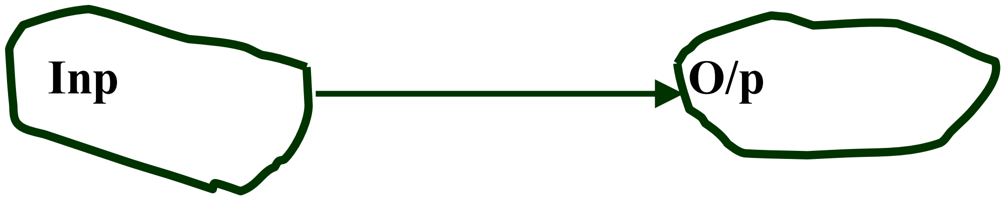
- It should be **concise**
  - and at the same time should not be ambiguous.
- It should specify what the system must do
  - and not say how to do it.
- Easy to change.,
  - i.e. it should be **well-structured.**
- It should be **consistent.**
- It should be **complete.**
- It should be **traceable**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

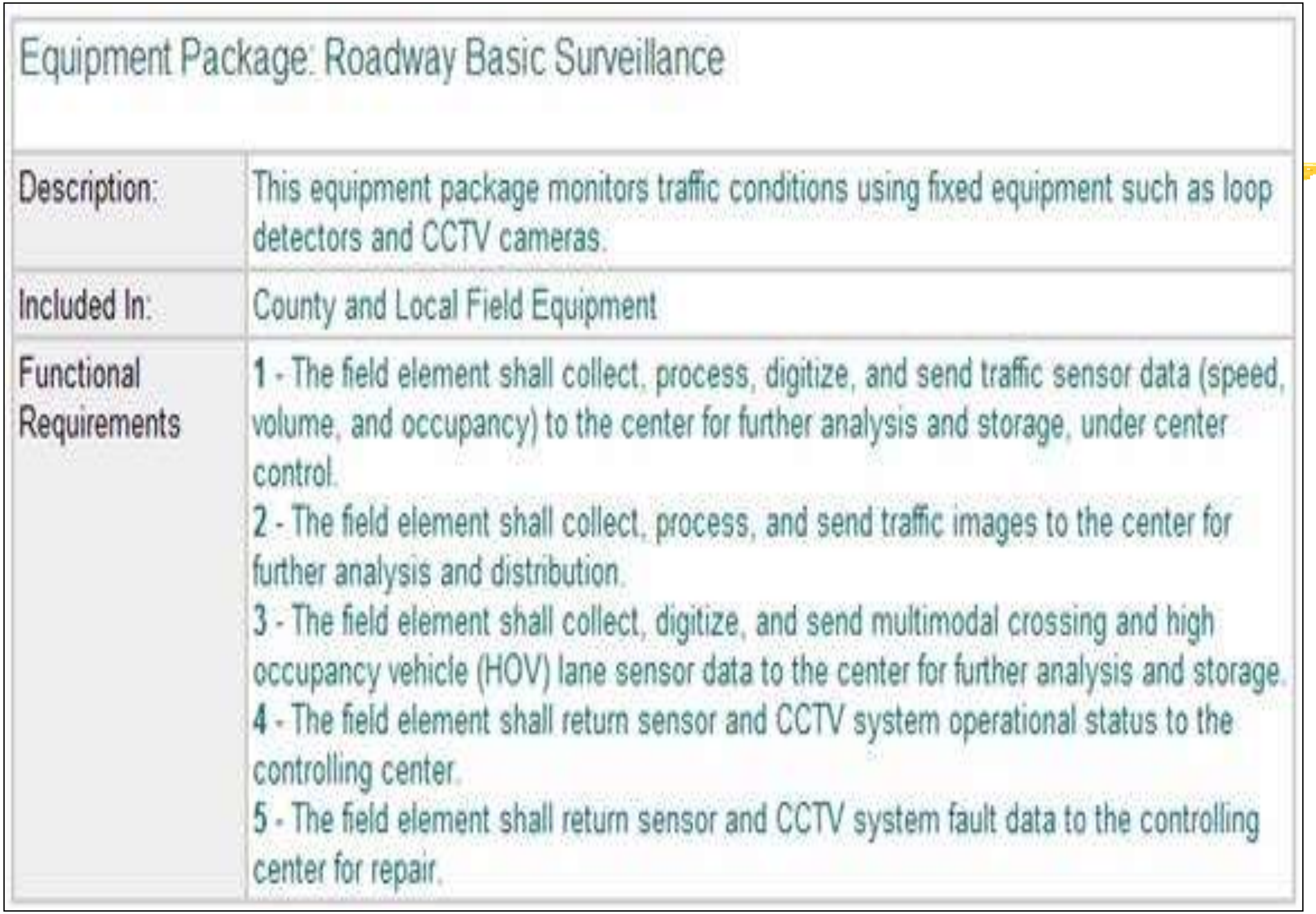
## SRS Document (CONT.)

- 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.



Equipment Package: Roadway Basic Surveillance

| | |
|---|---|
| Description: | This equipment package monitors traffic conditions using fixed equipment such as loop detectors and CCTV cameras. |
| Included In: | County and Local Field Equipment |
| Functional Requirements | 1 - The field element shall collect, process, digitize, and send traffic sensor data (speed, volume, and occupancy) to the center for further analysis and storage, under center control.<br>2 - The field element shall collect, process, and send traffic images to the center for further analysis and distribution.<br>3 - The field element shall collect, digitize, and send multimodal crossing and high occupancy vehicle (HOV) lane sensor data to the center for further analysis and storage.<br>4 - The field element shall return sensor and CCTV system operational status to the controlling center.<br>5 - The field element shall return sensor and CCTV system fault data to the controlling center for repair. |

## 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

- 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,
  - human-computer interface issues,
  - Interface with other external systems,
  - security, maintainability, etc.

## Non-functional Requirements examples

- **Security:** Security requirements define the *security attributes of the system, restricting access to features* or data to certain users and *protecting the privacy of data entered* into the software.
- **Performance:** The performance *constraints specify the timing characteristics* of the software.
- **Usability:** Ease-of-use requirements address the factors that constitute *the capacity of the software to be understood, learned, and used by its intended users.*

# NONFUNCTIONAL REQUIREMENT EXAMPLES

## OPERATION GROUP

Describes the user needs for using the functionality. The user perceives the system as an electronic tool that helps to automate what would otherwise be done manually. From this point of view, the user is concerned with how well the system operates.

## ACCESS SECURITY

The extent to which the system is safeguarded against deliberate and intrusive faults from internal and external sources.

