Learning Resource

On

Software Engineering

Chapter-4
Requirement Analysis and Specification

Prepared By:
Kunal Anand, Asst. Professor
SCE, KIIT, DU, Bhubaneswar-24

Chapter Outcomes:

After completing this chapter successfully, the students will be able to:

- Define the goal of Requirement Specification and Analysis.
- Explain different activities of requirement analysis and specification.
- Identify the different users of SRS along with their purposes.
- List the charactristics of SRS document.
- Explain the components of SRS document.
- Write functional requirements for a given problem statement.
- Illustrate complex processing logic using decision tree and decision table.
- Describe Software Configuration Management (SCM).

Organization of the Chapter:

- Introduction
- Requirement Analysis
  - Requirement Gathering
  - Requirement Analysis
- Requirements Specification
- Software Requirement Specification (SRS) document
- Decision Table
- Decision Tree
- Software Configuration Management

Introduction to Requirement Analysis and Specification

- Many projects fail because the team directly start implementing the system without determining whether they are building what the customer really wants!

- It is very important to understand the requirements of the system to be developed before starting the actual development.

- **Requirement Analysis and Specification** phase in the SDLC helps in obtaining a clear understanding of the requirements of the system to be developed.

contd..

- **Goals** of requirement analysis and specification phase:
  - To fully understand the user requirements
  - To remove anomalies like inconsistency, incompleteness, and ambiguity from the collected requirements
  - To document the requirements properly in a SRS document

- Consists of **two distinct activities**:
  - Requirement Gathering and Analysis
  - Requirement Specification

contd..

- An individual or the group of individuals who undertakes requirement analysis and specification is known as **system/business analyst**.
- They collect data pertaining to the product, analyse the collected data, and write **Software Requirements Specification (SRS)** document.
- Software Requirements Specification (SRS) Document is the tangible outcome of this phase.
  - The SRS is a formal document that contains the **well defined requirements** to be developed.
  - Once the SRS is prepared, it is shared with the client for their **review and necessary approval**. The actual development is not started till the SRS gets approved by the client.
  - If the client is satisfied with the SRS then they approve and send back the approved copy to the developers and retain a copy with themselves.

Requirement Analysis

- Requirement Analysis consists of two main activities:
  - Requirement gathering
  - Requirement Analysis

Requirement Gathering

- 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.
- If the project is to automate some existing manual procedures, the task of the system analyst becomes a bit easier.
- Analyst can immediately obtain input and output formats, accurate details of the operational procedures.
- However, 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.

Requirement Analysis

- After gathering all the requirements, they are analysed:
  - To clearly understand the user requirements,
  - To detect and remove inconsistencies, ambiguities, and incompleteness.
- In **Inconsistent requirement**, some part of the requirement contradicts with some other part.
  - For ex: One customer says turn off heater and open water shower when temperature > 100 degree. Another customer says turn off heater and turn ON cooler when temperature > 100 degree.
- In **Incomplete requirement**, some requirements are omitted due to oversight of the users or analyst.
  - For ex: The analyst has not recorded that when temperature falls below 90 C whether the heater should be turned ON or the water shower should be turned OFF

contd..

- 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 can be very subtle and can escape even the 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:
  - To **systematically organize** the requirements arrived during requirement analysis
  - To **document** requirements properly.

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

SRS as a Contract/Reference Document

- SRS document acts as a **contract document**. It 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 to the SRS document.
  - Once customer agrees to the SRS document, the 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.
- Again, the SRS document acts as a **reference document** for different kind of users who use the SRS in one way or another for their work.

Users of SRS Document

- Following are the main users of the SRS document:
  - **Developers**
    - Designers
    - Programmers
    - Testers
  - **Maintenance staff**
  - **Project Managers**
  - **Training Manual Writers**
  - **Client/Customer**

Properties of a Good SRS document

- It should be **concise** i.e., 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, complete, and unambiguous.**
- It should be **traceable** i.e., one 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** i.e., one should be able to measure the outcome using any suitable mechanism. e.g., **"system should be user friendly" is not verifiable.**

Bad SRS Document

- **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.

contd..

