BAOU
Education for All

DR.BABASAHEB AMBEDKAR OPEN UNIVERSITY

BCA

BCAR-402

Software Engineering

SOFTWARE ENGINEERING

DR. BABASAHEB AMBEDKAR OPEN UNIVERSITY
AHMEDABAD

Editorial Panel

Author : Mr. Parimal Patel
I/C Director,
Khyati School of Computer Application,
Ahmedabad.

Editor : Dr. Tulsidas. V. Nakrani
Assistant Professor,
Sankalchand Patel College of Engineering,
MCA Department, Visnagar.

Language Editor : Dr. Jagdish Vinayakrao Anerao
Associate Professor,
Smt A. P. Patel Arts and
N. P. Patel Commerce College,
Ahmedabad.

ISBN 978-93-91071-27-1

Edition : 2021

Acknowledgment

Every attempt has been made to trace the copyright holders of material reproduced in this book. Should an infringement have occurred, we apologize for the same and will be pleased to make necessary correction/amendment in future edition of this book.

ROLE OF SELF–INSTRUCTIONAL MATERIAL IN DISTANCE LEARNING

The need to plan effective instruction is imperative for a successful distance teaching repertoire. This is due to the fact that the instructional designer, the tutor, the author (s) and the student are often separated by distance and may never meet in person. This is an increasingly common scenario in distance education instruction. As much as possible, teaching by distance should stimulate the student's intellectual involvement and contain all the necessary learning instructional activities that are capable of guiding the student through the course objectives. Therefore, the course / self-instructional material is completely equipped with everything that the syllabus prescribes.

To ensure effective instruction, a number of instructional design ideas are used and these help students to acquire knowledge, intellectual skills, motor skills and necessary attitudinal changes. In this respect, students' assessment and course evaluation are incorporated in the text.

The nature of instructional activities used in distance education self-instructional materials depends on the domain of learning that they reinforce in the text, that is, the cognitive, psychomotor and affective. These are further interpreted in the acquisition of knowledge, intellectual skills and motor skills. Students may be encouraged to gain, apply and communicate (orally or in writing) the knowledge acquired. Intellectual-skills objectives may be met by designing instructions that make use of students' prior knowledge and experiences in the discourse as the foundation on which newly acquired knowledge is built.

The provision of exercises in the form of assignments, projects and tutorial feedback is necessary. Instructional activities that teach motor skills need to be graphically demonstrated and the correct practices provided during tutorials. Instructional activities for inculcating change in attitude and behaviour should create interest and demonstrate need and benefits gained by adopting the required change. Information on the adoption and procedures for practice of new attitudes may then be introduced.

Teaching and learning at a distance eliminate interactive communication cues, such as pauses, intonation and gestures, associated with the face-to-face method of teaching. This is

particularly so with the exclusive use of print media. Instructional activities built into the instructional repertoire provide this missing interaction between the student and the teacher. Therefore, the use of instructional activities to affect better distance teaching is not optional, but mandatory.

Our team of successful writers and authors has tried to reduce this.

Divide and to bring this Self-Instructional Material as the best teaching and communication tool. Instructional activities are varied in order to assess the different facets of the domains of learning.

Distance education teaching repertoire involves extensive use of self-instructional materials, be they print or otherwise. These materials are designed to achieve certain pre-determined learning outcomes, namely goals and objectives that are contained in an instructional plan. Since the teaching process is affected over a distance, there is need to ensure that students actively participate in their learning by performing specific tasks that help them to understand the relevant concepts. Therefore, a set of exercises is built into the teaching repertoire in order to link what students and tutors do in the framework of the course outline. These could be in the form of students' assignments, a research project or a science practical exercise. Examples of instructional activities in distance education are too numerous to list. Instructional activities, when used in this context, help to motivate students, guide and measure students' performance (continuous assessment)

PREFACE

We have put in lots of hard work to make this book as user-friendly as possible, but we have not sacrificed quality. Experts were involved in preparing the materials. However, concepts are explained in easy language for you. We have included many tables and examples for easy understanding.

We sincerely hope this book will help you in every way you expect.

All the best for your studies from our team!

SOFTWARE ENGINEERING

Contents

BLOCK 1 : INTRODUCTION TO SOFTWARE ENGINEERING

Unit 1 Introduction to Software & Software Engineering

Introduction, The Need for Software Engineering, Goals of Software Engineering, Typical Software Engineering Tasks, Characteristics of Good Software, Software Development Life Cycle, Software Model : Classical Waterfall Model, Iterative Model, Prototyping Model, Evolutionary Model, Spiral Model

Unit 2 Introduction to Software Project Management

Introduction, Introduction to Software Project Management, History of Project Management, Software Project, Need of Software Project Management, Software Project Manager, Sub–Team needed in Software Engineering Projects, Software Management Activity, Project Preparation, Resource Organization, Risk Organization, Project Implementation and Monitoring, Project Communication Organization, Configuration Organization, Concept of Tailoring, Extreme Programming

Unit 3 Software Project Planning Tools and Techniques

Introduction, Work Breaks Down Structure (WBS), Software Sizing, Milestones Baseline, Cost Estimation , Ray Leigh Curve, LOC and FP Estimation, Documenting the Schedule, Developing the Activity Network, Empirical Relationships, Effort Distribution, Empirical Estimation Techniques – COCOMO

Unit 4 Software Project Maintenance

Introduction, Software Configuration Management, Why do we need Configuration Management ?, Element of Configuration Management, Participant of Software Configuration Management, Software Maintenance Processes, Project Planning, Documentation Standards, Version Control

BLOCK 2 : SOFTWARE REQUIREMENT, DESIGN, QUALITY MANAGEMENT & SOFTWARE TESTING

Unit 5 Software Requirement

Introduction, Requirement Engineering, Requirement Engineering Process, Feasibility Study, Requirement Gathering, Software Requirement Specification (SRS),

Software Requirement Validation, Requirement Initiation Process, Requirement Initiation Techniques, Interview, Questionnaires, Observation, Document Review, Software Requirement Characteristics, Software Requirements, Functional Requirements, Non–Functional Requirements, User Interface Requirements, Software System Analyst, Role of System Analyst OR What a System Analyst Does ?, Attributes of a Good System Analyst OR Qualities of System Analyst

Unit 6 Software Design

Introduction, Software Design Basic, Software Design Level, Modularization, Concurrency, Coupling and Cohesion, Design Verification, Software Design Strategies, Structured Design, Function Oriented Design, Object Oriented Design, Software Design Approaches, Software User Interface Design, Command Line Interface (CLI), Graphical User Interface, User Interface Design Activities, GUI Implementation Tools, User Interface Golden Rules, Software Design Complexity, Halsted's Complexity Measures, Cyclomatic Complexity Measures, Function Point, Software Implementation, Structured Programming, Functional Programming, Programming Style, Software Documentation, Software Implementation Challenges

Unit 7 Software Quality Management

Introduction, Software Quality, Verification & Validation (V & V), Quality Control, Inspection, Walkthrough and Review, Why Standards ?, Software Quality Metrics or Parameters, Five levels of Capability Maturity Model (CMM)

Unit 8 Software Testing Technique

Introduction, Software Validation & Verification, Manual VS Automated Testing, Testing Approaches, Black–Box Testing, White–Box Testing, Testing Levels, Unit Testing, Integration Testing, System Testing, Acceptance Testing, Regression Testing, Function Test Plan, Process of Testing, Testing Documentation, Before Testing, While

Being Tested, After Testing, Grey Box Testing, Non–Functional Testing, Testing Artifacts

BLOCK 3 : SOFTWARE RISK ANALYSIS & MANAGEMENT

Unit 9 Software Risk Analysis

Introduction, Risk Analysis in Project Management, Risk Identification, Qualitative Risk Analysis, Quantitative Risk Analysis

Unit 10 Software Risk Management – I

Introduction, Software Risk Management Implementation, Planning Risk Responses, Monitoring and Controlling Risks

Unit 11 Software Risk Management – II

Introduction, Human Resource and Risk Management, The HR Executive and Risk Control, Team Risk Management

BLOCK 4 : CASE STUDIES

Unit 12 Case Study – I : Waste Management Inspection Tracking System

Introduction, Waste Management System, Basic Project Plan, Project Estimates, Risk Management, Project Schedule, Project Team Organization, Tracking and Control Mechanism

Unit 13 Case Study – II : Library Management System

Introduction, Library Management System, Objective, Project Life Cycle, Existing System, Proposed System, Requirement Determining, Development Phase, Design of System Model, Conceptual Model of our Proposed Library Management System

Unit 14 Case Study – III : Software Project Management

Introduction, Measuring a Software Project, Rapid Application Development (RAD) Method, Prototype Method, Agile Scrum Method, Hospital Management System

Dr. Babasaheb Ambedkar Open University Ahmedabad

BCAR-402

Software Engineering

BLOCK 1 : INTRODUCTION TO SOFTWARE ENGINEERING

UNIT 1 INTRODUCTION TO SOFTWARE & SOFTWARE ENGINEERING

UNIT 2 INTRODUCTION TO SOFTWARE PROJECT MANAGEMENT

UNIT 3 SOFTWARE PROJECT PLANNING TOOLS AND TECHNIQUES

UNIT 4 SOFTWARE PROJECT MAINTENANCE

INTRODUCTION TO SOFTWARE ENGINEERING

Block Introduction :

In this block, we will detail about the software, characteristics of good software, software engineering, goals, need for software engineering, software development life cycle, and the software model such as classical waterfall model, iterative model, prototyping model, evolutionary model and spiral model.

