UNIT 3 SOFTWARE ENGINEERING CONCEPTS AND STANDARDS

Structure **Page Nos.**

3.0 INTRODUCTION

Software Engineering deals with the development of software. Hence, understanding the basic concepts of software is essential. Before beginning the development of the project, it is essential to you to refresh your concepts about software development and engineering. During your first and third semester (in courses MCS-021 and MCS-034), you have already studied different approaches of software development. This, will be a complete refresher for you, however, for details you can also refer to these courses. As you know the field of software engineering is related to the development of software. Large software needs systematic development unlike simple programs, which can be developed in isolation, and there may not be any systematic approach being followed. There is a difference between programming and Software Engineering. Software Engineering includes activities like cost estimation, time estimation, designing, coding, documentation, maintenance, quality assurance, testing of software etc. whereas programming includes only the coding part.

The important task in software development is developing and maintaining documentation. Documentation is a process that helps users of software and other people to use and interact with the system. Effective and accurate documentation is very necessary for the success of any system. Documentation becomes a part of each step of system development throughout the process of system development even before the documentation starts officially. During the process of system development, study reports and other necessary information are documented to help people involved in system development understand the process.

There are many kinds of documentation namely, analysis documentation, design documentation, interface documentation, internal program documentation and user-oriented documentation. We will learn how to develop these documents in this unit. In most of your lab manuals and practical exercises you have developed different programs/softwares according to the given problem, however in this course: Mini-Project we will not only develop program but also learn the real time challenges/problems in software development and engineering.

3.1 OBJECTIVES

After going through this unit, you should be able to:

5 learn about the various phases of software development life cycle;

6 understand software development models;

7 understand the importance and concept of documentation;

8 learn about various documents and process of documentation;

9 understand application of various standards of documentation processes;

10 differentiate between various documentation processes;

11 understand the relation between documentation and quality of software, and

12 understand the good practices for documentation process.

3.2 SOFTWARE DEVELOPMENT APPROACHES

The software industry considers software development as a process. According to Booch and Rumbaugh, "A process defines who is doing what, when and how to reach a certain goal"? Software engineering is a field, which combines process, methods and tools for the development of software. The concept of process is the main step in software engineering approach. Thus, a software process is a set of activities. The various steps (called phases) which are adopted in the development of this process are collectively termed as Software Development Life Cycle (SDLC). The various phases of SDLC are discussed below. Normally, these phases are performed lineally or circularly, but it can be changed according to the project as well. The software is also considered as a product and its development as a process. Thus, these phases can be termed as Software Process Technology. In general, different phases of SDLC are defined as the following:

13 Analysis

14 Design

15 Coding

16 Testing.

As you have already learnt about different software development models (in the course MCS-034), in details, here, briefly, we provide the description of linear and spiral approach to software development which will be useful to you in your Mini- project. The following are some of the models adopted to develop software:

3.2.1 Waterfall Model

It is the simplest, oldest and most widely used process model. In this model, each phase of the life cycle is completed before the start of a new phase. It is actually the first engineering approach of software development. *Figure 1* depicts the Water Fall Model.

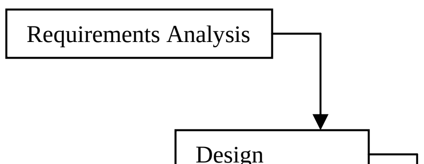


Figure 1: Waterfall model

The functions of various phases are discussed in software process technology.

The waterfall model provides a systematic and sequential approach to software development and is better than the build and fix approach. But, in this model, complete requirements should be available at the time of commencement of the project, but in actual practice, the requirements keep on originating during different phases. The waterfall model can accommodate new requirements only in the maintenance phase. Moreover, it does not incorporate any kind of risk assessment. In the waterfall model, a working model of software is not available. Thus, there is no way of judging the problems of the software in-between different phases.

A slight modification of the waterfall model is a model with feedback. Once software is developed and is operational, then the feedback to various phases may be provided.

3.2.2 Spiral Model

This model can be considered as the model, which combines the strengths of various other models. Conventional software development processes do not take uncertainties into account. Important software projects have failed because of unforeseen risks. The other models view the software process as a linear activity whereas this model considers it as a spiral process. This is made by representing the iterative development cycle as an expanding spiral.

The following are the primary activities in this model:

Finalising Objective: The objectives are set for the particular phase of the project.

Risk Analysis: The risks are identified to the extent possible. They are analysed and necessary steps are taken.

Development: Based on the risks that are identified, an SDLC model is selected and followed.

Planning: At this point, the work done till this time is reviewed. Based on the review, a decision regarding whether to go through the loop of spiral again or not will be decided. If there is need to do so, then planning is done accordingly.

In the spiral model, these phases are followed iteratively. *Figure 2* depicts the Boehm's Spiral Model (IEEE, 1988).

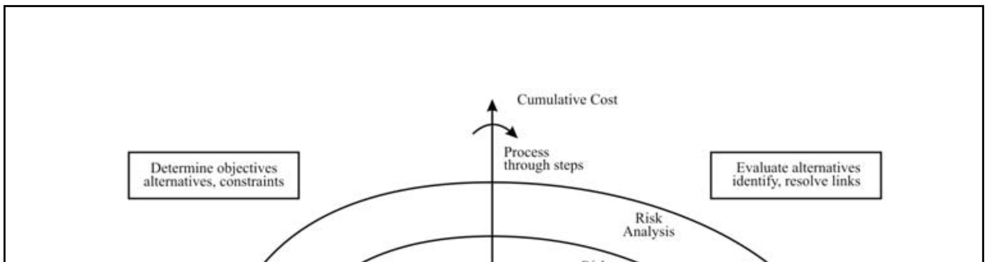

Cumulative Cost
Process through steps
Determine objectives alternatives, constraints
Evaluate alternatives identify, resolve links
Risk Analysis

Figure 1.6: Spiral model

Figure 2: Boehm's spiral model (IEEE, 1988)

In this model, Software development starts with lesser requirements specification, lesser risk analysis, etc. The radical dimension this model represents cumulative cost. The angular dimension represents progress made in completing the cycle.

The inner cycles of the spiral model represent early phases of requirements analysis and after prototyping of software, the requirements are refined.

In the spiral model, after each phase a review is performed regarding all products developed upto that point and plans are devised for the next cycle. This model is a realistic approach to the development of large scale software. It suggests a systematic approach according to classical life cycle, but incorporates it into iterative framework. It gives a direct consideration to technical risks. Thus, for high risk projects, this model is very useful. The risk analysis and validation steps eliminate errors in early phases of development.

3.2.3 Selection of Approach

"Which development process is best" or "Which development process is best for my project?" In search of these questions we can debate the advantage and disadvantages of one development process over another until we get fed up, but we will never reach the conclusion,Why? Because different processes have different strengths and weaknesses.

Each development process is viable and useful, but each one works better in some situations than it does in others.