Examples

a. Employees shall be forced to change their password the next time they log in if they have not changed it within the length of time established as "password expiration duration."
b. Users must change the initially assigned login authentication information (password) immediately after the first successful login. The initial password may never be reused.
c. The payroll system shall ensure that the employee salary data can be accessed only by authorized users. The payroll system shall distinguish between authorized and non-authorized users.
d. Employees shall not be allowed to update their own salary information, and any such attempt shall be reported to the security administrator.
e. Only holders of current security clearance can enter the national headquarters building.
f. The access permissions for system data may only be changed by the system's data administrator.
g. Passwords shall never be viewable at the point of entry or at any other time.
h. Each unsuccessful attempt by a user to access an item of data shall be recorded on an audit trail.
i. Users shall receive notification of profile changes via preferred communication method of record when profile information is modified.

## ACCESSIBILITY

The extent to which the software system can be used by people with the widest range of capabilities to achieve a specified goal in a specified context of use.

## Non functional requirements

- Execution qualities - such as safety, security and usability, which are observable during operation (at run time).
- Evolution qualities, such as testability, maintainability, extensibility and scalability, which are embodied in the static structure of the system.

## Difference between functional and non functional requirements

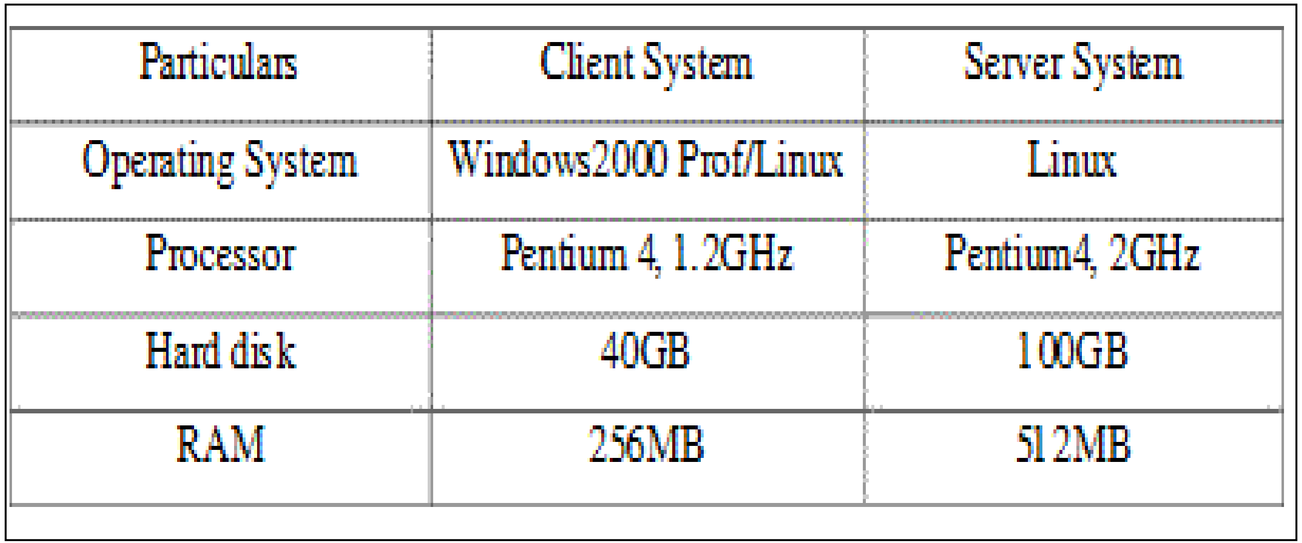
| Sl.No | Functional Requirement | Non-functional Requirement |
|---|---|---|
| 1. | Defines all the services or functions required by the customer that must be provided by the system | Defines system properties and constraints e.g. reliability, response time and storage requirements. Constraints are I/O device capability, system representations, etc. |
| 2. | It describes what the software should do. | It does not describe what the software will do, but how the software will do it. |
| 3. | Related to business. For example: Calculation of order value by Sales Department or gross pay by the Payroll Department | Related to improving the performance of the business. For example: checking the level of security. An operator should be allowed to view only my name and personal identification code. |
| 4. | Functional requirement are easy to test. | Nonfunctional requirements are difficult to test |
| 5. | Related to the individual system features | Related to the system as a whole |
| 6. | Failure to meet the individual functional requirement may degrade the system | Failure to meet a non-functional requirement may make the whole system unusable. |

## 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.

## Examples of constraints

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



| Particulars | Client System | Server System |
|---|---|---|
| Operating System | Windows2000 Prof/Linux | Linux |
| Processor | Pentium 4, 1.2GHz | Pentium4, 2GHz |
| Hard disk | 40GB | 100GB |
| RAM | 256MB | 512MB |

## Organization of the SRS Document

- Introduction.
- Functional Requirements
- Nonfunctional Requirements
  - External interface requirements
  - Performance requirements
- Constraints

## 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.

## 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. 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:
    - 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.

## 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.
- 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.
- 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

## What is a Decision tree

- A decision tree is a graph that uses a branching method to illustrate every possible outcome of a decision

# 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.

- 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.

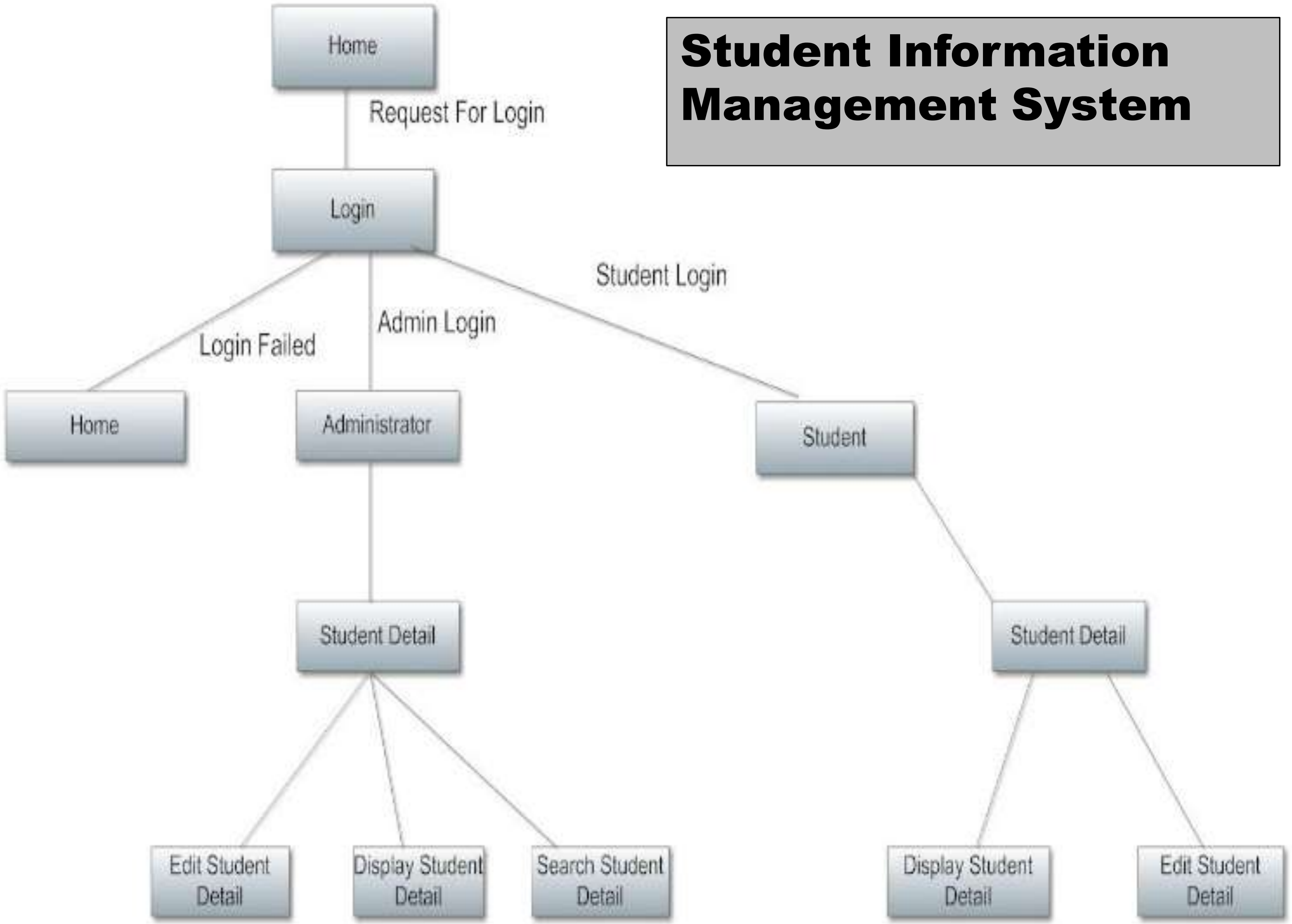
## Example(cont.)