Software Project Management applies to information of plan, measure and control the project in order to deliver on time and with required budget. It involves accumulation of requirements, risk, monitoring and controlling progress and following a software development process. Software project planning encompasses estimation, risk analysis, scheduling and SQA/SCM planning.

In this block, we will detail about the basic of Process based Management and idea about Standard Software Development Process. The block will focus on the study and concept of Managing the Software Development Process and identifying the Software Model. You will give an idea on Milestones Baseline, Cost Estimation, Ray Leigh Curve and Documenting the Schedule.

After studying this block, you will make to learn and understand about the basic of Empirical Estimation Techniques such as COCOMO model. The concept related to Software Configuration Management and Maintenance Processes along with Version Control are also explained to the students. The student will be demonstrated about Empirical Relationships techniques.

Block Objectives :

After learning this block, you will be able to understand :

- Software & Software Engineering
- Need & Goal of Software Engineering
- Characteristics of Good Software
- Software Development Life Cycle
- Knowledge related to Waterfall Model, Iterative Model, Prototyping Model, Evolutionary Model, Spiral Model
- Idea about Software Project Planning Tools and Techniques
- Idea about Software Configuration Management
- Project Scheduling
- Resource Management
- Risk Management
- Software Maintenance Processes
- Documentation Standards
- Version Control

Block Structure :

Unit 1 : Introduction to Software & Software Engineering

Unit 2 : Introduction to Software Project Management

Unit 3 : Software Project Planning Tools and Techniques

Unit 4 : Software Project Maintenance

Unit 01

INTRODUCTION TO SOFTWARE & SOFTWARE ENGINEERING

UNIT STRUCTURE

1.0 Learning Objectives

1.1 Introduction

1.2 The Need for Software Engineering

1.3 Goals of Software Engineering

1.4 Typical Software Engineering Tasks

1.5 Characteristics of Good Software

1.6 Software Development Life Cycle

1.7 Software Model

- **1.7.1 Classical Waterfall Model**
- **1.7.2 Iterative Model**
- **1.7.3 Prototyping Model**
- **1.7.4 Evolutionary Model**
- **1.7.5 Spiral Model**

1.8 Let Us Sum Up

1.9 Answers for Check Your Progress

1.10 Glossary

1.11 Assignment

1.12 Activities

1.13 Case Study

1.14 Further Readings

1.0 Learning Objectives :

After learning this unit, you will be able to understand :

- Introduction about the Software and Software Engineering
- Need, Goals, and Tasks of Software Engineering
- Detail idea about Software Development Life Cycle
- Idea about various Software Model

1.1 Introduction :

Software Engineering is about methods, tools and techniques used for developing software. Software surrounds us everywhere in the industrialized nations – in domestic appliances, communications systems, transportation systems and in businesses. Software comes in different shapes and sizes – from the program in a mobile phone to the software to design a new automobile.

Let us understand what Software Engineering stands for. The term is made of two words, **software** and **engineering**.

Software is program which is a collection of related instructions organized for a common purpose. Software accept input from the user and gives meaningful output.

On the other hand, **engineering** is a process in which product development by considering the analysis, development models and methods.

Software Engineering

Software engineering is an engineering in which product development is carry on by the developers and team by considering the analysis with development models and methods, which then going to test by the tester and release to the customer.

Software engineering is a discipline for solving business problems by designing and developing software–based systems.

❑ **Check Your Progress – 1 :**

1. _____ is program which is a collection of related instructions organized for a common purpose.

   a. Software b. Instruction c. Project d. None of Above

2. _____ is a process in which product development by considering the analysis, development models and methods.

   a. Software Development b. System Development

   c. Engineering d. All of Above

1.2 The Need for Software Engineering :

The nature of computer software has changed considerably in the last forty–five or so years, with accelerated changes in the last fifteen to twenty.

We begin with a brief history. In the late 1970s and early 1980s, personal computers were just beginning to be available at reasonable cost. There were many computer magazines available at newsstands and bookstores; these magazines were filled with articles describing how to determine the contents of specific memory locations used by computer operating systems. Other articles described algorithms and their implementation in some dialect of the BASIC programming language. High school students sometimes made more money by programming computers for a few months than their parents made in a year. Media coverage suggested that the possibilities for a talented, solitary programmer were unlimited. It seemed likely that the computerization of society and the

fundamental changes caused by this computerization were driven by the actions of a large number of independently operating programmers.

However, another trend was occurring, largely hidden from public view. Software was growing greatly in size and becoming extremely complex. The evolution of word processing software is a good illustration.

The explosive growth of personal computers is in the 1980s and 1990s. The growth has continued to the present day, with computing now done on smartphones, tablets, and other devices. As indicated, many of the initial versions of some of the earlier software products have evolved into very large systems that require more effort than one individual can hope to devote to any proj–ect.

A moment's thought might make you think that standards such as Hypertext Markup Language (HTML) and the Java programming language with its application programming interfaces have changed everything. There are more recent developments. Application frameworks and cloud computing have been major players, along with highly mobile devices such as smartphones and tablets with their specialized user interfaces.

There are also inexpensive, high–quality application development frameworks and software development kits. There are many sixteen–year–olds who are making a large amount of money as web page designers, although this is relatively less frequent in 2014, since many elementary school students in the United States are taught how to design a web page before they reach their teens. One of the job skills most in demand now is "web master"–a job title that did not even exist in 1993. A casual reading of online job listings might lead you to believe that we have gone back to the more freewheeling days of the 1980s.

Certainly, the effects of technology have been enormous. It is also true that nearly anyone who is so inclined can learn enough HTML in a few minutes to put together a flashy web page.

However, the problem is not so simple. Even the most casual user of the Internet has noticed major problems in system performance. Delays make waiting for glitzy pictures and online animations very unappealing if they slow down access to the information or services that the user desired. They are completely unacceptable if they cannot display properly on many portable devices with small screen "real estate."

Proper design of websites is not always a trivial exercise. As part of instruction in user interface design, a student of mine was asked to examine the main websites of a number of local universities to obtain the answer to a few simple questions. The number of selections (made by clicking a mouse button) ranged from five to eleven for these simple operations. Even more interaction was necessary in some cases because of the need to scroll through online documents that were more than one screen long, or even worse, more than one screen wide. Efficiency of design is often a virtue.

Several issues may not be transparent to the casual user of the Internet. Perhaps the most troublesome is the lack of systematic configuration management, with servers moving, software and data being reorganized dynamically, and clients not being informed. Who has not been annoyed by the following famous message that appears so often when attempting to connect to an interesting website ?

It is obvious what happened to the information that the user wanted, at least if there was no typing error. As we will see later in this book, maintenance of websites is often a form of "configuration management," which is the systematic treatment of software and related artifacts that change over time as a system evolves.

There are also major issues in ensuring the security of data on servers and preventing unwanted server interaction with the user's client computer. Finally, designing the decomposition of large systems into client and server subsystems is a nontrivial matter, with considerable consequences if the design is poor.

It is clear that software engineering is necessary to have modern software development done in an efficient manner. These new technologies have refocused software engineering to include the effects of market forces on software development. As we will see, these new technologies are amenable to good software engineering practice.

❑ **Check Your Progress – 2 :**

1. _____ is necessary to have modern software development.

   a. Software b. Software Engineering

   c. Software Structure d. None of Above

1.3 Goals of Software Engineering :

Clearly organizations involved with producing software have a strong interest in making sure that the software is developed according to accepted industry practice, with good quality control, adherence to standards, and in an efficient and timely manner.

For some organizations, it is literally a matter of life and death, both for the organization and for potential users of the software. *Software engineering* is the term used to describe software development that follows these principles.

Specifically, the term *Software engineering* refers to a systematic procedure that is used in the context of a generally accepted set of goals for the analysis, design, implementation, testing, and maintenance of software.

The software produced should be efficient, reliable, usable, modifiable, portable, testable, reusable, maintainable, interoperable, and correct.