If you're working on a somewhat traditional project for example, a school management system, where the features are defined a priori and it's possible to freeze the specification, the waterfall process is best in that case. If you're working on a project whose nature and scope are vague or unpredictable (for example, a project where you cannot determine your next steps until you have reached a certain plateau), consider either a spiral or feedback waterfall process. These two processes are very similar in that both involve a series of many frequent iterations of the design-implementation-integration-test cycle seen in the waterfall process.

3.3 ANALYSIS

Analysis describes the "What" of a system. The objectives that are to be achieved in Software process development, are the requirements. In the requirements analysis phase, the requirements are properly defined and noted down. The output of this phase is SRS (Software Requirements Specification) document written in the natural language. According to IEEE, requirements analysis may be defined as (1) the process of studying user's needs to arrive at a definition of system hardware and software requirements (2) the process of studying and refining system hardware or software requirements.

The main activities in the analysis phase are cost estimation, project planning and scheduling, feasibility report preparation, and the development of SRS (software requirement specifications) document of the project.

Feasibility Report

You may include the following contents in the feasibility report given in the *Table* shown below:

Topic	Some of the contents of the topic
Introduction	Include the project background and report layout here.
Terms of Reference	Define the expectations of the project, the Time Frame and Available Resources
Existing System	Define here the results of Fact Finding, working of the Current System and the problems in the Current System
System Requirements	Give the system requirements here, after discussion with the System User
Proposed System	Define the outline of the Proposed System, key Input and Output to and from the system
Development Plan, Cost Feasibility and other feasibilities	You can include the Gantt Chart, Pert Chart, cost estimation, etc.

Cost Estimation

Cost estimates are made to discover the cost to the developer, the cost of producing a software system. It is an important activity in the real time project management, however, this course may not require you to calculate cost estimation but it is important for you to understand different methods of cost estimation for future use. There are different cost estimation methods available like LOC, COCOMO, Putnams, Statistical method and functional point methods, you can refer your MCS-034 course material for further details.

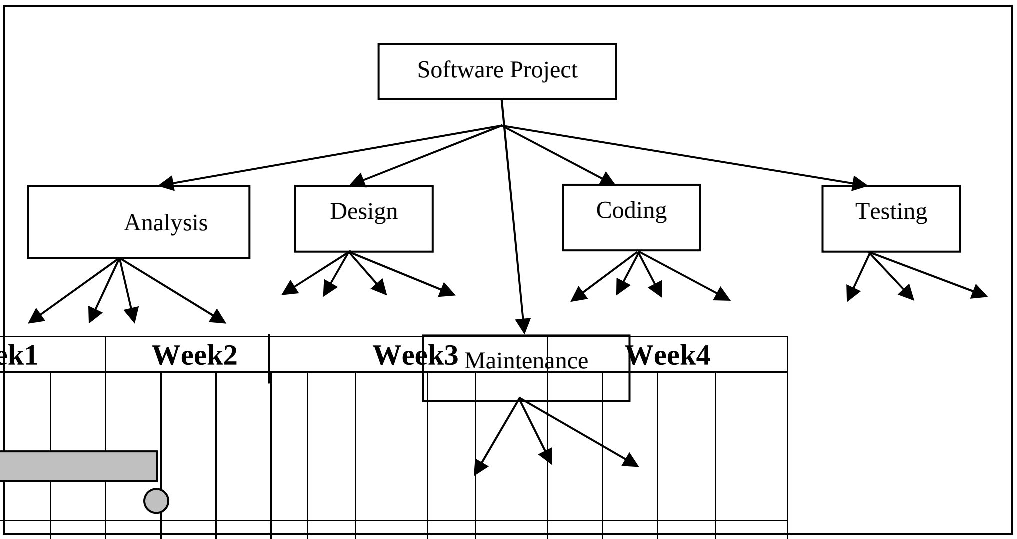
Project Planning and Scheduling

Scheduling of a software project can be correlated to prioritising various tasks (jobs) with respect to their cost, time and duration. Scheduling can be done with resource constraint or time constraint in mind. Depending upon the project, scheduling methods can be static or dynamic in implementation.

Scheduling Techniques

The following are the various types of scheduling techniques in software engineering:

Work Breakdown Structure: The project is scheduled in various phases following a bottom-up or top-down approach. A tree-like structure is followed without any loops. At each phase or step, milestone and deliverables are mentioned with respect to requirements. The work breakdown structure shows the overall breakup flow of the project and does not indicate any parallel flow. *Figure 3* depicts an example of a work breakdown structure.



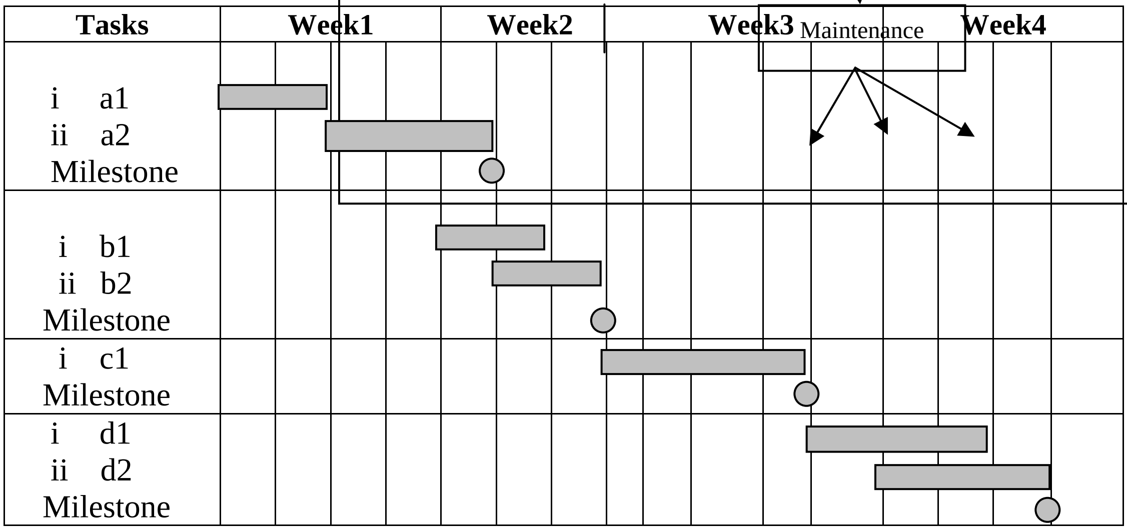


Figure 3: An example WBS

The project is split into requirement and analysis, design, coding, testing and maintenance phase. Further, requirement and analysis is divided into R1,R2, .. Rn; design is divided into D1,D2..Dn; coding is divided into C1,C2..Cn; testing is divided into T1,T2.. Tn; and maintenance is divided into M1, M2.. Mn. If the project is complex, then further sub-division is done. Upon the completion of each stage, integration is done.

Flow Graph: Various modules are represented as nodes with edges connecting nodes. Dependency between nodes is shown by flow of data between nodes. Nodes indicate milestones and deliverables with the corresponding module implemented. Cycles are not allowed in the graph. Start and end nodes indicate the source and terminating nodes of the flow. *Figure 4* depicts a flow graph.