- 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.

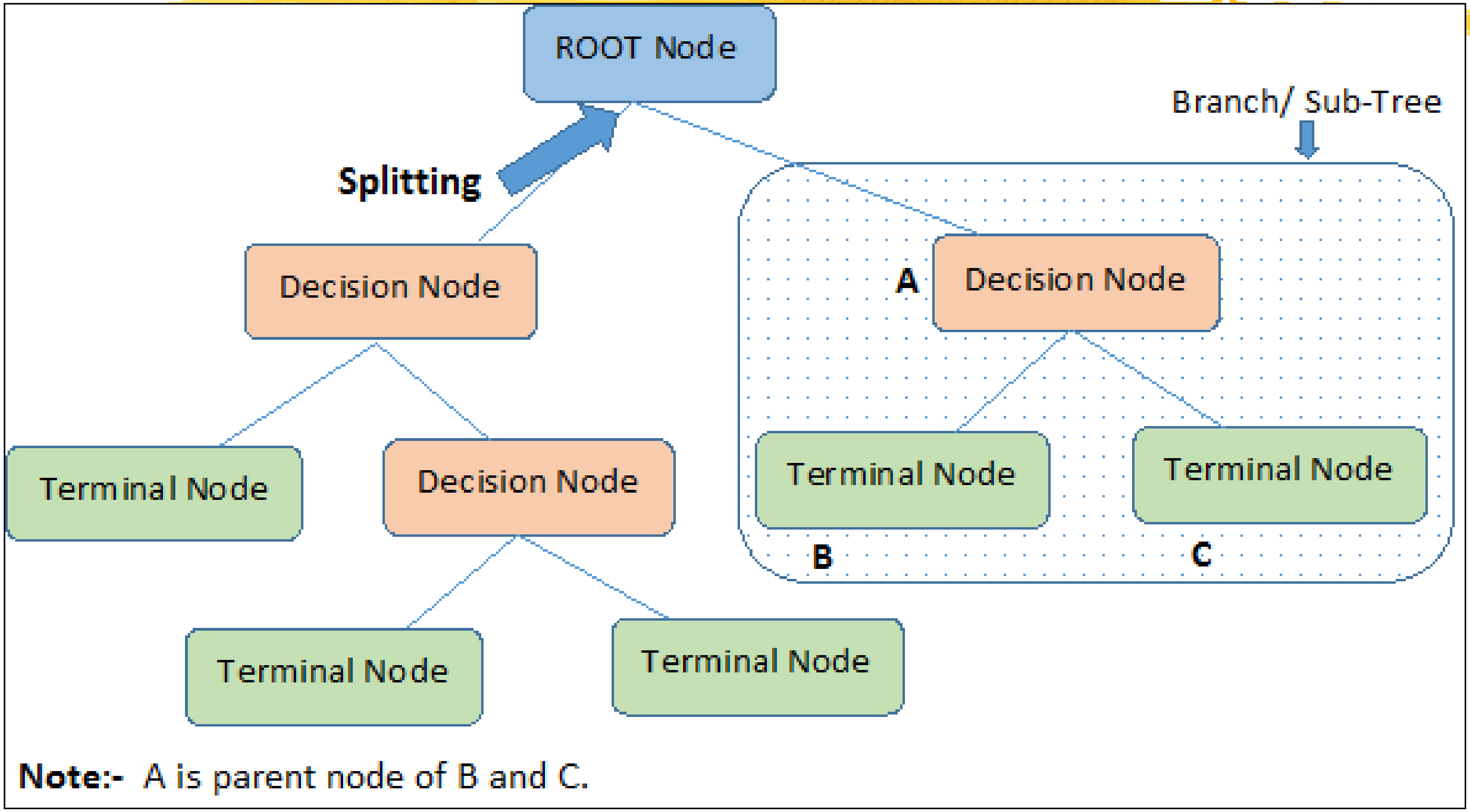
## Example(cont.)