- **Efficiency :** The software is produced in the expected time and within the limits of the available resources. The software that is produced runs within the time expected for various computations to be completed.
- **Reliability :** The software performs as expected. In multiuser systems, the system performs its functions even with other load on the system.
- **Usability :** The software can be used properly. This generally refers to the ease of use of the user interface but also concerns the applicability of the software to both the computer's operating system and utility functions and the application environment.
- **Modifiability :** The software can be easily changed if the requirements of the system change.

- **Portability :** The software system can be ported to other computers or systems without major rewriting of the software. Software that needs only to be recompiled in order to have a properly working system on the new machine is considered to be very portable.
- **Testability :** The software can be easily tested. This generally means that the software is written in a modular manner.
- **Reusability :** Some or all of the software can be used again in other projects. This means that the software is modular, that each individual software module has a well–defined interface, and that each individual module has a clearly defined outcome from its execution. This often means that there is a substantial level of abstraction and generality in the modules that will be reused most often.
- **Maintainability :** The software can be easily understood and changed over time if problems occur. This term is often used to describe the lifetime of long–lived systems such as the air traffic control system that must operate for decades.
- **Interoperability :** The software system can interact properly with other systems. This can apply to software on a single, stand–alone computer or to software that is used on a network.
- **Correctness :** The program produces the correct output.

❑ **Check Your Progress – 3 :**

1. The software performs as expected is known as _____.

   a. Reusability b. Portability c. Reliability d. Efficiency

2. The software can be easily understood and changed over time if problems occur is known as _____.

   a. Maintainability b. Correctness c. Reusability d. Interoperability

3. Write a detail note on goals of Software Engineering.

_____

_____

_____

_____

_____

1.4 Typical Software Engineering Tasks :

There are several tasks that are part of every software engineering project :

- Analysis of the problem
- Determination of requirements
- Design of the software
- Coding of the software solution
- Testing and integration of the code
- Installation and delivery of the software
- Documentation

- Maintenance
- Quality assurance
- Training
- Resource estimation
- Project management

Analysis of a problem is very often undertaken by experts in the particular area of application.

The **requirements** phase of an organization's software life cycle involves precisely determining what the functionality of the system will be. If there is a single customer, or set of customers, who is known in advance, then the requirements process will require considerable discussion between the customer and the requirements specialists on the software team.

The **design** phase involves taking the requirements and devising a plan and a representation to allow the requirements to be translated into source code. A software designer must have considerable experience in design methodology and in estimating the trade–offs in the selection of alternative designs. The designer must know the characteristics of available software systems, such as databases, operating systems, graphical user interfaces, and utility programs that can aid in the eventual process of coding.

Coding activity is most familiar to students. We note, however, that many decisions about coding in object–oriented or procedural languages might be deferred until this point in the software life cycle, or they might have been made at the design or even the requirements phase. Since coding standards are often neglected in many first– and second–year programming courses, and they are critical in both large, industrial–type systems and for small apps that are evaluated for quality in an app store before they are allowed to be placed on sale.

Software testing is an activity that often begins after the software has been created but well before it is judged ready to be released to its customer. It is often included with software integration, which is the process of combining separate software modules into larger software systems.

Documentation includes much more than simply commenting the source code. It involves rationales for requirements and design, help files, user manuals, training manuals, technical guides such as programming reference manuals, and installation manuals.

The term **software maintenance** is used to describe those activities that are done after the software has been released. Maintenance includes correcting errors found in the software; moving the software to new environments, computers, and operating systems; and enhancing the software to increase functionality.

Quality assurance, or QA, is concerned with making certain that both the software product that is produced and the process by which the software is developed meet the organization's standards for quality. The QA team is often responsible for setting the quality standards. In many organizations, the QA team is separate from the rest of the software development team.

❑ **Check Your Progress – 4 :**

1. Software engineering involves _____.

   a. Design b. Coding

   c. Documentation d. All of Above

2. Explain tasks involves in Software Engineering.

1.5 Characteristics of Good Software :

A software is identified by what it gives and how well it can be used. The software must satisfy the following criteria:

- **Functionality**
- **Transitional**
- **Maintainability**

- **Functionality :** It refers to the performance of the software. Performance can be measured as usability, correctness, dependability, security and safety of the software.
- **Transitional :** It refers when the software is moved from one platform to another means cross platform. It can be measured as an adaptability, interoperability, portability and reusability.
- **Maintainability :** It refers to as ease modification can be done in software to extend the functionality of the software. In this maintainability modularity, flexibility and scalability play major role.

❑ **Check Your Progress – 5 :**

1. Explain characteristics of good software.

1.6 Software Development Life Cycle :

Software development is a process of consisting of two major parts which are as follows : *System Analysis* and *System Design*. When Management of organization feels that a new system or improvement in existing system is required, at that time the phase of system development starts.

"Software Development Life Cycle is a set of activities that analysts, designers and users carry out to develop and implement an information system".

Software Development Life Cycle method consists of Six phase which are as follows :

1. **Preliminary Investigation :** The first phase of SDLC is Preliminary Investigation. Before any system are developed users, managers or concerned person have to request for the particular system to be developed. This activity is called preliminary investigation. When the request is made at that time the preliminary investigation begins. There are three parts of Preliminary Investigation which are as follows :

   1.1 **Request Clarification :** This is the main or very important phase of preliminary investigation. The user and employee should be cleared about actual requirement because many requests from the user and employee are not clearly defined. If the requests are not clearly defined then we cannot develop or improve the new system or existing system. Before any further steps can be taken, the project request must be clearly stated.

   1.2 **Feasibility Study :** Before any request is passed or approved a feasibility study is conducted because you can determine that the requested system is feasible or not. There are three aspects in feasibility study which are as follows :

      A. **Technical Feasibility :** This involves the study whether a current equipment or technology is sufficient or not. Can the work for the project be done with current equipment, existing software technology, and available workers ? If new technology is required then what is the possibility that it can be developed?

      B. **Economic Feasibility :** This study finds out whether the request in the particular project will be beneficial in future or not. If the return on the investment is good then it is advisable to start the project.

      C. **Operational Feasibility :** User acceptance level is checked in this study. This phase also determines whether user will accept or reject a new system. Will the system be used if it is developed and implemented ?

      The feasibility study carried out by a small group of people, who are familiar with information system techniques; understand the part of the business or organization.

   1.3 **Request Approval :** It is not necessary that all requested projects are desirable or feasible. However, those projects are not feasible and desirable should be put into schedule. The management decides which projects are most urgent and schedule them accordingly. After a project request is approved, its cost, priority, completion time, and workers requirement are estimated.

2. **Determination of System Requirements :** At the heart of system analysis is a detailed understanding of all important fact of business are under investigation. Analysts, working closely with employees and managers, must study the existing business process to answer the following questions :

   ✓ What is being done ?

   ✓ How it is being done ?

   ✓ How frequently does it occur ?

   ✓ How great is the volume (amount) of transaction or decisions ?

- ✓ How well is the task being performed ?
- ✓ Does a problem exist ?
- ✓ If a problem is existing, how series is it ?
- ✓ If a problem is existing, what is the underlying (basic) cause ?

To answer the above questions, system analyst talks to variety of person together to collect the details about the business process and their opinions. Some tools are used in analysis like data flow diagrams, interviews, on–site observations and questionnaires. Following is the Information Sources :

- ✓ User of a system.
- ✓ Forms and documents used in the organization.
- ✓ Procedure manuals and rulebook, which specifies how various activities are carried out in the organization.
- ✓ Various reports used in the organization.
- ✓ Computer programs of existing systems.

3. **Design of System / System Design :** The design of an information system produces the details that state how a system will meet the requirements identified by during system analysis. There are two types of system design which are as follows:

   3.1 Logical Design : The process of how system will meet the requirements identified during system analysis is known as a Logical Design. System Specialists often refers to this stage as Logical Design.

   3.2 Physical Design : The process of developing program software, which is known as Physical Design. System analysts start the process of designing by identifying reports and other outputs the system will produce. During the design phase, the logic of the program is designed, files or database are designed, and program testing and implementation plans and draw up. Designers are responsible for providing the program with complete and clearly outline software applications.

4. **Development of Software :** Software development process involves installation of software, purchase of software or they may write completely new software. The choice depends on the cost of each opinion, the time available to write software, and the availability of programmers. Programmers are also responsible for documenting the program, providing an explanation of how and why certain procedures are coded in specific ways. Documentation is essential to test the program and carry–on maintenance once the application has been installed.

5. **Software Testing :** During software testing, the software is used experimentally to ensure that the software works perfectly. Data are input in the software and the result of software is checked for that data. Testing is performed by analyst, users and persons who are not at all connected with the system.