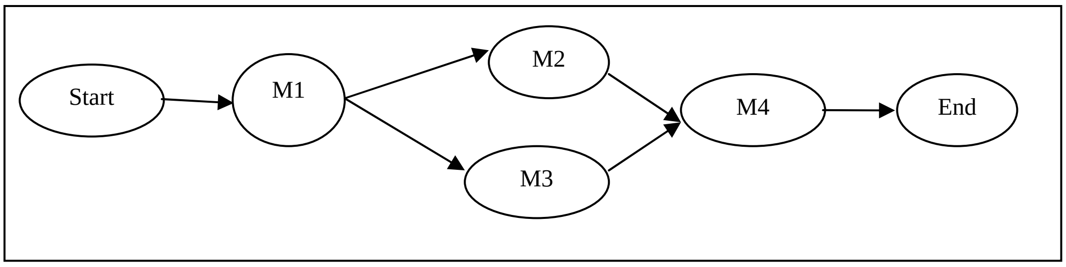


Figure 4: Flow graph

M1 is the starting module and the data flows to M2 and M3. The combined data from M2 and M3 flow to M4 and finally the project terminates. In certain projects, time schedule is also associated with each module. The arrows indicate the flow of information between modules.

Gantt Chart or Time Line Charts: A Gantt chart can be developed for the entire project or a separate chart can be developed for each function. A tabular form is maintained where rows indicate the tasks with milestones and columns indicate duration (weeks/months) . The horizontal bars that spans across columns indicate duration of the task. *Figure 5* depicts a Gantt Chart. The circles indicate the milestones.

Program Evaluation Review Technique (PERT): Mainly used for high-risk projects with various estimation parameters. For each module in a project, duration is estimated as follows:

Figure 5: Gantt chart

1) Time taken to complete a project or module under normal conditions, *tnormal.*

2) Time taken to complete a project or module with minimum time (all resources available), *tmin.*

3) Time taken to complete a project or module with maximum time (resource constraints), *tmax.*

4) Time taken to complete a project from previous related history, *Thistory*.

An average of tnormal, tmin, tmax and thistory is taken depending upon the project. Sometimes, various weights are added as 4*tnormal, 5*tmin, 0.9*tmax and 2*thistory to estimate the time for a project or module. The Project manager fixes the parameters.

The software project management tools support project planning and scheduling. Some of the sample information that you can produce using these tools are:

A Sample Partial Project Plan

Overall Goal of the Project

The proposed software system should be able to:

- *read the structured data available in the Excel files and store it in a Database system,*
- *validate data of the database on the basis of predefined business rules,*
- *prepare reports on data that is found to be in error during validation, and*
- *prepare MIS reports from the validated data.*

The Primary Data

The Primary data in the system is the employee data of the participant companies.

Delivery Deadlines

6 months from the date of Approval.

PROJECT PLAN

1. OBJECTIVES

The objective of the system can be defined here as:

- *The proposed system should be able to read the data from Excel files and store validated data in the Database.*
- *………………………………………*

2. SPECIFIC PRODUCTS TO BE DELIVERED

The products that will be delivered (You need not include all for an actual system):

The tested system and Network

Clint Workstations

A robust Database Management Server

Any other third party software.

3. ACTIVITIES AND MILESTONES

The activities in the system, after including the provisions for security, are:

- *Verification of the Users.*
- *Migration of the Excel data.*
- *Validation of the migrated data.*
- *……………………………….*

The milestones in the system are:

• *Start of the Project*	:	*1st June, 2006*
• *SRS Completion*	:	*28th June, 2006*
• *Requirements finalisation*	:	*1st July, 2006*
• *…………………………………………………*		

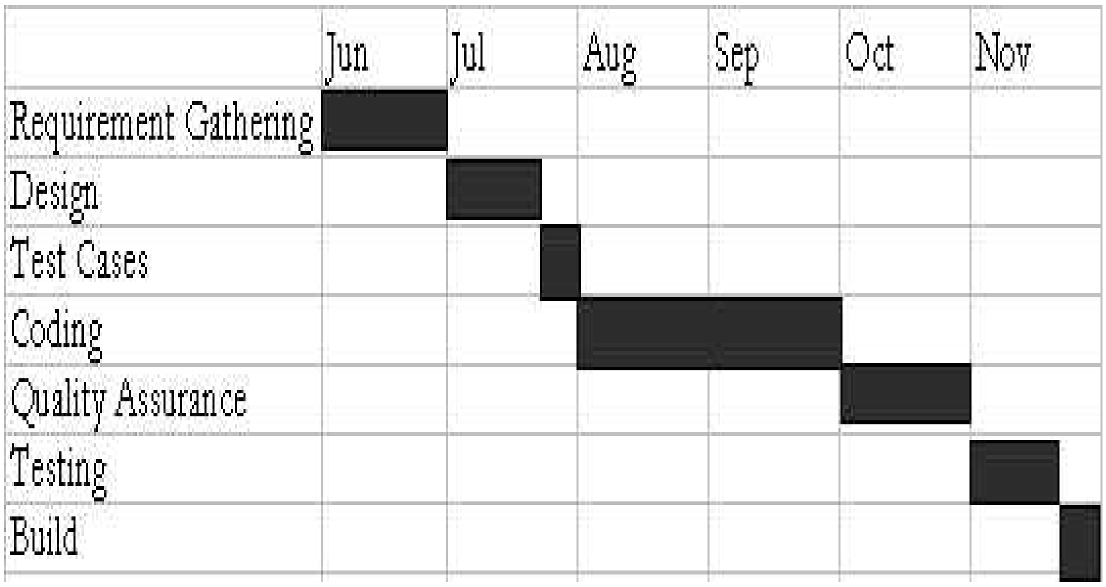
4. RESOURCE REQUIREMENT

The Hardware, Software, and Tools are required for the three different environments, viz., Development environment, Quality Control environment, Deployment environment. These resources may be documented accordingly.

5. SCHEDULING

GANTT Chart

A very elementary Gantt or Timeline Chart for the development plan is given below. The plan explains the tasks versus the time they will take to complete.



PERT Chart

A suitable PERT Chart for the development plan can be given here.

6. BUDGETARY ESTIMATES

The whole budgetary estimates may be divided into resource procurement and software development costs. The software development efforts and cost may be estimated with the help of COCOMO or any other model used by the organisation.

Software Requirement Specification

This document is generated as output of requirement analysis. The requirement analysis involves obtaining a clear and thorough understanding of the product to be developed. Thus, SRS should be a consistent, correct, unambiguous and complete, document. The developer of the system can prepare SRS after detailed communication with the customer. An SRS clearly defines the following:

- **External Interfaces of the system:** They identify the information which is to flow '*from and to'* to the system.
- Functional and non-functional requirements of the system. They stand for the finding of run time requirements.
- Design constraints:

Normally IEEE standard is followed. A typical format of SRS is as follows:

TABLE OF CONTENTS

Introduction

- Purpose
- Scope
- Definition
- Product and its function
- Benefits and Goals
- Overview.