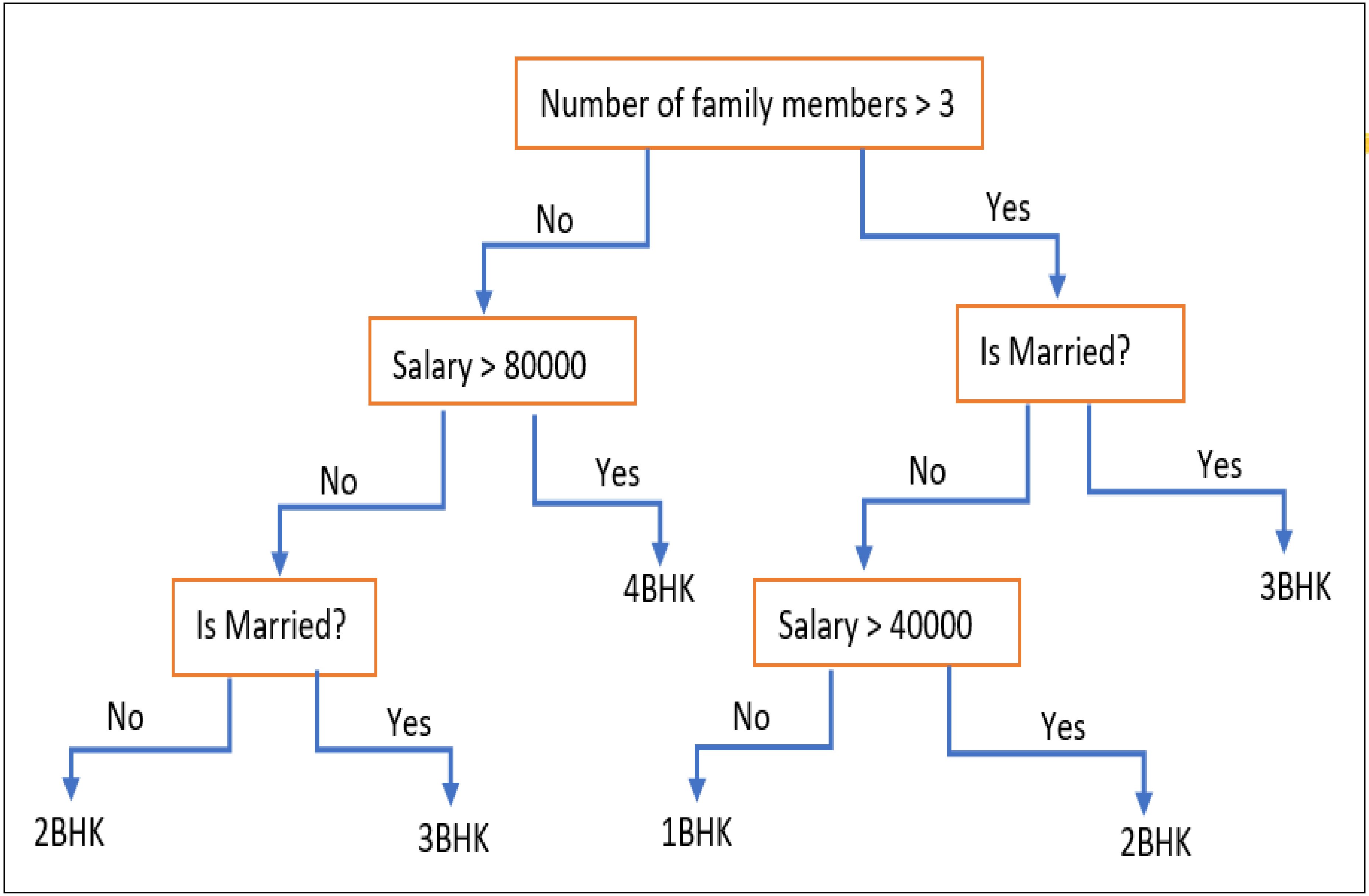
- If the cancel membership option is selected and the name of a valid member is entered,
  - the membership is cancelled,
  - a cheque for the balance amount due to the member is printed
  - the membership record is deleted.

# Student Information Management System



# Decision Tree as we understand




Number of family members > 3
No
Yes
Salary > 80000
Is Married?
No
Yes
No
Yes
Is Married?
4BHK
Salary > 40000
3BHK
No
Yes
No
Yes
2BHK
3BHK
1BHK
2BHK

## Decision table

- Decision tables are a concise visual representation for specifying which actions to perform depending on given conditions.
- They are algorithms whose output is a set of actions.
- The information expressed in decision tables could also be represented as decision trees or in a programming language as a series of if-then-else and switch-case statements.

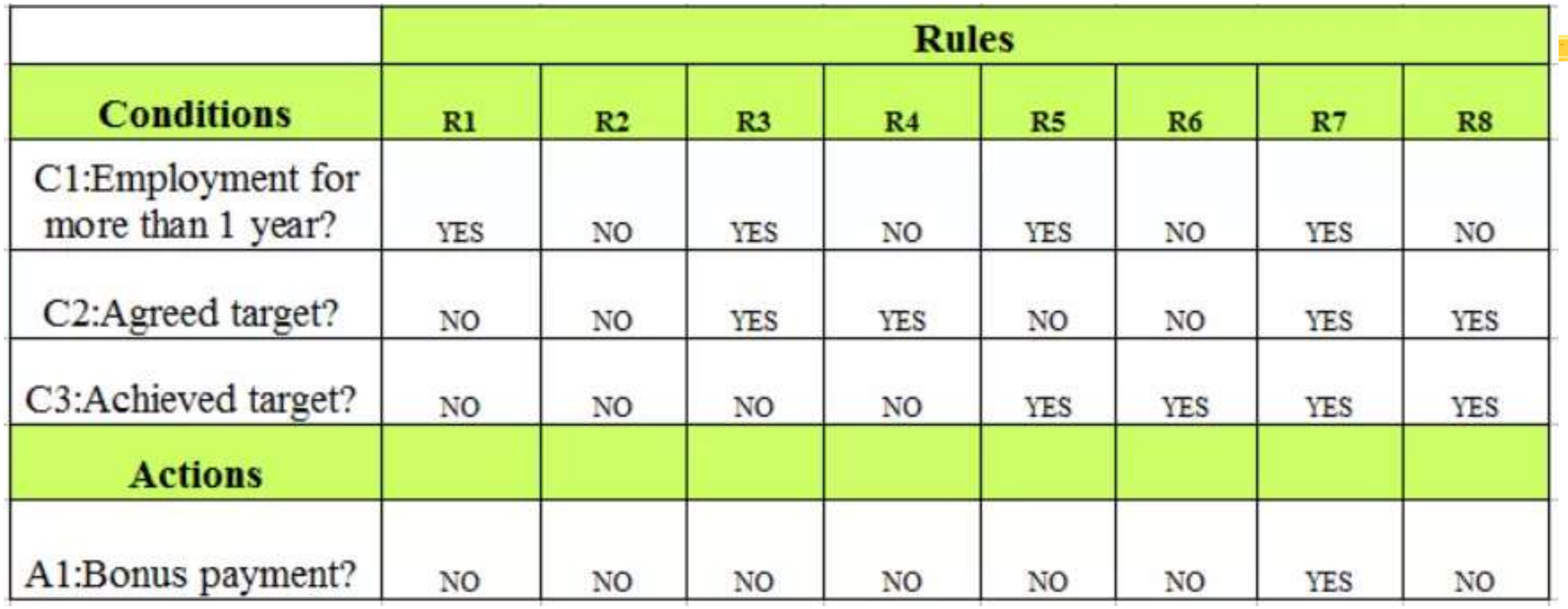
## Decision Table

- 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.
- 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.