6. Implementation and Evaluation :

6.1 Implementation : Implementation is the process of having system workers to check out, and put new equipment into use, train users, install the new applications and construct any files or data needed to use it. During this phase, they will run both old and new systems in parallel way to complete the result.

6.2 Evaluation : Evaluation of the system is performed to identify its strength and weakness and a plan for its improvements is draw up. Evaluation consists of four phase which are as follows:

A. Operational Evaluation : Measurement of the way in which software functions including difficulty in use, response time, suitability of information formats, overall reliability, and level of utilization.

B. Organizational Impact : Identification and measurement of the benefits to organization in such areas as financial concern, operational efficiency, and competitive effects include the effects of internal and external information flow.

C. User Manager Assessment : It is the evaluation of attitude of senior and user managers within the organization, as well as end–users.

D. Development Performance : It is the evaluation of overall development time and efforts Conformance to budgets and standards, and other project management criteria.

❑ **Check Your Progress – 6 :**

1. Full form of SDLC is _____.
   a. Software Development Life Cycle
   b. Software Design Life Cycle
   c. Software Documentation Life Cycle
   d. Software Decision Life Cycle
2. Explain SDLC with its all phase.

1.7 Software Model :

A software life cycle model also called process model is an expressive and pictorial representation of the software life cycle. A life cycle model represents all the activities required to make a software product transit through its life cycle phases. It also captures the order in which these activities are to be undertaken. In other words, a life cycle model maps the different activities performed on a software product from its inception to retirement.

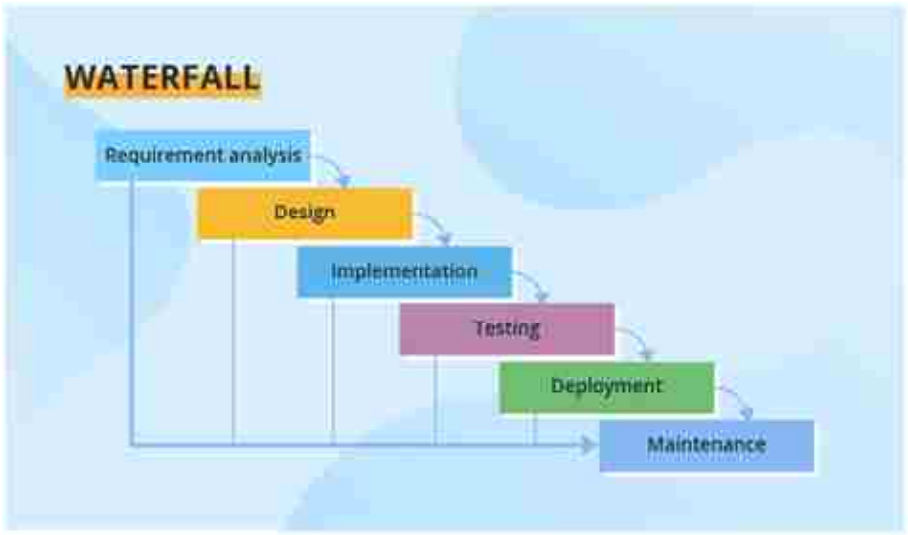
Different life cycle models may map the basic development activities to phases in different ways. Thus, no matter which life cycle model is followed,

the basic activities are included in all life cycle models though the activities may be carried out in different orders in different life cycle models. During any life cycle phase, more than one activity may also be carried out.

1.7.1 Classical Waterfall Model :

The waterfall model is very simple to understand and use, it was first process model to be announced. It is also known as liner sequential life cycle model. In a waterfall model, you cannot start with next phase before completed previous phase. It is used for the project which is small and there are no undefined requirements. After completion of the certain phases a baseline is established that "freezes" the product of development at that point. If need of a change is identified, a formal change process is followed to make the change. Output from each phase includes documentation.

The phases below the detailed design phase include software as part of their output. Transition from phase to phase is accomplished by holding formal reviews which provide insight into the progress. Baselines are established at critical points on the waterfall model, the last of which is the product baseline. The Final baseline is going with audits.



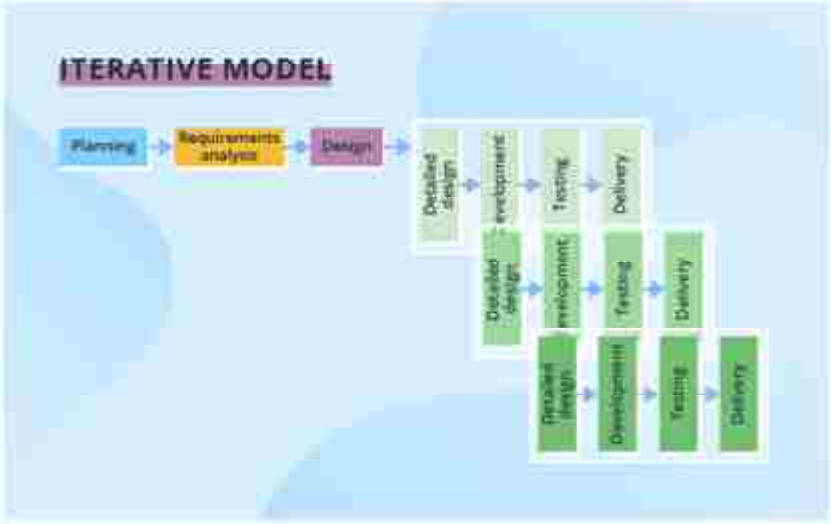
Waterfall Model

- **Advantages :**
  - Easy to implement and maintain.
  - The initial phase of difficult examination of requirements and systems helps in saving time later in the developmental phase.
  - The requirement of resources is minimal and testing is done after completion of each phase.
- **Disadvantages :**
  - It is not possible to alter or update requirements.
  - You cannot make changes once you are into the next phase.
  - Cannot start the next phase until the previous phase is completed.

1.7.2 Iterative Model :

In this model, iteration play an important role in software development process. Here iteration means repetition of every step in a cyclic way after every cycle of SDLC process.

In this Model, you can start with some of the software specifications and develop the first version of the software. After the first version if there is a need to change the software, then a new version of the software is created with a new iteration. Every release of the Iterative Model finishes in an exact and fixed period that is called iteration.



Iterative Model

- **Advantages :**

– It is easier to control the risks as high–risk tasks are completed first.

– The progress is easily measurable.

– Problems and risks defined within one iteration can be prevented in the next sprints.

- **Disadvantages :**

– Iterative model requires more resources than the waterfall model.

– The process is difficult to manage.

– The risks may not be completely determined even at the final stage of the project.

1.7.3 Prototyping Model :

Prototype is a working system – not just an idea on paper – that is developed to test ideas and assumptions about the new system.

- **Steps in prototyping :**

✓ Identify the user's known information requirements and features need in the system.

✓ Develop a working prototype.

✓ Use the prototype, and expand the lists of known system requirements.

✓ Revise the prototype based on information gained through user experience.

✓ Repeat these steps as needed to achieve a satisfactory system.

✓ As the steps suggest, prototyping is not a trial – and – error development method.

Both users and analyst collect the sufficient information from the prototype used. And determine to meet the requirements they have collected. Usually, one of the four alternatives is selected.

Prototype Model

- **Advantages :**

– Prototyping involves the use of labor–saving software; this software has embedded database management system.

– Even personnel computer can provide an effective approach to prototyping as there is no interfere from another user database can be used.

– Prototyping can be used with SDLC or in combination with SSADM or independently develop a new system.

- **Disadvantages :**

– Leads to implementing and then repairing way of building systems.

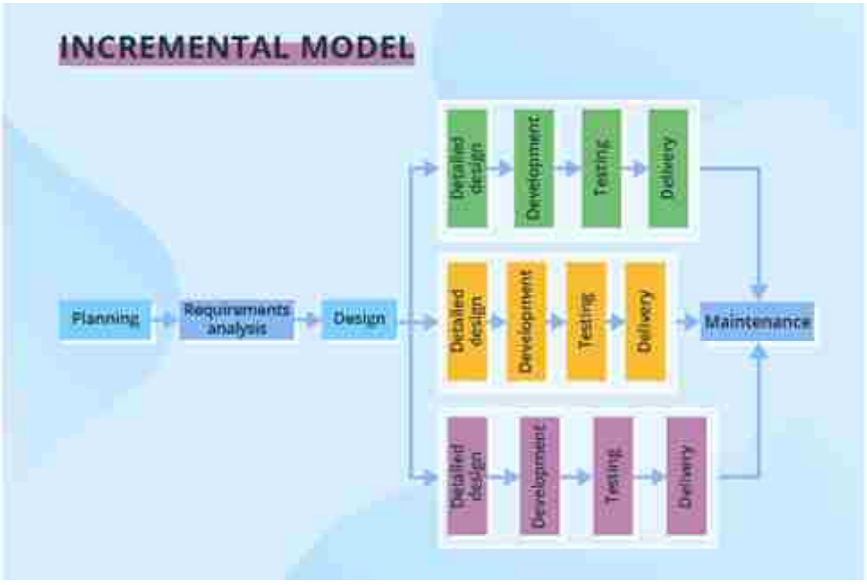
– Practically, this methodology may increase the complexity of the system as scope of the system may expand beyond original plans.