Overall Description

- Product Description
- Product Functioning
- Functions of Project
- Users of Project
- Assumptions made.

Specific Requirements

- Interface Requirements
- User Requirements
- Hardware Requirements
- Software Requirements
- Logical Database Requirements.

Basic Processing Actions of the System
Appendices

- Input/Output Formats
- Instruction for Security
- Results of Fact Finding
- Data Model
- Functional Model
- Process Specification.

A sample portion of SRS is given below:

INTRODUCTION

Purpose

SRS contains details of the proposed software system, sufficient enough for the designers to design the system. Thus, SRS is a means of communicating the findings of the analysis stage to the design stage. The SRS includes:

5 Interface,

6 Logical Database,

7 Hardware, and

8 Performance and other constraints.

It also contains the assumptions made by the analyst and any systemic dependency.

Scope

The scope of SRS contains all the areas related to the project. The scope of SRS includes:

- Proposed software description,
- Users of the proposed software,
- Functional requirements of the software,
- Assumptions and dependencies in the system, and
- Constraints.

Definition

……………………..

Product and its function

………………………..

Benefits and Goals

………………………..

Overview

..………………………

Overall Description

Product Description

The Client should be able to upload the raw data in Excel files into the Database. The raw data is then validated using ………

Product Functioning

- The Raw data from the Clients is put into the database.

………………………………………………..

Functions in the Project

There are five functions of the system.

- User Verification

The User is asked to enter the Username and Password. The system checks the validity of Username and Password, otherwise the User is asked to do so again. A maximum of three valid attempts are given to the user.

- Upload Raw Data

……………………….

- Validate Data

……………………….

- Put the Validated Data

……………………….

- Generating Reports

…………………………..

Users of the Product

………………………

Assumptions

……………………

SPECIFIC REQUIREMENTS

Interface Requirements

The Interface requirements include:

- Easy to follow Interface,
- Very few graphics,
- No hidden buttons, and
- Relevant error messages.

……………………………….

User Requirements

After a careful study of the requirements of the Client, analysts have come up with a set of requirements.

…………………………………..

Hardware and Software Requirements

There are three environments that have been created for the project, viz.,

- Development environment,
- Quality Control environment, and
- Production environment.

The hardware requirements for all the platforms may be indicated here.

Logical Database Requirements

The following information is to be stored in the database.

- The Clients Raw data,
- The Clients Validated data, and
- Username and Password.

BASIC PROCESSING ACTIONS OF THE SYSTEM

The basic processing actions of the system are.

- Verification of the User

………………………………..

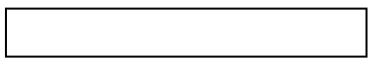
- Upload Data

APPENDICES

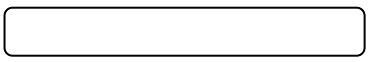
APPENDIX A

INPUT / OUTPUT FORMATS

The input formats for the system contains the following screens.
The convention used while making the input formats are.



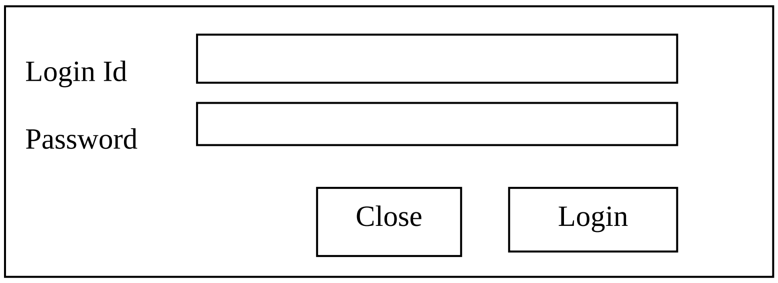
Square box is used for user input



Rounded Square box is used for system display

Login Screen

The following screen that inputs the Username and Password from the User for authentication of the User to the system is:



APPENDIX B

Instructions For Security

Security is an integral part of any system. Clients send their confidential data with trust and it is our foremost duty to protect the security and privacy of the data of the Clients.

……………………….

APPENDIX C

RESULTS OF FACT FINDING

Working of the Current System and its Problems

APPENDIX D

DATA MODEL

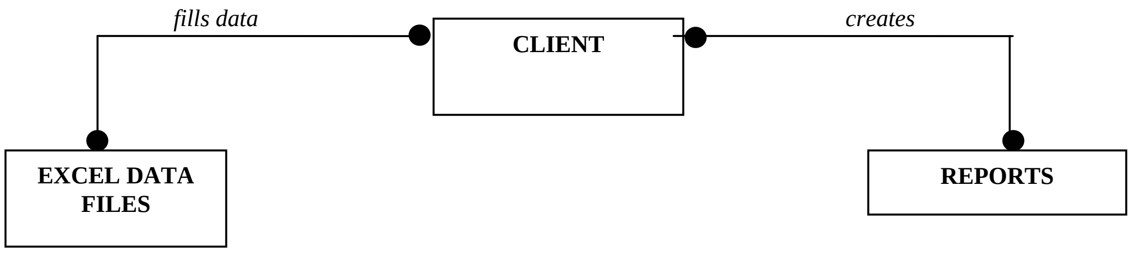
Classes involved in the Project

- Clients
- Excel Data Files
- Reports

Association between the classes

- Clients fill data in the Excel Data Files (M .. M)
- Clients generate Reports (M … M)

E-R/Object Diagram for the System



Attributes of the Entities are:

Object Classes	Attribute
Clients	number name address

	phone number fax email
Excel data Files	excel file number client number
Reports	report number report name client number

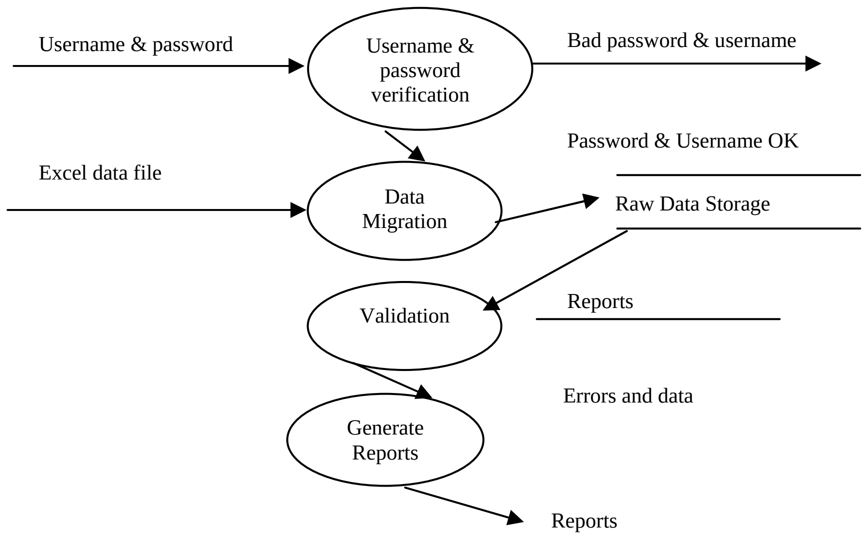
APPENDIX E

FUNCTIONAL MODEL

One sample DFD at level 1 contains the following processes:

- Verification of username and Password
- Data Migration,
- Validation of Migrated data, and
- Generating Reports.