## Algorithm for Bonus Payment



| | Rules | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Conditions** | **R1** | **R2** | **R3** | **R4** | **R5** | **R6** | **R7** | **R8** |
| C1:Employment for more than 1 year? | YES | NO | YES | NO | YES | NO | YES | NO |
| C2:Agreed target? | NO | NO | YES | YES | NO | NO | YES | YES |
| C3:Achieved target? | NO | NO | NO | NO | YES | YES | YES | YES |
| **Actions** | | | | | | | | |
| A1:Bonus payment? | NO | NO | NO | NO | NO | NO | YES | NO |

***Upper rows - conditions***
***Lower rows - Actions***
***Column - Rule***

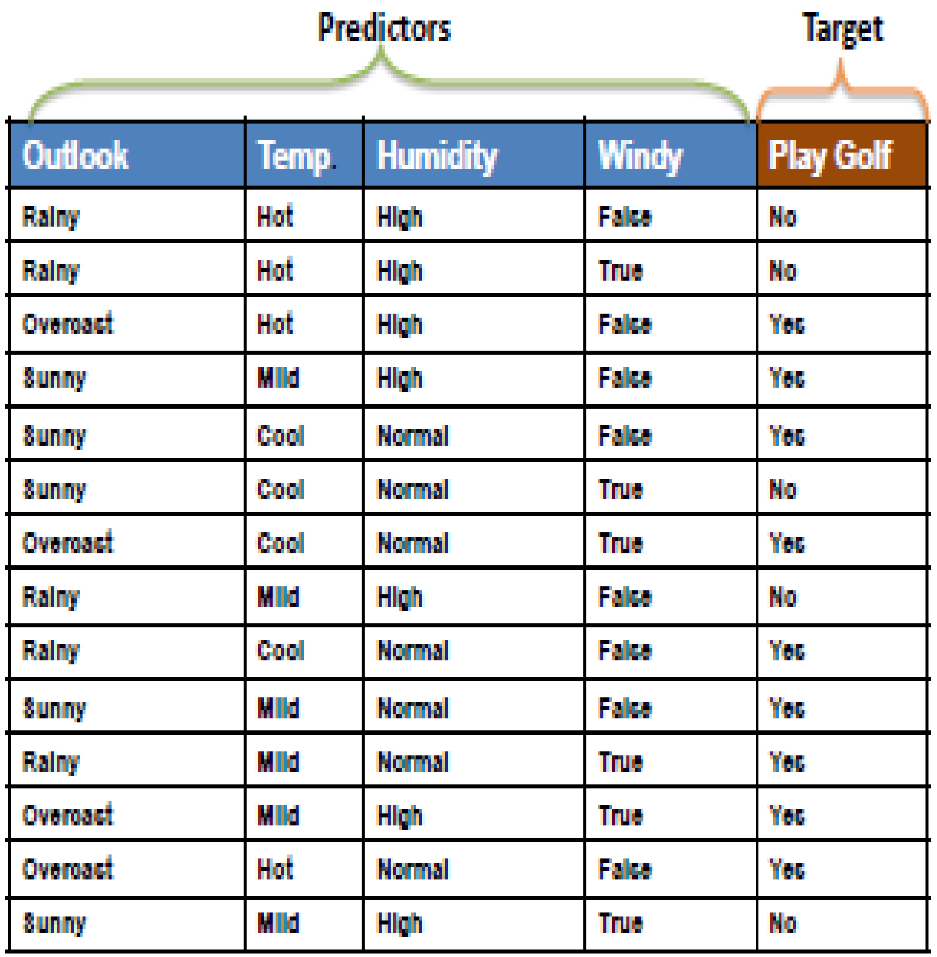
## Components of a Decision Table

rules

| | | R1 | R2 | R3 | R4 | R5 | R6 | R7 | R8 | |
|---|---|---|---|---|---|---|---|---|---|---|
| | C1 | T | T | T | T | F | F | F | F | |
| conditions | C2 | T | T | F | F | T | T | F | F | values of conditions |
| | C3 | T | F | T | F | T | F | T | F | |
| | a1 | x | | | x | x | | | x | |
| | a2 | x | | | | | | | x | |
| actions | a3 | | x | | | | x | | | actions taken |
| | a4 | | | x | x | | | x | x | |
| | a5 | x | | | x | | | | | |

*Read a Decision Table by columns of rules : R1 says when all conditions are T, then actions a1, a2, and a5 occur*

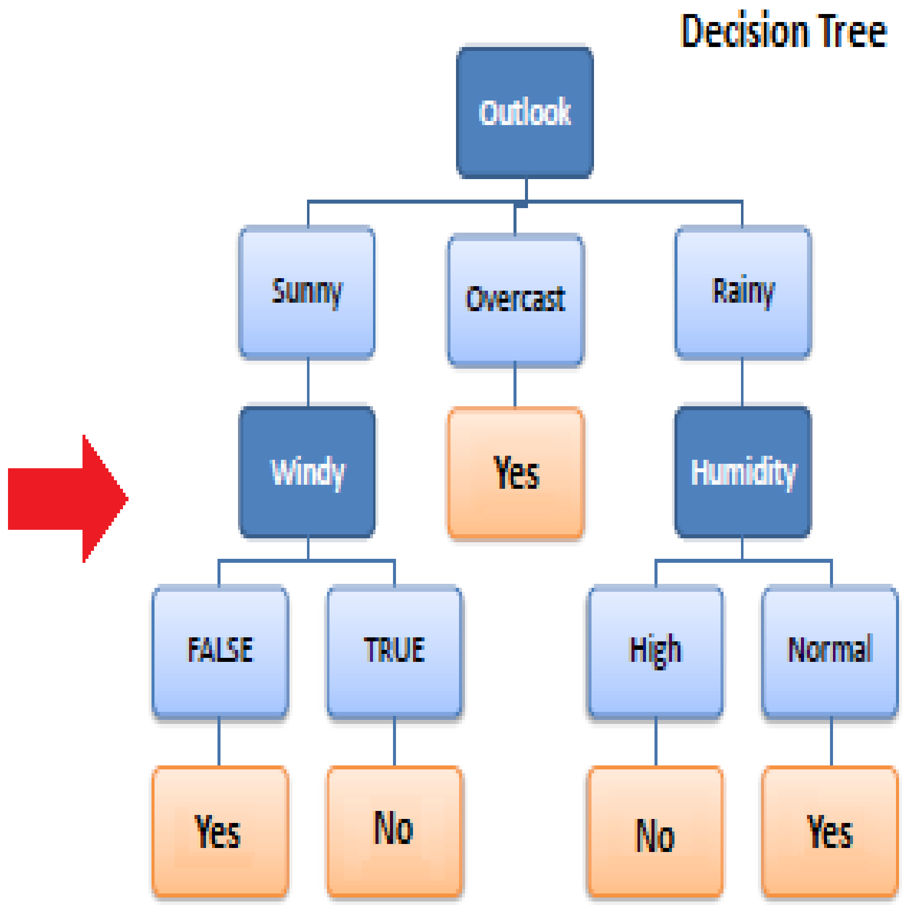
# Table to Tree ?