– Incomplete application may cause application not to be used as the full system was designed Incomplete or inadequate problem analysis.

1.7.4 Evolutionary Model :

It is also called successive versions model or incremental model. At first, a simple working model is built. Subsequently it undergoes functional improvements & we keep on adding new functions till the desired system is built.

Large projects where you can easily find modules for incremental implementation. Often used when the customer wants to start using the core features rather than waiting for the full software. Also used in object–oriented software development because the system can be easily portioned into units in terms of objects.



Incremental Model

- **Advantages :**

– User gets a chance to experiment partially developed system

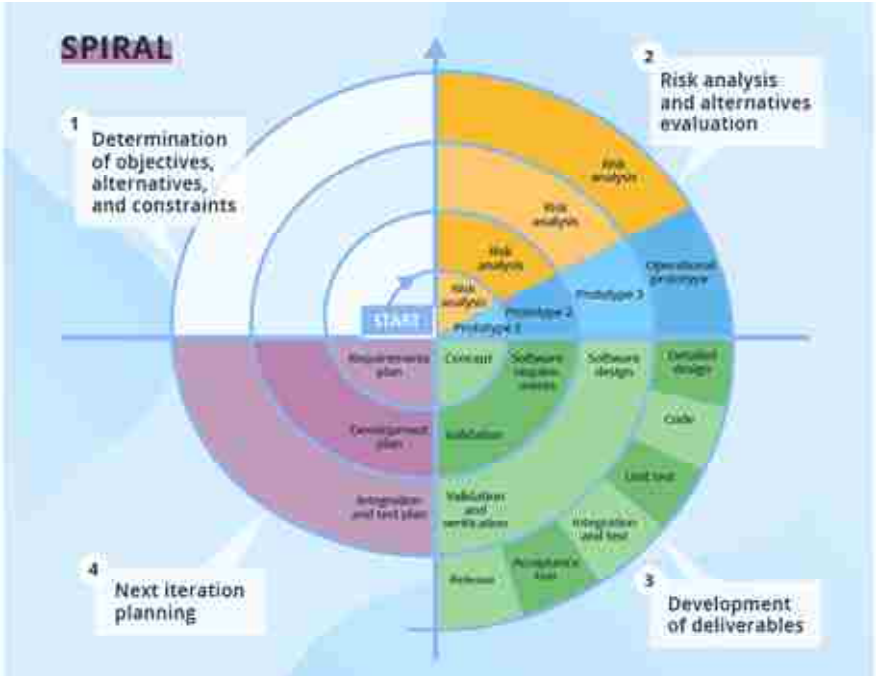
– Reduce the error because the core modules get tested thoroughly.

- **Disadvantages :**

– It is tough to divide the problem into numerous types that would be suitable to the customer which can be incrementally executed & delivered.

1.7.5 Spiral Model :

The spiral model is proposed by Barry Boehm in which prototyping is used to control cost. The Boehm spiral model has become quite popular among ADE (Aerospace Defense and Engineering) specialists, and is not so familiar among business developers. It is particularly useful in projects, which are risky in nature. Business projects are more traditional. They tend to use mature technology and to work well–known problems.



Spiral Model

- **Advantages :**

– Risk avoidance chance is enhanced due to the importance on risk analysis.

– It's a good model for complex and large systems.

– Depending on the changed circumstances additional functionalities can be added later on.

– Software is produced early in the cycle.

- **Disadvantages :**

– It's a costly model and requires highly specialized expertise in risk analysis.

– It does not work well in simpler projects.

❑ **Check Your Progress – 7 :**

1. Write a note on Waterfall Model.

2. Write a note on Prototype Model.

3. Explain Spiral Model.

4. _____ model also known as linear – sequential life cycle model.

   a. Incremental Model b. Iterative Model

   c. Prototype Model d. Waterfall Model

1.8 Let Us Sum Up :

In this unit we have learnt that Software engineering is an engineering in which product development is carry on by the developers and team by considering the analysis with development models and methods, which then going to test by the tester and release to the customer.

We have also seen about the need, goal and task of the software engineering. It is noted that software development life cycle is a task which is a mixture of many operations to having goal in terms of software development and delivery.

It requires trained and experienced Engineers which will help to increase likelihood of project success as development for large projects is complex and follows definite principles in reducing risks linked with project.

1.9 Answers for Check Your Progress :

- **Check Your Progress 1 :**

  1. (a) 2. (c)

- **Check Your Progress 2 :**

  1. (b)

- **Check Your Progress 3 :**

  1. (c), 2. (a) 3. (Refer 1.3)

- **Check Your Progress 4 :**

  1. (d), 2. (Refer 1.4)

- **Check Your Progress 5 :**

  1. (Refer 1.5)

- **Check Your Progress 6 :**

  1. (a), 2. (Refer 1.6)

- **Check Your Progress 7 :**

  1. (Refer 1.7.1) 2. (Refer 1.7.3)

  3. (Refer 1.7.5) 4. (d)

1.10 Glossary :

1. **Software –** It is program which is a collection of related instructions organized for a common purpose. Software accept input from the user and gives meaningful output.
2. **Engineering –** It is a process in which product development by considering the analysis, development models and methods.
3. **Software Engineering –** It is an engineering in which product development is carry on by the developers and team by considering the analysis with development models and methods, which then going to test by the tester and release to the customer.

1.11 Assignment :

1. Explain Software Development Life Cycle in detail.

1.12 Activities :

1. Differentiate all the Software Model.

1.13 Case Study :

Comment, Incremental model is more flexible than the waterfall model.

1.14 Further Reading :

1. Software Engineering: A Practitioner's Approach Book by Roger S. Pressman
2. W. Shewhart, Statistical Method from the Viewpoint of Quality Control, Dover, 1986.
3. W.E. Deming, Out of the Crisis, SPC Press, 1982; reprinted in paperback by MIT Press, 2003.
4. T. Gilb, Software Metrics, Little, Brown, and Co., 1976.
5. R. Zultner, "The Deming Approach to Quality Software Engineering," Quality Progress, vol. 21, no. 11, 1988, pp. 58–64.
6. W.H. Dana, The X–15 Lessons Learned, tech. report, NASA Dryden Research Facility, 1993

Unit 02

INTRODUCTION TO SOFTWARE PROJECT MANAGEMENT

UNIT STRUCTURE

2.0 Learning Objectives

2.1 Introduction

2.2 Introduction to Software Project Management

2.3 History of Project Management

2.4 Software Project

2.5 Need of Software Project Management

2.6 Software Project Manager

2.7 Sub–Team needed in Software Engineering Projects

2.8 Software Management Activity

2.9 Project Preparation

2.10 Resource Organization

2.11 Risk Organization

2.12 Project Implementation and Monitoring

2.13 Project Communication Organization

2.14 Configuration Organization

2.15 Concept of Tailoring

2.16 Extreme Programming

2.17 Let Us Sum Up

2.18 Answers for Check Your Progress

2.19 Glossary

2.20 Assignment

2.21 Activities

2.22 Case Study

2.23 Further Readings

2.0 Learning Objectives :

After learning this unit, you will be able to understand :

- Software Project management
- Software Development Process
- Software Project and Need
- Software Project Manager and Software Management Activity
- Resource Management
- Configure Management
- Risk analysis and measurement
- Basics of Extreme programming

2.1 Introduction :

A project is a collection of numerous processes performed in order to complete a goal in terms of software development and delivery. Characteristics of project are as follows :

- ✓ Exclusive and different goal.
- ✓ No day–to–day operations.
- ✓ Start time and end time.
- ✓ Goal achieved which serve as temporary phase in lifetime of an organization.
- ✓ Acceptable resources in terms of time, workforce, money, material and knowledge–bank.

2.2 Introduction to Software Project Management :

A software project is the whole process of software development form gathering requirement to Implementation and evaluation, which is carried out according to steps of execution procedures, in a stated time to complete proposed software product.

The goal of software project management is to understand, plan, measure and control the project such that it is delivered on time and on budget. This involves gathering requirements, managing risk, monitoring and controlling progress, and following a software development process.

Software project management requires trained and experienced Software Engineers in order to increase the likelihood of project success because software development for large projects is extremely complex and following strict engineering principles will help reduce the risks associated with the project.