You can make various levels of DFDs



Process Specification

Let us give one sample process verification.

Process: Username and Password Verification

Input: Username & Password

Output: Access granted or denied message

Processing

The Client enters the Username and Password and clicks the Login button.

The system connects with the DBMS and verifies them in the related database. If both are valid: allow user to enter the system with the allowed access rights for that user. Otherwise prompt wrong Username-Password message and take the user to the screen where s/he can re-enter the Username-Password.

Interface Description

The interface contains the text boxes where the user can enter Username and Password. It also contains a Login button for login and a Close button on closing the application.

Internal Data Structure

The process uses the encrypted username and password table that also contains information on access permission.

3.4 DESIGN

Software design is all about developing the blue print for designing workable software. The goal of a software designer is to develop models that can be translated into software. Unlike design in civil and mechanical engineering, software design is a new and evolving discipline contrary to classical building design etc. In early days, software development mostly concentrated on writing codes. Software design is central to the software engineering process. Various design models are developed during the design phase. The design models are further refined to develop detailed design models, which are closely related to the program.
Software Design Process

Software design is the process of applying various software engineering techniques for developing models in order to define a software system, which provides sufficient details for the actual realisation of the software. The goal of software design is to translate user requirements into an implementable program.

Software design is the only way through which we can translate user requirements to workable software. In contrary to designing, a building software design is not a fully developed and mature process. Nevertheless the techniques available provides us with tools for a systematic approach to the design of software. *Figure 6* depicts the process of software design.

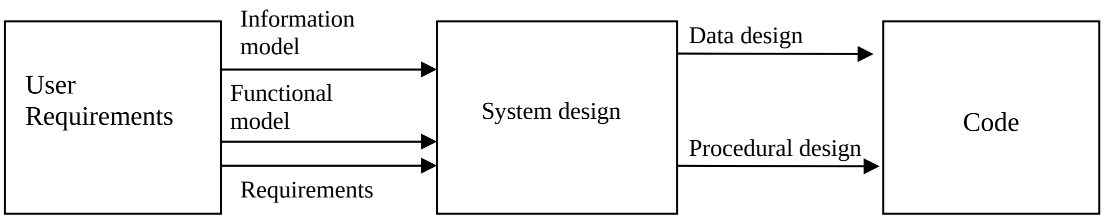


Figure 6: Process of software Design

During the process of software design, the information model is translated in to data design. Functional model and behavioural model are translated to architectural design, which defines major component of the software. Keeping in view the importance of design, it should be given due weightage before rushing to the coding of software.

Software design forms the foundation for implementation of a software system and helps in the maintenance of software in the future too. Software quality and software design process are highly interrelated. Quality is built into the software during the design phase.

High level design gives a holistic view of the software to be built, whereas low level refinements of the design are very closely related to the final source code. A good design can make the work of programmer easy and hardly allows the programmer to forget required details. Sufficient time should be devoted to the design process to ensure good quality software.

The following are some of the fundamentals of design:

9 The design should follow a hierarchical organisation,

10 Design should be modular or logically partitioned into modules, which are relatively independent and perform independent task,

11 Design leading to interface that facilitates interaction with the external environment,

12 Step-wise refinement to more detailed design, which provides necessary details for the developer of the code, and

13 Modularity is encouraged to facilitate parallel development, but, at the same time, too many modules lead to the increase of effort involved in integrating the modules.

System Design Specification

The system design specification or software design specification as referred to, has a primary audience, the system implementer or coder. It is also an important source of information the system verification and testing. The system design specification gives a complete understanding of the details of each component of the system, and its associated algorithms, etc.

The system design specification documents the requirements of the system be implemented. It consists of the final steps of describing the system in detail before coding begins.

The system design specification is developed in a two stage process. In the first step, design specification generally describes the overall architecture of the system at a higher level. The second step provides the technical details of low-level design, which will guide the implementer. It describes exactly what the software must perform to meet the requirements of the system.

Tools for Describing Design

Various tools are used to describe the higher level and lower level aspects of system design. The following are some of the tools that can be used in the System Design Specification to describe various aspects of the design.

Data Dictionary

Definition of all the data and control elements used in the software product or sub-system. Complete definition of each data item and its synonyms are included in the data dictionary. A data dictionary may consist of description of data elements and definitions of tables.

Description of Data Element:

14 Name and aliases of data item (its formal name).

15 Uses (which processes or modules use the data item; how and when the data item is used).

16 Format (standard format for representing the data item).

17 Additional information such as default values, initial value(s), limitations and constraints that are associated with the data elements.

Table Definitions

- Table name and Aliases.
- Table owner or database name.

- Key order for all the tables, possible keys including primary key and foreign key.
- Information about indexes that exist on the table.

Database Schema: Database schema is a graphical presentation of the whole database. Data retrieval is made possible by connecting various tables through keys. Schema can be viewed as a logical unit from the programmer's point of view.

E-R Model: Entity-relationship model is database analysis and design tool. It lists real-life application entities and defines the relationship between real life entities that are to be mapped to the database. E-R model forms the basis for database design.

Security Model: The database security model associates users, groups of users or applications with database access rights.

Trade-off Matrix

A matrix that is used to describe decision criteria and relative importance of each decision criterion. This allows comparison of each alternative in a quantifiable term.

Decision Table

A decision table shows the way the system handles input conditions and subsequent actions on the event. A decision table is composed of rows and columns, separated into four separate quadrants.

Input Conditions	Condition Alternatives
Actions	Subsequent Action Entries

Timing Diagram

Describes the timing relationships among various functions and behaviours of each component. They are used to explore the behaviours of one or more objects throughout a given period of time. This diagram is specifically useful to describe logic circuits.

State Machine Diagram

State machine diagrams are good at exploring the detailed transitions between states as the result of events. A state machine diagram is shown in *Figure 7.*

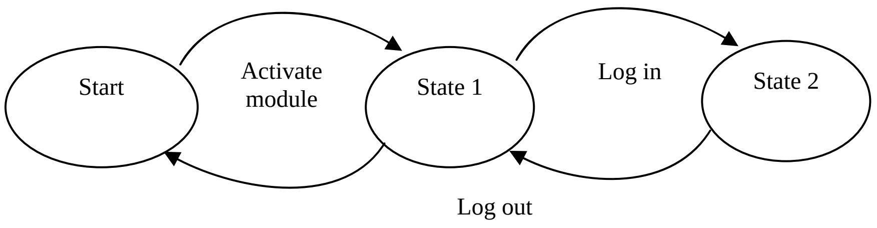
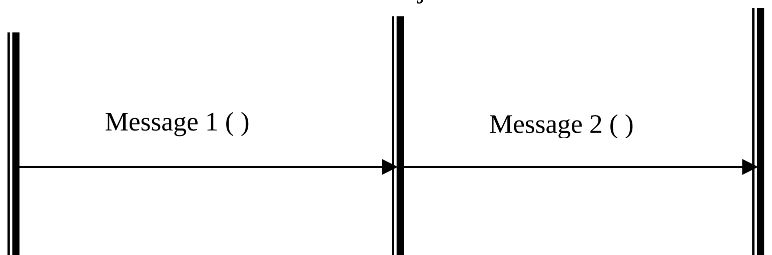


Figure 7: A state machine diagram

Object Interaction Diagram

It illustrates the interaction between various objects of an object-oriented system. Please refer to *Figure 8.* This diagram consists of directed lines between clients and servers. Each box contains the name of the object. This diagram also shows conditions for messages to be sent to other objects. The vertical distance is used to show the life of an object.