- **Overspecification:**
  - Addressing **"how to do"** 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:** Reference to the aspects of problem defined only later on in the text.
- **Wishful Thinking:** Description of the aspects for which realistic solutions will be hard to find.

SRS Document (contd..)

- The SRS document is known as **Black-Box specification** because 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.

Fig. 4.1: Black-box view of SRS

- The SRS document concentrates on **"what to do"** and carefully avoids the **"how to do"** aspects.
- As, it serves as a **contract** between development team and the customer and hence it should be carefully written.

Components of SRS document

- It is desirable to consider every system performing a **set of functions {fi}.** Each function **fi** is considered transforming a set of input data to corresponding output data.

Fig. 4.2: High-level functionality

- The SRS document contains following components:
  - **Introduction**
  - **Functional Requirements (FR)**,
  - **Non-functional requirements (NFR)**,
    - External interface
    - Performance related
  - **Constraints on the system**.

Functional Requirements (FR)

- Functional requirements describe a set of **high-level requirements.**
- Each high-level requirement takes some input data from the user and returns some data as output 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

contd..

- **FR** represents an application of the system to be developed in the form of a function.
- **Example:**
  - **FR1: Search a Book**
    - **Input:** an author's name:
    - **Output:** details of the author's books and the locations of these books in the library.

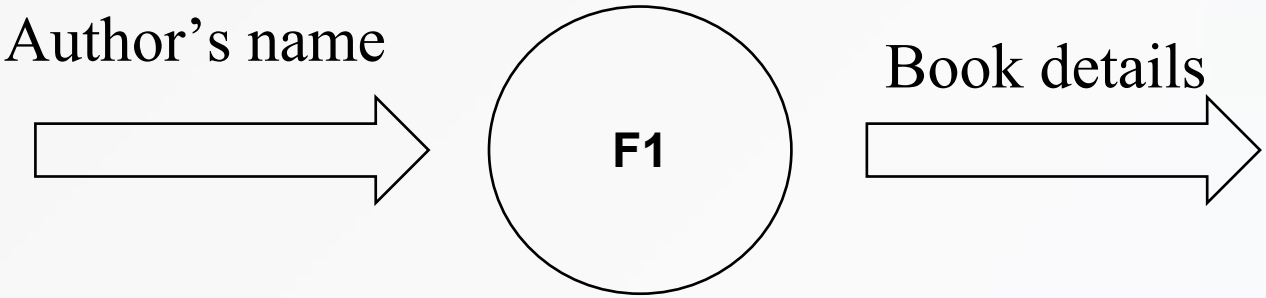


Fig. 4.3: Book Search as a Functional requirement

Example: Functional Requirements

List all functional requirements with proper numbering.

- **FR1: Search a Book**
  - Once the user selects the **"search"** option, he/she is asked to **enter the keywords**.
  - The system should **output details** of all books whose title or author name matches any of the keywords entered.
  - **Details include:** Title, Author Name, Publisher name, Year of Publication, ISBN Number, Catalogue Number, Location in the Library.
- **FR2: Renew a Book**
  - When the **"renew"** option is selected, the user is asked to enter his/her **membership number and password**.
  - After **password validation**, the list of the books borrowed by the user is displayed.
  - The user can **renew** any of the books by clicking in the corresponding renew box.

FR1: Search a Book

- **FR.1.1:**
  - **Input:** "search" option,
  - **Output:** user prompted to enter the key words.

- **FR.1.2:**
  - **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, Catalogue Number, Location in the Library.
  - **Processing:** Search the book list for the keywords entered.

FR2: Renew a Book

- **FR.2.1:**
  - **Input:** "renew" option selected,
  - **Output:** user prompted to enter his/her membership number and password.

- **FR.2.2:**
  - **Input:** membership number and password
  - **Output:**
    - list of the books borrowed by user is 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.

contd..