Software project management is extremely important as :

- ✓ It is difficult to predict as only 10% of projects are delivered in particular budget and on schedule.
- ✓ Management result in more effect on success or failure of project than technology advances.
- ✓ There exists too much scrap and rework which results in very immature process.

❑ Check Your Progress – 1 :

1. What is the goal of Software Project Management ?
   a. To build a Software
   b. To plan, measure and control the project development
   c. Maintenance of Software Project
   d. None of the above
2. What is Software Project Management ? Explain

2.3 History of Project Management :

However, project management has been around for thousands of years. Project management was involved in the planning, coordination, and construction of the Ancient Wonders of the World. However, project management has grown to include today's energy production sectors, construction efforts, and more.

Project Management in 18th Century : Elements of Project Management Transcend Time. The basic principles of project management have remained the same throughout history, regardless of technology and capacity. These elements include managing resources, maintaining schedules, and coordinating of different activities and tasks. However, ancient and other historic marvels of project management do not routinely involve schedule optimization.

Project Management in 19th Century : In the late 19th century, the need for more structure in construction, manufacturing, and transportation sectors gave rise to modern project management tactics. For example, the creation of the Transcontinental Railroad, corduroy roads, and the rebuilding of the South after the Civil War were primary feats of project management applications.

The Transcontinental Railroad is considered to be the first, large–scale project management undertaking. Often, people forget about the true scope of this project as it included traversing treacherous terrain and weather to construction a railway and telegraph line.

Project Management in 1900 to 1950 : The Birth of Modern Project Management and Henry Gantt.

As the 19th century progressed, business leaders began to face the challenges of labor laws and regulations from the federal government. Henry Gantt, considered the founding father of modern project management, developed planning and control techniques, such as the famous Gantt Chart to ensure monitoring and control of the project schedule. This basic bar chart shows the phases of a project from inception to completion.

Project Management in 1911 under Frederic Taylor : Frederic Taylor published a book, "The Principles of Scientific Management," in 1911, which was based on his experience in the steel industry. The goal of the book was to give unskilled workers to opportunity to work on new, complex projects by learning skills rapidly and through simplicity.

In addition, he identified how many workers would routinely work below capacity through soldiering to ensure future job security. Furthermore, he identified the need to create incentive–based wage systems and take advantage of time saving techniques.

Project Management in 1950 to 1980s : PERT and CPM : After WWII, project managers began to follow two mathematical ways of conducting and managing projects. Program Evaluation Review Technique, or PERT, analyzes individual tasks by asserting a minimum amount of time for completion. The Critical Path Method, or CPM, factored in all activities, the completion time of such activities, and how they relate to identify inefficiencies. However, CPM quickly became riddled with confusion.

Project Management in 1980 to 2000 : Computers and Project Management: Computers brought connectivity and communication to the forefront of project management in the 1980s. As technology grew into the 1990s, the

Internet became widely available through dial–up means. Some project management entities created systems for project management purposes, but it was not until the late 19th century when the newfound era of computers and project management truly began.

Project Management in 2000 till today : Rise of Automation and Maturity of Efficiency: As computer–controlled options and complex algorithms were developed, project manager began to complete more work in less time with fewer errors than ever before in history. As the Internet grew, web–based project management applications were developed. Today, web–based project management applications may be seen on mobile devices, individual computers, and wide–scale ERP systems.

❑ **Check Your Progress – 2 :**

1. The principles of scientific management are published by _____.

   a. Henry Gantt b. Frederic Taylor

   c. Both of these d. None of these

2.4 Software Project :

A software project is the whole process of software development form gathering requirement to Implementation and evaluation, which is carried out according to steps of execution procedures, in a stated time to complete proposed software product.

2.5 Need of Software Project Management :

Software is supposed to be an immaterial product. Software development in a new thing in the business world. Most software is made to fulfill requirements of the client. As the technology changes and advances so regularly and fast one software may not be useful to the other one. All such business and environmental restrictions carry risk in software development hence it is important to manage software projects efficiently.

Above image shows three restrictions for software projects. Time, Cost and Quality is an important part of software organization because deliver the project on time as per the schedule looking into consideration of client's budget and deliver the quality product. There are numerous internal and external factors, which may affect these three restrictions.

Therefore, software project management is important to combine user requirements along with budget and time restriction.

❑ **Check Your Progress – 3 :**

1. Software project management is important to combine user requirements along with _____.

   a. Time b. Cost c. Quality d. All of Above

2.6 Software Project Manager :

A software project manager is a person who accepts the responsibility of implementing the software project. Software project manager is aware of SDLC and it's all phase. The project manager controls and manages the activities of software project, but he is not directly involved in development of the final product.

A project manager carefully monitors the process of development, makes and implements various plans, organize required resources, keeps communication among all team members in order to consider issues of cost, resources, time, quality and customer satisfaction.

Responsibilities of project manager are as follows :

- **Managing People**
  - ✓ Act as project leader
  - ✓ Lesion with investors
  - ✓ Managing human resources
  - ✓ Setting up reporting order etc.
- **Managing Project**
  - ✓ Defining and setting up project opportunity
  - ✓ Managing project management activities
  - ✓ Monitoring development and performance
  - ✓ Risk study at each phase
  - ✓ Take required step to avoid or come out of problems
  - ✓ Act as project spokesperson

❑ **Check Your Progress – 4 :**

1. _____ is a person who accepts the responsibility of implementing the software project.

   a. Software Project Manager b. Software Design Manager

   c. Software Testing Manager d. Software Development Manager

2. Write a note on Software Project Manager.

2.7 Sub–Team needed in Software Engineering Projects :

It is clear that the systematic development of large, modern, software projects require teams. It is often helpful to think of a software project team as being made up of several "sub–teams." These sub–teams need not be as formally constituted as they are described here and, in fact, some of them will consist of a single person in many cases. In fact, the same person may perform all the actions of multiple sub–teams. However, the team's activities still need to be performed, regardless of the overall team organization and the size of

the project. The activities performed by these teams occur regardless of the software development life cycle used, although the timing of these activities will almost certainly differ.

Several of these sub–teams are not likely to be in existence during the entire lifetime of the software project. For some software projects, one or more of the teams may consist of a single person whose duties may be split among several projects or activities. In other projects, the teams may be very large and geographically separated. Team members may even report to different companies in some of the larger cooperative software development projects.

Some typical software engineering sub–teams and their duties are listed next. Although all activities are important, this list highlights some cases where a particular software development life cycle model requires special knowledge from a sub–team.

Systems Analysis Team – This team is responsible for determining if the project is feasible. The feasibility study includes cost analysis, estimated revenues, and an estimate of the difficulty of engineering the project. After it produces the feasibility study, this team should interact with the requirements team, receiving its feedback. If the software development process is iterative, as in the rapid prototyping and spiral models, then the interaction and feedback should be more frequent and may occur with additional sub–teams.

Planning Team – This team is responsible for developing the overall management plan for the project and making sure that the project is proceeding within the expected time frame for various activities. This often involves staffing, which becomes especially critical for agile software development processes, to make sure that the team has adequate knowledge of the application domain to know the capabilities of existing software components and systems. The same is true for open–source projects.

Requirements Team – The duties of this team are to meet with the customer and determine a complete, precise set of requirements for this project. This will require a set of formal and informal meetings with the customer to finalize the requirements from relatively imprecise and incomplete initial requirements. If no customer is available, then the requirements team is to obtain the same information from one or more potential users. If no potential users are available, then surrogate customers may be used in their place.

After it produces the system's requirements, this team should interact with the design team, receiving its feedback. If the software development process is iterative, as in the rapid prototyping and spiral models, then the interaction and feedback should be more frequent and may occur with additional sub–teams. This type of feedback is crucial to agile software development processes.

System Design Team – The duties of this team will be to produce a detailed design of the system after the requirements have been set by the requirements team. If the software development process uses the classical waterfall model, then the system design team should provide feedback to the requirements team about any difficulties encountered.

After it produces the design, the system design team should interact with the implementation team, receiving its feedback. If the software development process is iterative, as in the rapid prototyping and spiral models, then the interaction and feedback should be more frequent and may occur with additional sub–teams.

Implementation Team – The duties of this team will be to implement the software designed by the system design team. After they produce the implementation, this team should interact with the testing and integration team, receiving its feedback. If the software development process is iterative, as in the rapid prototyping and spiral models, then the interaction and feedback should be more frequent and may occur with additional sub–teams.