Object: class1
Object:class2
Object:class2

Figure 8: An object interaction diagram

A Flow chart shows the flow of processing control as the program executes. Please refer to *Figure 9.*

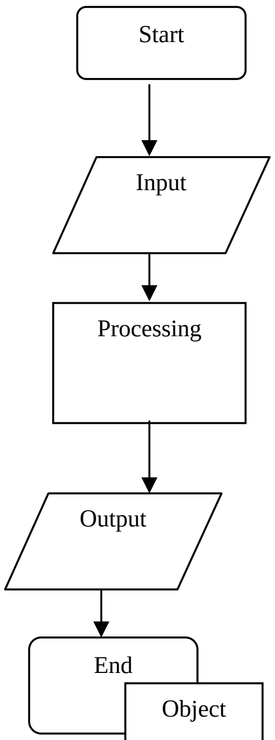
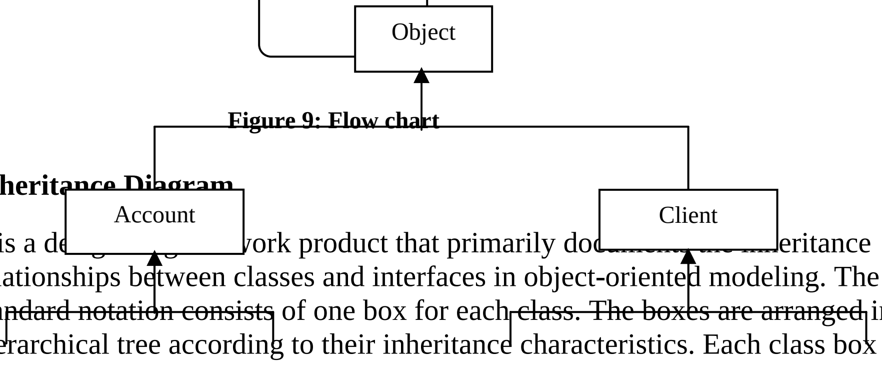


Figure 9: Flow chart

Inheritance Diagram



It is a de[illegible]vork product that primarily doc[illegible]eritance relationships between classes and interfaces in object-oriented modeling. The standard notation consists of one box for each class. The boxes are arranged in a hierarchical tree according to their inheritance characteristics. Each class box includes the class name, its attributes, and its operations. *Figure 10* shows a typical inheritance diagram.

Figure 10: A typical inheritance diagram

Aggregation Diagram

The E-R model cannot express relationships among relationships. An aggregation diagram shows relationships among objects. When a class is formed as a collection of other classes, it is called an aggregation relationship between these classes. Each module will be represented by its name. The relationship will be indicated by a directed line from container to container. The directed line is labelled with the cardinality of the relationship. It describes "has a" relationship. *Figure 11* shows an aggregation between two classes (circle *has a* shape).

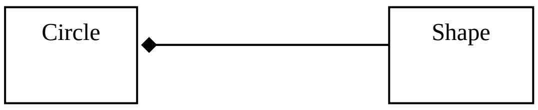


Figure 11: Aggregation

Structure Chart

A structure chart is a tree of sub-routines in a program (Refer to *Figure 12*). It indicates the interconnections between the sub-routines. The sub-routines should be labelled with the same name used in the pseudo code.

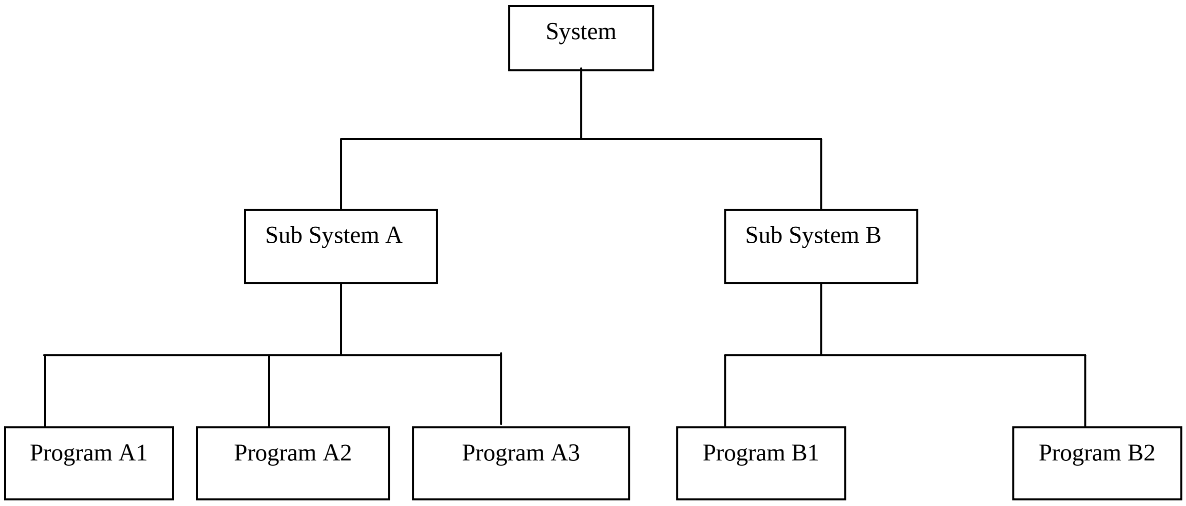
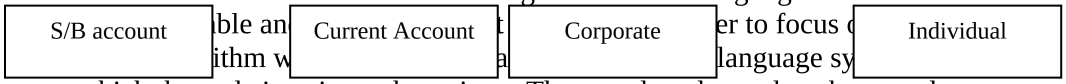


Figure 12: A structures chart

Pseudocode

Pseudocode is a kind of structured English for describing algorithms in an [illegible]ble an[illegible] t [illegible] er to focus [illegible] [illegible]ithm w[illegible] a[illegible] language sy[illegible] which the code is going to be written. The pseudocode needs to be complete. It describes the entire logic of the algorithm so that implementation becomes a routine mechanical task of translating line by line into the source code of the target language. Thus, it must include flow of control.