- **FR.2.3:**
  - **Input:** user choice for renewal of the books issued to him through mouse clicks in the corresponding renew box.
  - **Output:** Confirmation of the books renewed
  - **Processing:** Renew the books selected by the user in the borrower list.

Non-functional Requirements

- Characteristics of the system that **can not be expressed as functions** are known as non-functional requirements.
- Non-functional requirements include
  - Reliability issues
  - Performance issues
  - Human-computer interface issues
  - Interface with other external systems
  - Security
  - Maintainability
  - Portability etc.

Constraints

- Constraints describe things that the system **should or should not do.**
- For example:
  - Hardware to be used,
  - Operating system
  - DBMS to be used
  - Capabilities of I/O devices
  - Standards compliance
  - Data representations by the interfaced system
  - how fast the system can produce results so that it does not overload another system to which it supplies data, etc.

Representation of Complex Processing Logic

- **Decision Tree:** A decision tree is a graph that uses a branching method to illustrate **every possible outcome of a decision.**
- In **Decision tree:**
  - edges 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: A **Library Membership automation Software (LMS)** that supports the following three options:
  - Adding a new member,
  - Renewal of an existing member
  - cancel membership

Example: Library Management System (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,
  - The LMS software asks the member's name and his membership number and checks whether he/she 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 for LMS