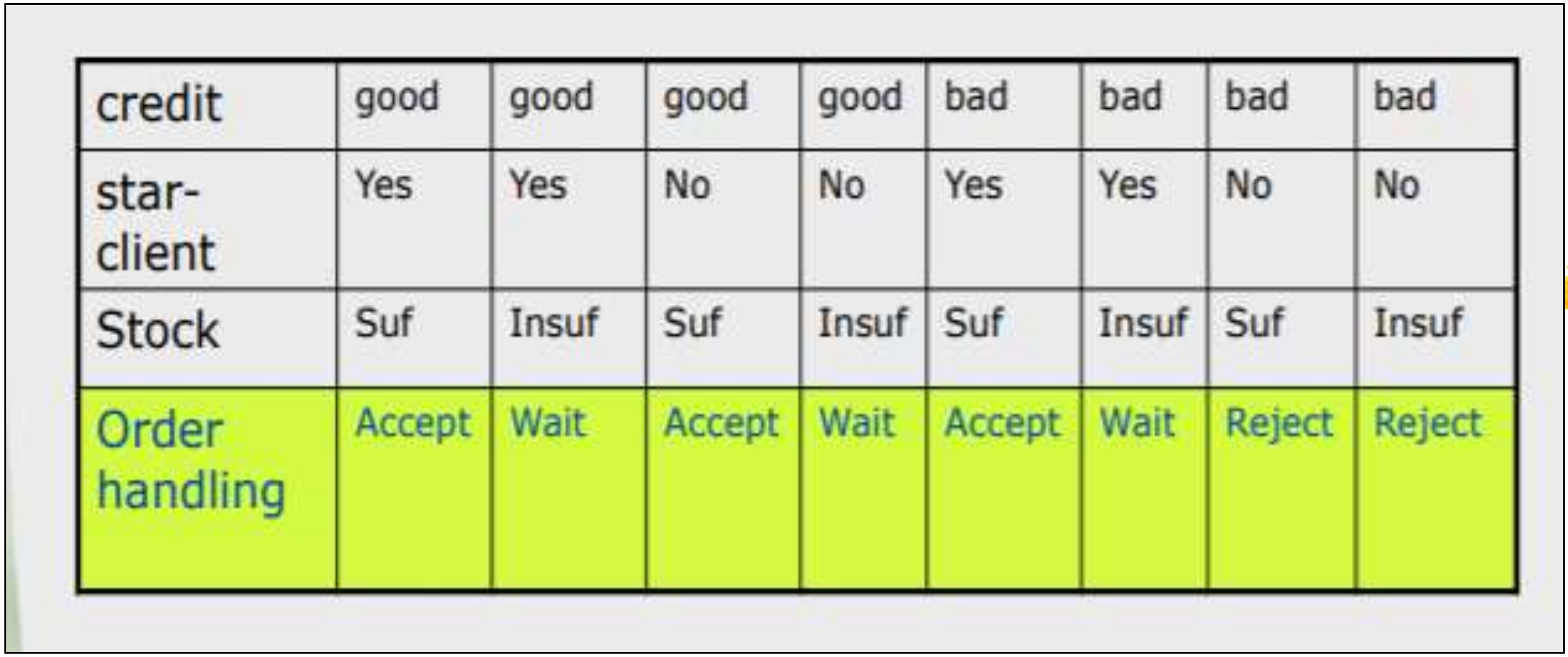


Predictors

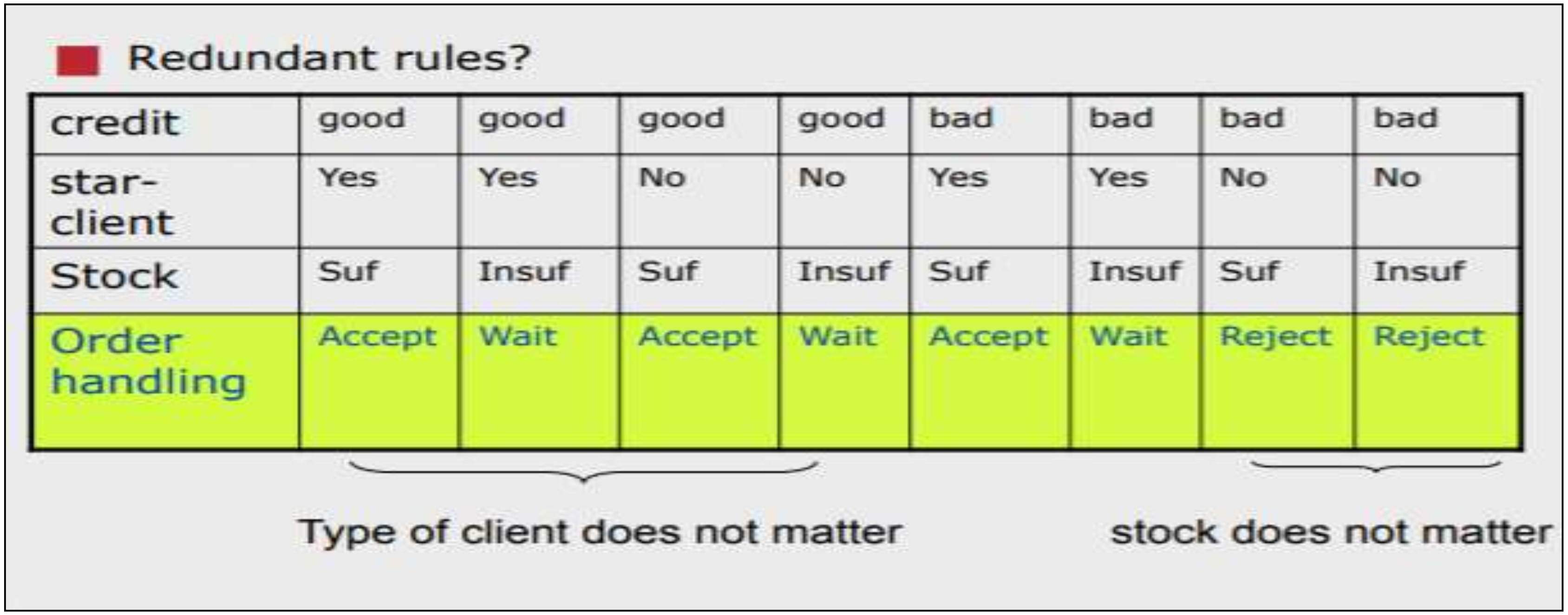
Target

| Outlook | Temp. | Humidity | Windy | Play Golf |
|---|---|---|---|---|
| Rainy | Hot | High | False | No |
| Rainy | Hot | High | True | No |
| Overcast | Hot | High | False | Yes |
| Sunny | Mild | High | False | Yes |
| Sunny | Cool | Normal | False | Yes |
| Sunny | Cool | Normal | True | No |
| Overcast | Cool | Normal | True | Yes |
| Rainy | Mild | High | False | No |
| Rainy | Cool | Normal | False | Yes |
| Sunny | Mild | Normal | False | Yes |
| Rainy | Mild | Normal | True | Yes |
| Overcast | Mild | High | True | Yes |
| Overcast | Hot | Normal | False | Yes |
| Sunny | Mild | High | True | No |





| credit | good | good | good | good | bad | bad | bad | bad |
|---|---|---|---|---|---|---|---|---|
| star-client | Yes | Yes | No | No | Yes | Yes | No | No |
| Stock | Suf | Insuf | Suf | Insuf | Suf | Insuf | Suf | Insuf |
| Order handling | Accept | Wait | Accept | Wait | Accept | Wait | Reject | Reject |