This helps in describing how the system will solve the given problem. "Pseudocode" does not refer to a precise form of expression. It refers to the simple use of Standard English. Pseudocode must use a restricted subset of English in such a way that it resembles a good high level programming language. *Figure 13* shows an example of pseudo code.

```
IF HoursWorked > MaxWorkHour THEN
Display overtime message
ELSE
```

Figure 13: Example of a pseudocode

Contents of a Typical System Design Specification Document

1. Introduction

   1.1 Purpose and scope of this document:
   Full description of the main objectives and scope of the SDS document is specified.

   1.2 Definitions, acronyms, abbreviations and references
   Definitions and abbreviations used are narrated in alphabetic order. This section will include technical books and documents related to design issues. It must refer to the SRS as one of the reference book.

2. System architecture description

   2.1 Overview of modules, components of the system and sub-systems
   Structure and relationships. Interrelationships and dependencies among various components are described. Here, the use of structure charts can
be
   useful.

3. Detailed description of components

17.2.1.1.1.1 3.1 Name of the component
17.2.1.1.1.2 3.2 Purpose and function
17.2.1.1.1.3 3.3 Sub-routine and constituents of the component
17.2.1.1.1.4 3.4 Dependencies, processing logic including pseudocode
17.2.1.1.1.5 3.5 Data elements used in the component.

4. Appendices.

3.5 SAMPLE DESIGN DOCUMENT

SCOPE

Define the System's Objectives here

……………………

Architecture Design

Give the architecture of the system based on the analysis and flow models.

DATA DESIGN

Refine the E-R diagram if needed and make one table for each entry and data store and table for all M.M relationships.

For example, the Client table may be:

COLUMN HEADING	*CONSTRAINTS*

client_no	Primary Key
client_number	Not Null
client_addr	Not Null

…….

INTERFACE DESIGN

Human-Machine Interface Design Rules

Follow the basis rules for the interface design. These can be classified into three main types:

- External Interface design
- Interface to the External Data
- Interface to the external system or devices

External Interface Design

18 Easy to follow Interface

19 Zero or very less graphics (as not being used commercially)

20 No hidden buttons

21 Proper error messages

…………………………………….

Interface to the External Data

The system has to use proper external data. The system makes a connection with the DBMS to do so. The rules that have to be followed while interfacing with the external data are:

- You must do the type checking.
- Field overrun that is maximum size should be enough to accommodate the largest data of that type
- It is always better to encrypt the data and then store it in the database.

Interface to the External System or Device

…………………………………..

PROCEDURAL DESIGN

Verification of the User

Algorithm

Input: ………..

Output: …………….

STEPS

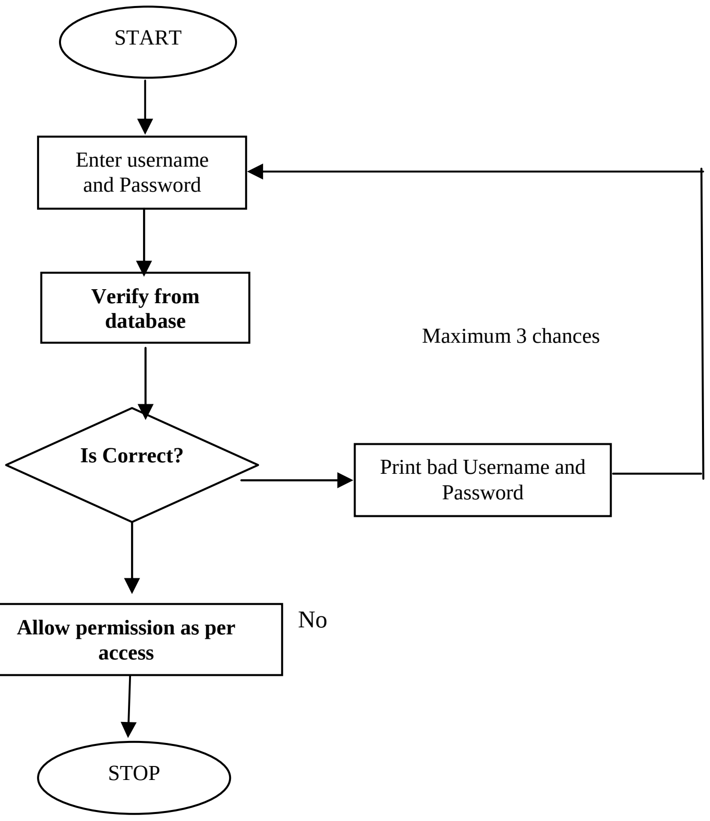
Ask the Client to enter Username-Password.

Check Username and Password combination in the encrypted database.

If the Username and Password is correct then give permission with allowed access rights.

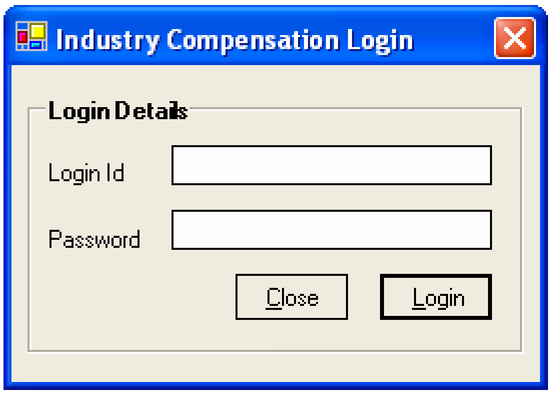
Else, access is denied and the enter Username and Password screen is shown again. This cycle may be repeated for a maximum of three times.

Flow Chart


START
Enter username and Password
Verify from database
Maximum 3 chances
Is Correct?
Print bad Username and Password
Allow permission as per access
No
STOP

Interface Design

Login Screen



Output / Reports Design

Design your output reports

Data Security and Rights

Security is the integral part of any system. You can use encryption and authentication or any other security mechanism as the need may be.

3.6 CODING

The input to the coding phase is the SDD document. In this phase, the design document is coded according to the module specification. This phase transforms the SDD document into a high-level language code. Currently, major software companies adhere to some well-specified and standard style of coding called coding standards. Good coding standards improve the understanding of the code. Once a module is developed, a check is carried out to ensure that coding standards are followed. Coding standards generally give guidelines on the following:

i) Name of the module
ii) Internal and External documentation of source code
iii) Modification history
iv) Uniform appearance of codes.

It must contain proper comments. The coding should be structured as per software architecture.

VALIDATION CHECKS

Verification of the User

Both User Id and Password are mandatory
The size of User Id must be 10 characters long

ERROR AND EXCEPTION HANDLING

A proper mechanism of Error Handling is necessary in any system.

COMMENTS

Comments are an integral part of any system. Like any properly developed and maintained system with high quality, this system too has sufficient comments and description of the processes, functions, classes, and data structures.

CODING STANDARDS

The defined coding standards of software developers may be followed. A sample clause in coding standard may be: “A consistent naming pattern is one of the most important elements of predictability and discoverability in a managed code. For each type, you must follow the defined capitalisation styles, case sensitivity and word choice”.