Testing and Integration Team – The duty of this team is the formulation of test cases for the modules and systems that are created by the implementation team. This team may take some modules from an incomplete system for testing by mutual agreement with the implementation team. After it produces the test plan and tests the software modules produced, this team will integrate the software modules into a working system. This team should interact with the implementation team, receiving its feedback. If the software development process is iterative, as in the rapid prototyping and spiral models, then the interaction and feedback should be more frequent and may occur with additional sub–teams. The integration team is responsible for an interface control document (ICD) that precisely describes the interfaces between major system components. This can be in the form of specifying APIs, or the interfaces between software subsystems.

Training Team – This team is responsible for the development and production of training materials.

Delivery and Installation Team – This team is responsible for the delivery and installation of the software.

Maintenance Team – This team is responsible for the maintenance of the software after it is delivered and installed. After the system is delivered and installed, this team should interact with the implementation team, receiving its feedback. If the software development process is iterative, as in the rapid prototyping and spiral models, then the interaction and feedback should be more frequent and may occur with additional sub–teams.

Quality Assurance (QA) Team – This team has two duties. The first is to set standards for the adherence of the project team to a set of agreed–upon processes for the system's creation and set standards for performance of the software produced. The second is to provide an evaluation of how well the project teams meet those standards. Standard industry practice is for the information obtained by this team to be kept internal and not shared with the customer. The information can be released in the event of a legal action and, thus, cannot be destroyed. This information is to be presented to the project manager who will use it to evaluate performance of the QA team.

Metrics Team – This team is responsible for keeping statistics on the performance of the teams on the project. Depending on the organization's data collection procedures, some typical statistics kept might be the number of maintenance requests generated, number of maintenance requests serviced, number of lines of code written, number of hours performed on each task, and values produced by the tool on each new version of the system. This team will interact with the requirements, design, implementation, testing and integration, and maintenance teams, providing assessments of quality and efficiency, as well as feedback to these sub–teams.

Documentation Team – This team is responsible for the project documentation. This includes external documentation of the requirements, design, source code, and other supporting documents.

System Administration Team – This team is responsible for ensuring that the underlying computer hardware, software, and network support are working as needed by the project team. This team often includes the network administration team.

Reuse and Reengineering Team – This team is responsible for selection and use of appropriate existing reusable software components. Reengineering may be necessary if the software project depends upon some old code that must be changed because of new advances in technology.

It is not surprising that there are several managerial tasks involved here, one for each sub–team. Of course, if the teams are small enough, a manager may be one of the technical members of the team (or even the only team member).

❑ **Check Your Progress – 5 :**

1. _____ is responsible for developing the overall management plan for the project.

   a. System Administration Team b. Metrics Team

   c. Planning Team d. System Analysis Team

2. _____ is responsible for keeping statistics on the performance of the teams on the project.

   a. Metrics Team b. QA Team

   c. System Administration Team d. Training Team

3. Write a detail notes on sub–team that are required in software engineering.

2.8 Software Management Activity :

Software project management includes numeral of activities, which includes planning, scope and cost estimation of software product. Software management activities include following :

- Project Preparation
- Scope Management
- Project Estimation
- **Project Preparation :** Project preparation is activity, which is completed before actual software development starts. It is there for software development but it is not containing concrete movement that has direct connection with the software development; somewhat it is a set of multiple process, which enables development.

- **Scope Management :** Scope management contains all the activities, process required to perform in order to make a deliverable software. Scope management is important because it makes boundaries of the project by clearly defining what would be done and what would not be done in the project.

  During Scope management, it is essential to –

  - ✓ State the possibility
  - ✓ Decide its confirmation and control
  - ✓ Divide the project into smaller parts for ease of management.
  - ✓ Validate the scope
  - ✓ Control the scope by including changes to the scope

- **Project Estimation :** There must be accurate estimation of various measures for an effective management. With the exact estimation, managers can manage and control the project more professionally and successfully.

  Project estimation involves the following :

- **Software Size Estimation :** Size of software estimated either by calculating number of functions in the software or by KLOC (Kilo Line of Code). Functions differ according to the requirement of the user.

- **Effort Estimation :** It refers to employee requirement and man–hour required to produce the software. Software size should be known for effort estimation. This is done by manager's experience, and old data of organization.

- **Time Estimation :** It refers to estimation of the time required to develop the software. Required efforts is separated into sub categories as per the customer requirements and interdependency of the software components. Software development is divided into smaller responsibilities, activities or events by Work Breakthrough Structure (WBS). The responsibilities are planned on day–to–day basis. The sum of time required to complete all responsibilities in hours or days is the total time invested to complete the project.

- **Cost Estimation :** It is the most difficult of all because it depends on more elements than any of the previous ones. Following criteria is going to consider in cost estimation :

  - ✓ Size of the software
  - ✓ Quality of the Software
  - ✓ Hardware
  - ✓ Additional software or tools, licenses etc.
  - ✓ Skilled employees with task–specific skills
  - ✓ Travel involved
  - ✓ Communication
  - ✓ Training and support

❑ **Check Your Progress – 6 :**

1. Explain software management activity.

2.9 Project Preparation :

Project preparation refers to roadmap of all activities to be done according to order and within time slot allotted to each activity.

Project managers define various responsibilities, and project milestones and then arrange them. They look for tasks which is serious in path in the plan, which are essential to complete in specific way and strictly within the time due.

It is necessary to identify below points in scheduling a project :

✓ Break down the project tasks into smaller, manageable form

✓ Find out various tasks and correlate them

✓ Estimate time required for each task

✓ Divide time into work–units

✓ Allocate acceptable number of work–units for each task

✓ Calculate total time required for the project from start to finish

❑ **Check Your Progress – 7 :**

1. Write a short note on Project Scheduling.

2.10 Resource Organization :

All elements used in software development is assumed as resource for that project. As a resource there may be human, tools, and software libraries. The available resources are in limited quantity and stay in the organization as a pool of assets.

The shortage of resources hampers development of the project and it can pause behind the schedule. Assigning additional resources increases development cost in the end. It is therefore necessary to estimate and assign acceptable resources for the project.

Resource management includes –

- Defining organization project by creating a project team and assigning tasks to each team member
- Defining resources required at a particular stage and their availability

- Manage Resources by producing resource request when they are required and de–allocating them when they are no more needed.

❑ **Check Your Progress – 8 :**

1. Write a short note on Resource Management.

2.11 Risk Organization :

Risk management is a strategy to deal with hazard and risk, designed to ensure that priorities are made, action taken and available resources put to maximum effect.

- **Initial Threat Appraisal (Risk Analysis) :**

The first stage of a risk strategy is to identify the key issues. An Initial Threat Appraisal provides a systematic examination of defined risks which could threaten the operations of a business and in addition, identify those areas where further operational analysis may be required.

Working in partnership with our client is a fundamental aspect of the process and the parameters of the project are agreed at the outset with the client agreeing the areas of risk for investigation.

Threats which may be either insurable or un–insurable may then identified and evaluated using the widely accepted components of Likelihood, Severity and Control, and a prioritized action plan is produced.

- **Risk Strategy :**

Information collected during the initial investigations is used to formulate a plan designed to reduce risks to a minimum. A program will be organized over a realistic time frame, usually 2 to 3 years, and will include allocation of responsibilities, upgrading of policies and procedures where appropriate, and on–going risk evaluation and management.

Training is a vital component of a Risk Strategy assisting with an enhanced "risk" culture, and acceptance of responsibility.

- **Benefits :**

The major features and associated benefits are :

✓ Identification of the major threats to a business within prescribed areas of risk.

✓ Relative assessment of threats in terms of the combination of likelihood, severity and control.

✓ Benchmarking where studies are undertaken at multiple locations.

✓ Development of a prioritized action plan with broad costing indications.

✓ Guidance on the development of an appropriate risk strategy.

✓ Assistance with related Corporate Governance.

✓ Measurement of the effectiveness of existing management controls.

- ✓ Development of existing risk management initiatives through a program of education and training.
- ✓ Control of the strategic process and over a period a reduction in the "total cost of risk".
- ✓ Minimum cost for insurance and risk transfer.

❑ **Check Your Progress – 9 :**

1. Write a note on risk management in software management.

2. What is the first step of Risk Analysis ?
   a. Analysis the risk impact
   b. Finding solution to minimize risk
   c. Identifying the key issues that could threaten the business operations
   d. None of these

2.12 Project Implementation and Monitoring :

In this phase, described tasks of the project plans are implemented as per the time–table. Implementation requires monitoring to check all is working as per the plan. Monitoring is detecting chance of risk and taking measures to address the risk or report the status of numerous tasks.

These measures include –

- **Monitoring Activity –** All events planned within some responsibilities can be monitored on day–to–day basis. When all events in a task are completed, it is considered as complete.
- **Status Reports –** The reports contain status of events and tasks completed as per given time, generally a week. Status can be marked as finished, pending or work–in–progress etc.
- **Milestones Checklist –** Every project is divided into multiple stages where main tasks are completed (milestones) based on the phases of SDLC. This milestone checklist is prepared once every few weeks and reports the status of milestones.