- Redundant rules?

| credit | good | good | good | good | bad | bad | bad | bad |
|---|---|---|---|---|---|---|---|---|
| star-client | Yes | Yes | No | No | Yes | Yes | No | No |
| Stock | Suf | Insuf | Suf | Insuf | Suf | Insuf | Suf | Insuf |
| Order handling | Accept | Wait | Accept | Wait | Accept | Wait | Reject | Reject |

Type of client does not matter

stock does not matter

## Decision table simplified

| credit | good | | bad | | bad |
|---|---|---|---|---|---|
| star-client | - | | Yes | | No |
| Stock | Suf | Insuf | Suf | Insuf | - |
| Order handling | Accept | Wait | Accept | Wait | Reject |

# 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

- 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

- A formal software specification is a statement expressed in a language whose vocabulary, syntax, and semantics are formally defined.

- The need for a formal semantic definition means that the specification languages cannot be based on natural language;

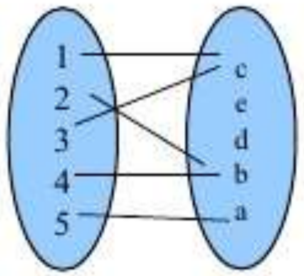
- **It must be based on mathematics.**

## Formal Specification

- Describe (specify) software requirements using mathematical notation – mostly set and logic operations
- Because of the precise meaning of the Maths symbols, one is forced to think in a more thorough manner
- Demands clear understanding of the requirements, so helps identify ambiguities and inconsistencies at an early stage in the software lifecycle thus potentially saving many man-hours further on

## What are Formal Specification Methods?

$$\Sigma$$



Pre / post

1. Mathematical notation

2. Well-defined data structures:
- Sets
- Relations
- Functions

3. Clear Behavior specification, based on logical expressions

## Acceptance of formal methods

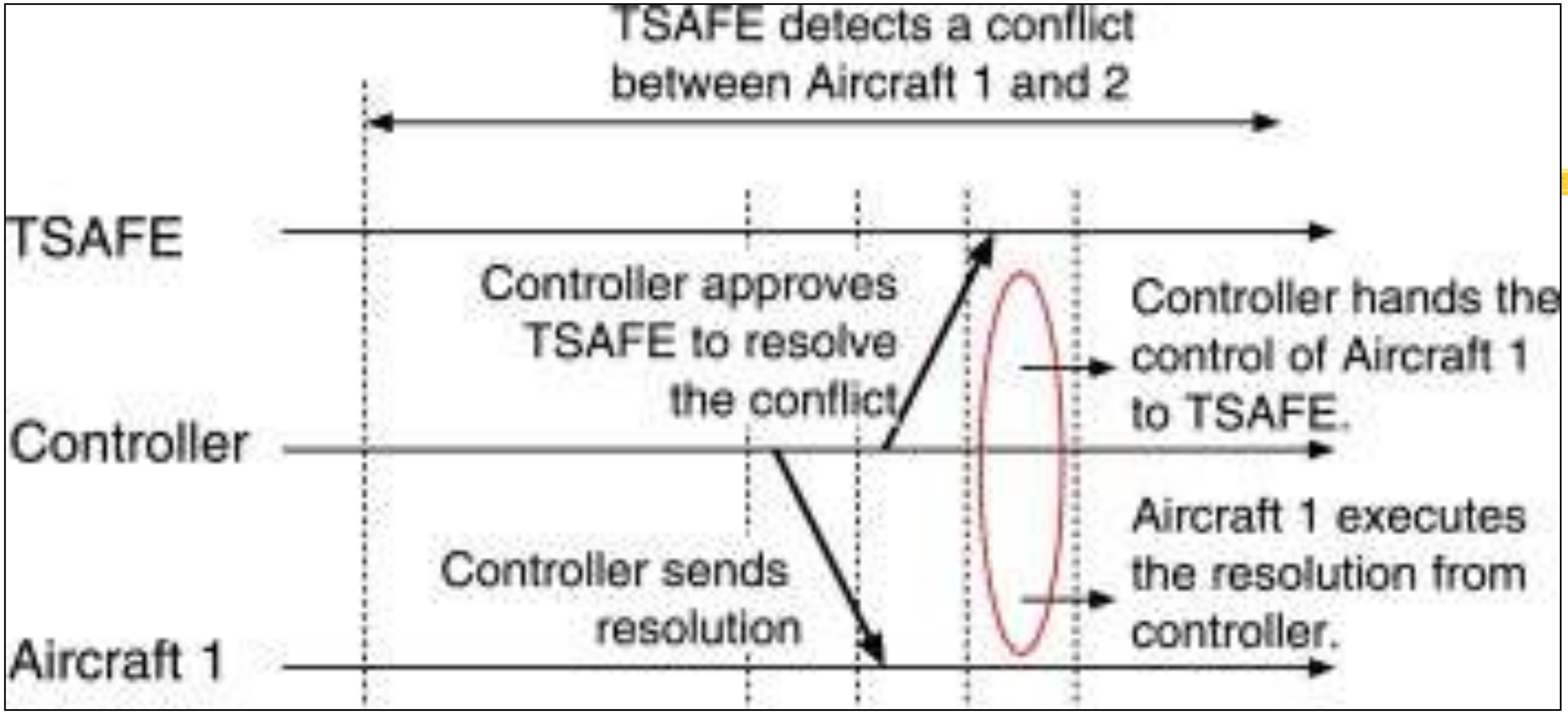
- Formal methods have not become mainstream software development techniques as was once predicted
  - Other software engineering techniques have been successful at increasing system quality. Hence the need for formal methods has been reduced
  - Market changes have made time-to-market rather than software with a low error count the key factor. Formal methods do not reduce time to market
  - The scope of formal methods is limited. They are not well-suited to specifying and analysing user interfaces and user interaction
  - Formal methods are hard to scale up to large systems

# Use of formal methods

- **Their principal benefits are in reducing the number of errors in systems** so their main area of applicability is **critical systems:**
  - **Air traffic control information systems,**
  - **Railway signalling systems**
  - **Spacecraft systems**
  - **Medical control systems**

- In this area, the **use of formal methods is most likely to be cost-effective**

- Formal methods have limited practical applicability


TSAFE detects a conflict between Aircraft 1 and 2
TSAFE
Controller approves TSAFE to resolve the conflict
Controller hands the control of Aircraft 1 to TSAFE.
Controller
Aircraft 1 executes the resolution from controller.
Controller sends resolution
Aircraft 1

# Formal 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

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

## Semiformal Specification

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

## Executable Specification Language

- If specification is expressed in formal language:
  - it becomes possible to execute the specification to provide a system prototype.
- However, executable specifications are usually slow and inefficient.