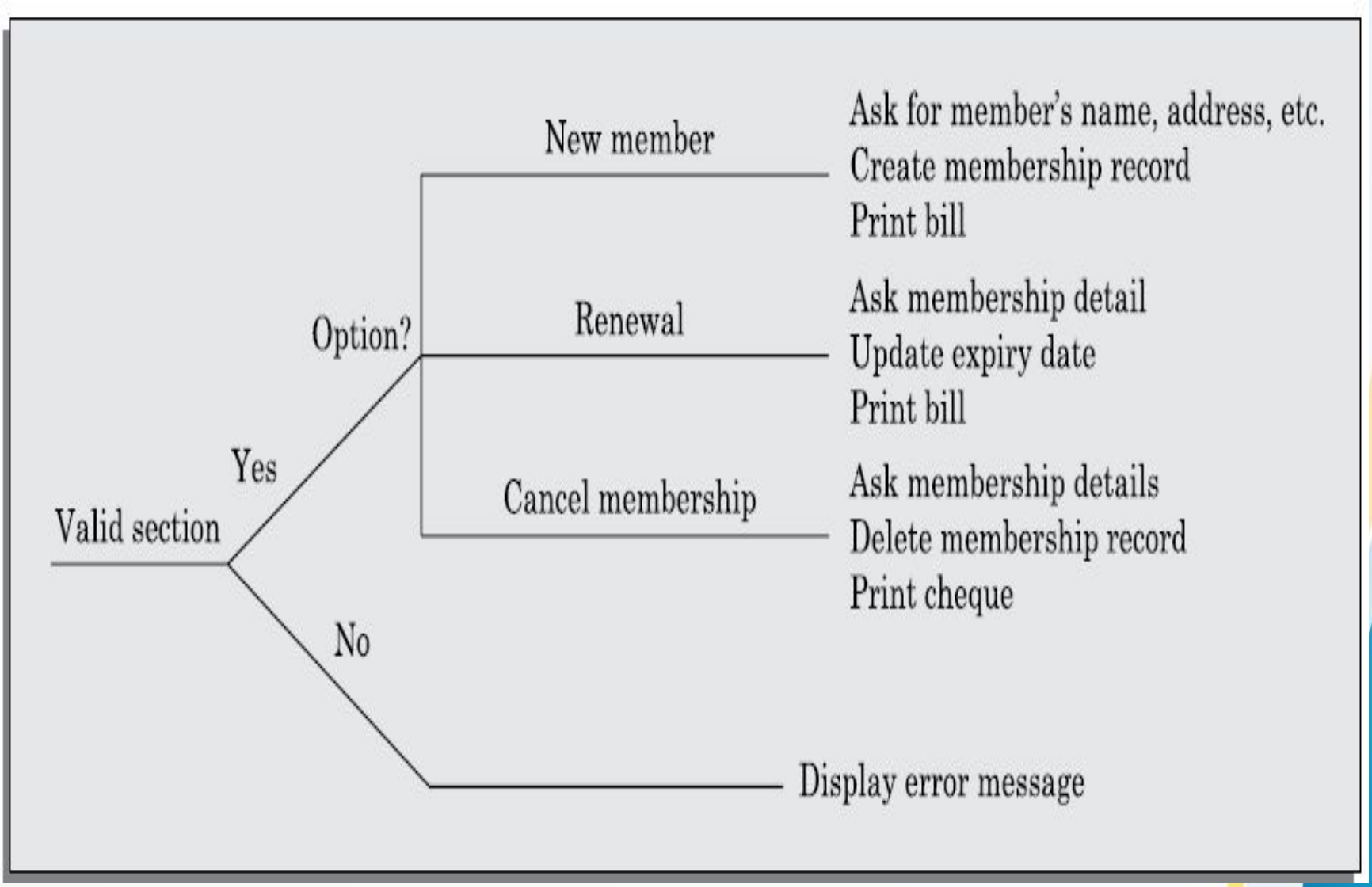


Fig. 4.4: Decision tree for LMS

Decision Table

- **Decision Table:** Decision table is a **concise visual representation** for specifying the **actions to be performed depending on the 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 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.

Decision Table for 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	×			
Ask member's name, etc.		×		
Build customer record		×		
Generate bill		×	×	
Ask membership details			×	×
Update expiry date			×	
Print cheque				×
Delete record				×

Fig. 4.5: Decision table for LMS

Comparison of Decision Table and Decision Tree

- 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.

Software Configuration Management (SCM)

- The **deliverables** of a software product consist of a **number of objects**, e.g., source code, design document, SRS document, test document, user's manual, etc.
- These objects are **modified** by many software engineers through out development cycle.
- The state of each object **changes** as bugs are **detected and fixed** during development .
- The **configuration** of the software is the **state of all project deliverables at any point of time.**
- **SCM** deals with effectively **tracking and controlling** the **configuration of a software** during its life cycle.

Version vs. Revision vs. Release of Software

- A **new version** results from a **significant change** in functionality, technology, or the platform
  - **Example:** one version of a software might be Unix-based, another version might be Windows based.
- A **revision** results from **bug fix, minor enhancements** to the functionality, usability.
- A **new release** of software product is done whenever we get a **revision or a new version** is an improved system replacing the old one.
- A **new release** of a software product is described as **Version m, Revision n**; or simple **m.n; where m is the version and n is the revision.**

Necessity of SCM

- To **control access to deliverable objects** with a view to avoiding the following problems
  - **Inconsistency** problem when the objects are replicated.
  - Problems associated with **concurrent access**.
  - Providing a **stable development environment**.
  - System **accounting and status** information.
  - **Handling variants**. If a bug is found in one of the variants, it has to be fixed in all variants

SCM Activities

- **SCM activities**
  - **Configuration identification** : deciding which objects (configuration items) are to be kept track of.
  - **Configuration control (CC) :** ensuring that changes to a system happen without **ambiguity**.

- **Baseline:** When an effective SCM is in place, **the manager freezes the objects to form a baseline.**
  - A **baseline** is the status of **all the objects under configuration control**. When any of the objects under configuration control is **changed** , a **new baseline is formed**.