3.7 TESTING

Testing

Testing is the process of running the software on manually created inputs with the intention to detect errors. In the process of testing, an attempt is made to detect errors, to correct the errors in order to develop error free software. Testing is performed keeping the user’s requirements in mind and before the software is actually launched on a real system, it is tested.

Normally, while developing the code, a software developer also carries out some testing. This is known as debugging. This unearths the defects that must be removed from the program. Testing and debugging are different processes. Testing is meant for finding the existence of defects while debugging stands for locating the place of errors and correcting the errors during the process of testing. The following are some guidelines for testing:

i) Test the modules thoroughly, cover all the access paths, generate enough data to cover all the access paths arising from conditions.
ii) Test the modules by deliberately processing/entering wrong data.
iii) Specifically create data for conditional statements. Enter data in test file, which will satisfy the conditions and test the script again.
iv) Test for locking by invoking multiple concurrent processes.

The following objectives are to be kept in mind while performing testing:

i) It should be done with the intention of finding errors.
ii) Good test cases should be designed with a probability of detecting undiscovered errors.
iii) A successful test is one that uncovers yet undiscovered error(s).

The following are some of the principles of testing:

i) All tests should be performed according to user requirements.
ii) Planning of tests should be done long before testing.
iii) Starting with a small test, it should proceed towards large tests.

The following are different levels of testing:

Large systems are built out of subsystems, subsystems are made up of modules, of procedures and functions. Thus, in large systems, testing is performed at various levels, like unit level testing, module level testing, subsystem level, and system level testing.

In all levels, testing is performed to check interface integrity, information content, and performance.

The following are some of the strategies for testing: This involves design of test cases. Test case is a set of designed data for which the system is tested. Two testing strategies are present.

i) **Code Testing:** The code testing strategy examines the logic of the system. In this, the analyst develops test cases for every instruction in the code. All the paths in the program are tested. This test does not guarantee against software failures. Also, it does not indicate whether the code is according to requirements or not.

ii) **Specification Testing:** In this, testing with specific cases is performed. The test cases are developed for each condition or combination of conditions and submitted for processing.

The objective of testing is to design test cases that systematically uncover different classes of errors and do so with the minimum amount of time and effort. Testing cannot show the absence of errors. It can only find the presence of errors. The test case design is as challenging as software development. Still, however effective the design is, it cannot remove 100% errors. Even, the best quality software are not 100 % error free. The reliability of software is closely dependent on testing.

Some testing techniques are the black box and the white box methods.

White Box Testing: This method, also known as glass box testing, is performed early in the testing process. Using this, the software engineer can derive tests that guarantees exercises of all independent paths within the module at least once. It has the following features:

i) Exercises all logical decisions on their true and false sides.

ii) Executes all loops at their boundaries and within their operational bounds.
iii) Exercises internal data structures to assure their validity.

Black Box Testing: This is applied during the later stage of testing. It enables the software developer to derive a set of input conditions that will fully exercise the functional requirements of a program. It enables him/her to detect errors like incorrect or missing functions, interface errors, data structures or external data base access errors and performance errors etc.

3.8 TEST DESIGN DOCUMENT

During system development, this document provides the information needed for adequate testing. It also lists approaches, procedures and standards to ensure that a quality product meets the requirement of the user. This document is generally supplemented by documents like schedules, assignments and results. A record of the final result of the testing should be kept externally.

This document provides valuable input for the maintenance phase.

The following IEEE standards describe the standard practices on software test and documentation:

829-1998 IEEE Standard for Software Test Documentation
1008-1987 (R1993) IEEE Standard for Software Unit Testing
1012-1998 IEEE Standard for Software Verification and Validation

The following is the typical content of the Test Design Document:

1. Introduction

Purpose

The purpose of this *document* and its intended audience are clearly stated.

Scope

Give an overview of testing process and major phases of the testing process. Specify what is not covered in the scope of the testing such as, supporting or not third party software.

Glossary

It defines technical terms used in this document.

References

Any references to other external documents stated in this document including references to related project documents. They usually refer to the System Requirement Specification and the System Design Specification documents.

Overview of Document

Describe the contents and organisation of the document.

Test Plan

A test plan is a document that describes the scope, approach, resources and schedule of intended testing activities. It identifies test items, the features to be tested, the testing tasks, and the person who will do each task, and any risks that require contingency planning.

Schedules and Resources

An overview of the testing schedule in phases along with resources required for testing is specified.

Recording of Tests

Specify the format to be used to record test results. It should very specifically name the item to be tested, the person who did the testing, reference of the test process/data and the results expected by that test, the data tested. If a test fails, the person responsible for correcting and re-testing is also documented. The filled out format would be kept with the specific testing schedule. A database could be used to keep track of testing.

Reporting Test Results

The summary of what has been tested successfully and the errors that still exist which are to be rectified is specified.

Verification Testing

Unit Testing

For each unit/component, there must be a test, which will enable the tester to know the accurate functioning of that unit.

Integration testing

Integration test is done on modules or sub-systems.

Validation Testing

System Testing

This is top level integration testing. At this level, requirements are validated as described in the SRS.

Acceptance and Beta Testing

List test plans for acceptance testing or beta testing. During this test, real data is used for testing by the development team (acceptance testing/alpha testing) or the customer (beta testing). It describes how the results of such testing will be reported and handled by the developers.

A sample test case may be:

Login Screen

S.No.	Test Case ID	Do	Expected Result
1.	QT1-001	Enter user id in the text box specified.	Successful login in to the system if the values

		User id must not be more than 510 characters and it should not contain any special characters and no spaces including in the start. Enter password in the text box specified. Password must not be more than 10 characters. Click on the Login button.	are found in the database.
…..			

Test Log

Function	**Purpose of Set of Test Cases per Area**	**Test Case ID(s)**	**No of Test Cases run**	**Number of Test Cases Successful**
Login	The verification of username and password	QT1-001	8	100%
……				

3.9 SUMMARY

Software engineering covers the entire range of activities used to develop software. The activities include requirements analysis, program development using some recognised approach like structured programming, testing techniques, quality assurance, management and implementation and maintenance. Further, software engineering expects to address problems, which are encountered during software development.

3.10 FURTHER READINGS

1. *Software Engineering, Sixth Edition, 2001*, Ian Sommerville; Pearson Education., New Delhi.

2. *Software Engineering – A Practitioner's Approach*, Roger S. Pressman; McGraw-Hill, New Delhi.

3. Jeffrey A. Hoffer, Joey F. George, Joseph S. Valacich; *Modern Systems Analysis and Design*; Pearson Education; Third Edition; 2002.

4. ISO/IEC: 18019: Guide lines for the design and preparation of user documentation for application software.

5. MCS-034 Course material of MCA (revised syllabus), IGNOU.

6. MCS-014 Course material of MCA (revised syllabus), IGNOU.

7. http://www.rspa.com

8. http://www.ieee.org

9. http://standards.ieee.org

10. http://www.sce.carleton.ca/squall

11. http://en.tldp.org/HOWTO/Software-Release-Practice-HOWTO/documentation.html

12. http://www.sei.cmu.edu/cmm/