## Executable Specification Language

- Executable specifications only test functional requirements:
  - If  non-functional requirements are important for some product,
    - the utility of an executable specification prototype is limited.

## 4GLs

- 4GLs (Fourth Generation Languages) are examples of
  - executable specification languages.
  - Eg. SQL, Mathematica

- 4GLs are successful
  - because there is a lot of commonality across data processing applications.

# 4GLs

- 4GLs rely on software reuse
  - where common abstractions have been identified and parameterized.

- Rewriting 4GL programs in higher level languages:
  - result in upto 50% lower memory requirements
  - also the programs run upto 10 times faster.

## 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

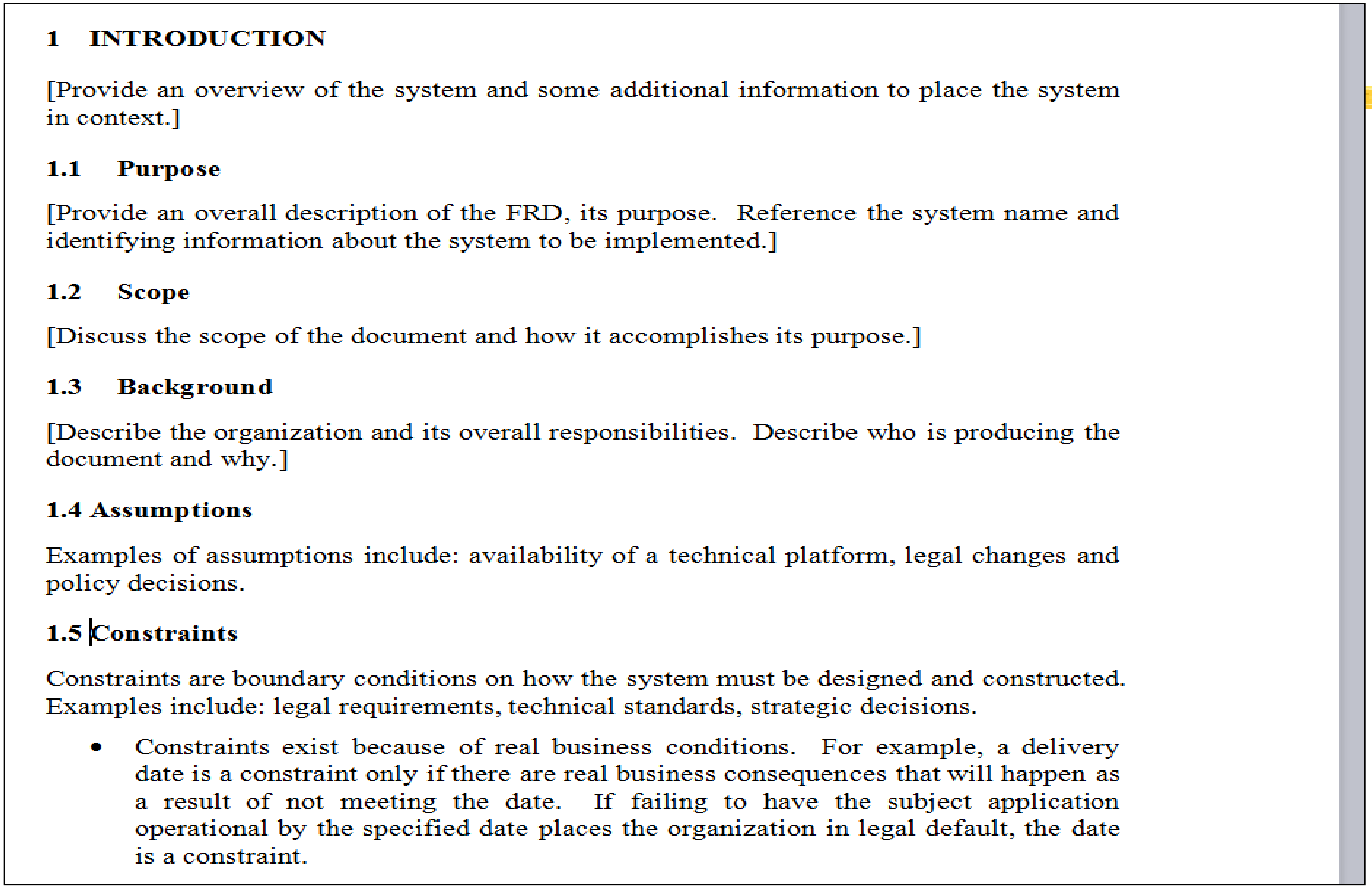
- Main components of SRS document:
  - functional requirements
  - nonfunctional requirements
  - constraints
- Techniques to express complex logic:
  - Decision tree
  - Decision table

## Summary

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

## Activity 2 - write SRS

# SRS template



## 1 INTRODUCTION

[Provide an overview of the system and some additional information to place the system in context.]

### 1.1 Purpose

[Provide an overall description of the FRD, its purpose. Reference the system name and identifying information about the system to be implemented.]

### 1.2 Scope

[Discuss the scope of the document and how it accomplishes its purpose.]

### 1.3 Background

[Describe the organization and its overall responsibilities. Describe who is producing the document and why.]

### 1.4 Assumptions

Examples of assumptions include: availability of a technical platform, legal changes and policy decisions.

### 1.5 Constraints

Constraints are boundary conditions on how the system must be designed and constructed. Examples include: legal requirements, technical standards, strategic decisions.

- Constraints exist because of real business conditions. For example, a delivery date is a constraint only if there are real business consequences that will happen as a result of not meeting the date. If failing to have the subject application operational by the specified date places the organization in legal default, the date is a constraint.

# Functional requirements

Functional Requirements should include:

- Descriptions of data to be entered into the system
- Descriptions of operations performed by each screen
- Descriptions of work-flows performed by the system
- Descriptions of system reports or other outputs
- Who can enter the data into the system
- How the system meets applicable regulatory requirements

## Example

| Section/ Requirement ID | Requirement Definition |
|---|---|
| FR1.0. | The system shall [parent requirement group 1]. |
| FR1.1 | The system shall [child/parent requirement]. |
| FR1.1.1 | The system shall [child requirement]. |
| FR1.1.2 | The system shall [child requirement]. |

## Non Functional requirements

- Performance – for example Response Time, Throughput, Utilization, Static Volumetric.
- Scalability.
- Capacity.
- Availability.
- Reliability.
- Recoverability.
- Maintainability.

And also write constraints