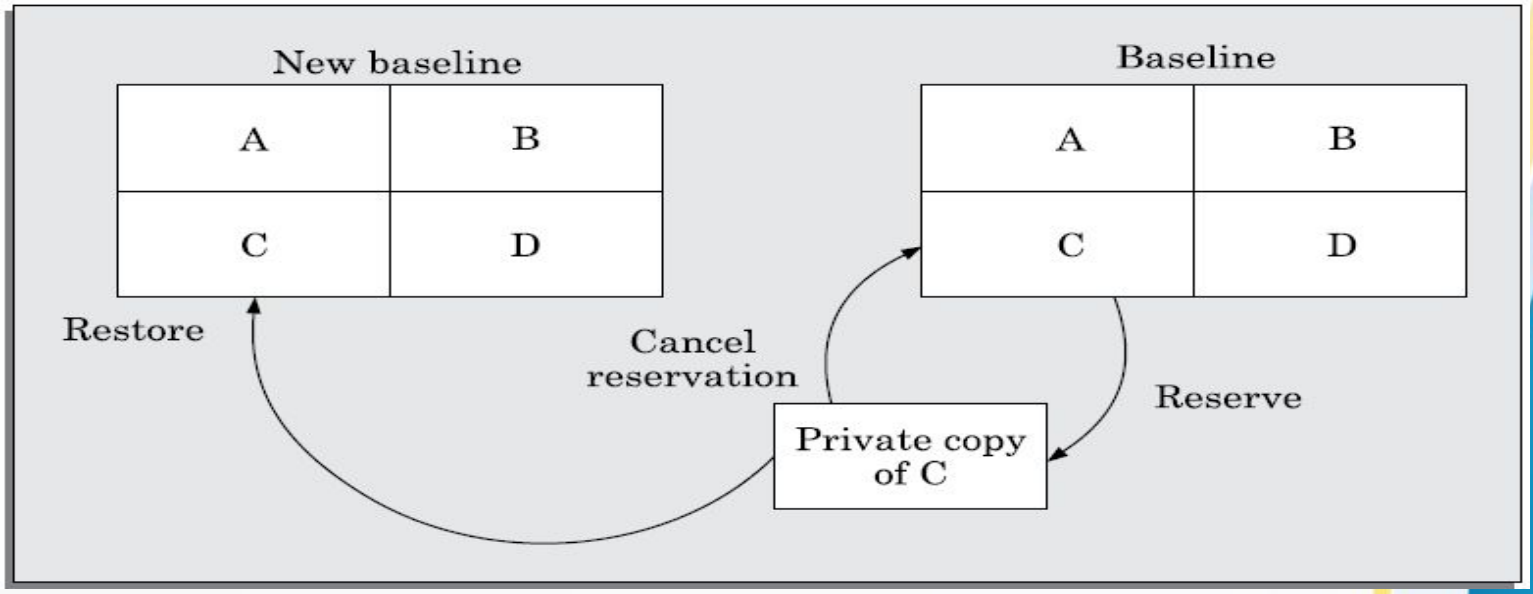
Configuration Item identification

- **Categories of objects:**
  - **Uncontrolled objects:** are not subjected to CC
  - **Controllable objects** include both **controlled and precontrolled objects**; examples: SRS document, Design documents, Source code , Test cases
    - **Controlled objects:** are under Configuration Control (CC) . Formal procedures are followed to change them.
    - **Precontrolled objects** are not yet under CC, but will eventually be under CC.

- The SCM plan is written during the project planning phase, and it lists all controlled objects.

Configuration Control

- **Configuration Control (CC):** process of managing changes to controlled objects. Allows authorized changes to the controlled objects and prevents unauthorized changes .
  - In order to change a controlled object such as a module, a developer can get a private copy of the module by a reserve operation .
  - Configuration management tools allow only one person to reserve a module at a time. Once an object is reserved, it does not allow any one else to reserve this module until the reserved module is restored .



Changing the Baseline

- When one needs to change an object under configuration control, he is provided with a copy of the base line item.
- The requester makes changes to his private copy.
- After modifications, updated item replaces the old item and a new base line gets formed .

Reserve and Restore operation in configuration control

- obtains a private copy of the module through a reserve operation.
- carries out all changes on this private copy.
- restoring the changed module to the baseline requires the permission of a change control board(CCB).
- Except for very large projects, the functions of the CCB are discharged by the project manager himself.