❑ **Check Your Progress – 10 :**

1. Write a short note on Project execution and monitoring.

2.13 Project Communication Organization :

Effective communication plays important role in the success of a project. It ties gaps between client and the organization, among the team members as well as other stake holders in the project such as hardware suppliers.

Communication can be oral or written. Communication management process include the following steps :

- **Planning –** It refers to the identifications of all the investors in the project and the method of communication among them. It also reflects if any extra communication facilities are essential.
- **Sharing –** It refers to manager focuses on sharing correct information to the correct person at the correct time. This keeps everyone involved in the project up–to–date with project progress and its status.
- **Feedback –** Project managers use feedback and create status and performance reports. This ensures that input from various investors is coming to the project manager as their feedback.
- **Closure –** At the end of each major event, end of a phase of SDLC or end of the project itself, administrative closure is formally announced to update every investor by sending email, by issuing a hardcopy of document or by other mean of effective communication.

❑ **Check Your Progress – 11 :**

1. Explain Project Communication Management.

2.14 Configuration Organization :

It refers to a process of tracking and controlling the modification in the software related to the requirements, design, functions and development of the product.

IEEE defines it as "the process of finding and defining the objects in the system, controlling the change of these items throughout their life cycle, recording and reporting the status of items and change requests, and verifying the completeness and correctness of items".

Normally, once the SRS is confirmed there is less chance of requirement of changes from user. If they occur, the changes are addressed only with prior approval of higher management, as there is a possibility of cost and time overrun.

- **Baseline :**

A stage of SDLC is supposed over if it baselined, i.e., baseline is a measurement that defines completeness of a stage. When all activities relating to it are finished and well documented a stage is baselined. If it was not the final phase, its output would be used in next immediate phase.

- **Change Control :**

It is referred to as a function of configuration management, which ensures that all modification made to software system are reliable and complete as per organizational rules and regulations.

A change in the configuration of product goes through following steps –

- **Identification –** A modification request arrives from internal or external source. When change request is recognized formally, it is correctly documented.
- **Validation –** Validity of the modification request is checked and its management procedure is confirmed.
- **Analysis –** The effect of modification request is analyzed in terms of schedule, cost and required efforts. Overall effect of the probable modification on the system is analyzed.

❑ **Check Your Progress – 12 :**

1. Write a detail note on Configuration Management.

2.15 Concept of Tailoring :

Tailoring in Software development is the process of extracting a set of processes, tasks and artifacts from the organizations established processes, tasks and artifacts so as to best suit a project to achieve its objectives.

- **Practices and Guidelines :**

It is very important that we have a set of tailoring guidelines that will guide the development teams to let them decide what is best for the projects. The guidelines must also specify is that what is tailorable and what is mandatory.

For example, in software development project if a project manager says that the project need not maintain a project management plan then it should be not acceptable as per the tailoring guidelines. So basically, there will be some processes, tasks and artifacts that will have to be developed and maintained in a software product development project.

So, these guidelines must take into account multiple aspects before being released as part of the organization quality management system. These aspects must look into the current state of process implementation, customer's needs and objectives, project characteristics etc.

Organizations that are offerings services for different types of projects like Full life cycle development, maintenance, QA, Technical support must also look into coming up with life cycle models for each of these type of software projects. These life cycle models must go hand in hand with the Process tailoring guidelines established at the organization level.

❑ **Check Your Progress – 13 :**

1. What is meant by Tailoring ?
   a. Tool for quality assurance
   b. Software development tool
   c. Process of extracting a set of processes, tasks and artifacts from the organizations established processes, tasks and artifacts so as to best suit a project to achieve its objectives.
   d. None of these

2.16 Extreme Programming :

Extreme Programming teams use a simple form of planning and tracking to decide what should be done next and to predict when the project will be done. Focused on business value, the team produces the software in a series of small fully–integrated releases that pass all the tests the Customer has defined.

Extreme Programming is about team responsibility for all code, for coding in a consistent pattern so that everyone can read everyone's code, about keeping the system running and integrated all the time.

There are four basic activities that XP proposes for software development process :

1. **Coding :** In XP coding is considered the only important product of the system development process. XP programmers start to generate codes at the very beginning so "At the end of the day, there has to be a program."
2. **Testing :** XP emphasizes to always check if a function works is by testing it. XP uses Unit Tests which are automated tests, and the programmer will write tests as many as possible to try to break the break the code he or she is writing.
3. **Listening :** Obviously, coding and testing need to be done no matter how a system is developed, but listening is very important in XP. For XP developers the ability and expertise in technical aspects should be accompanied by the ability to be good listeners. This ability will enable them to understand what customers want and develop solutions which match customers ? needs and desires as close as possible.
4. **Designing :** XP's simplicity principle doesn't mean that it can exclude designing process. Without proper design in the long run system becomes too complex and projects could come to a halt. It is then important to create a design structure that organizes the logic in the system so too many dependencies in the system can be avoided.

❑ **Check Your Progress – 14 :**

1. What is extreme programming ?
   a. Coding of software
   b. Designing of software
   c. Coding, Testing, Listening and Designing of software
   d. None of these

2.17 Let Us Sum Up :

In this unit we have learnt that how software project management is carried on as well as need of software project management such as software project manager and sub–team needed to perform various operation at different stages.

We also discussed software project management activity, project scheduling, resource management, and risk management which plays an important role in each stage. It is also noted that in each stage of software project management is monitor by project management so that project can fulfill the user requirements.

2.18 Answers for Check Your Progress :

- **Check Your Progress 1 :**

  1. (b) 2. (Refer 2.2)

- **Check Your Progress 2 :**

  1. (b)

- **Check Your Progress 3 :**

  1. (d)

- **Check Your Progress 4 :**

  1. (a) 2. (Refer 2.6)

- **Check Your Progress 5 :**

  1. (c) 2. (a) 3. (Refer 2.7)

- **Check Your Progress 6 :**

  1. (Refer 2.8)

- **Check Your Progress 7 :**

  1. (Refer 2.9)

- **Check Your Progress 8 :**

  1. (Refer 2.10)

- **Check Your Progress 9 :**

  1. (Refer 2.11) 2. (c)

- **Check Your Progress 10 :**

  1. (Refer 2.12)

- **Check Your Progress 11 :**

  1. (Refer 2.13)

- **Check Your Progress 12 :**

  1. (Refer 2.14)

- **Check Your Progress 13 :**

  1. (c)

- **Check Your Progress 14 :**

  1. (c)

2.19 Glossary :

1. **SPM –** Software project management understand, plan, measure and control of project as delivering on time and budget problems of contiguous and chained allocation.
2. **Extreme Programming –** The team responsibility for codes or coding in consistent pattern which keeps system running and integrated at all times.
3. **Risk Management –** It is a strategy to deal with hazard and risk, designed to ensure that priorities are made, action taken and available resources put to maximum effect.

2.20 Assignment :

1. Explain Software Project management in detail.

2.21 Activities :

1. Study activities of all sub–team of software project management.

2.22 Case Study :

Create project scheduling for Hospital Management System.

2.23 Further Reading :

1. W. Shewhart, Statistical Method from the Viewpoint of Quality Control, Dover, 1986.

Unit 03

SOFTWARE PROJECT PLANNING TOOLS AND TECHNIQUES

UNIT STRUCTURE

3.0 Learning Objectives

3.1 Introduction

3.2 Work Breaks Down Structure (WBS)

3.3 Software Sizing

3.4 Milestones Baseline

3.5 Cost Estimation

3.6 Ray Leigh Curve

3.7 LOC and FP Estimation

3.8 Documenting the Schedule

3.9 Developing the Activity Network

3.10 Empirical Relationships

3.11 Effort Distribution

3.12 Empirical Estimation Techniques – COCOMO

3.13 Let Us Sum Up

3.14 Answers for Check Your Progress

3.15 Glossary

3.16 Assignment

3.17 Activities

3.18 Case Study

3.19 Further Readings

3.0 Learning Objectives :

After learning this unit, you will be able to understand:

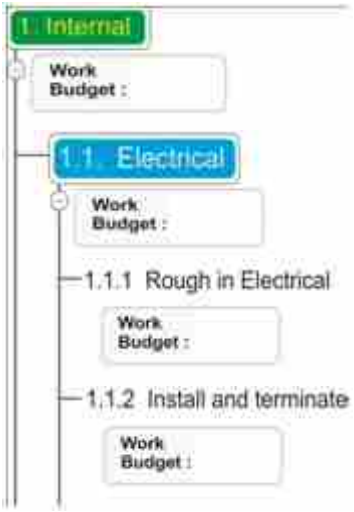
- Work breakdown structure
- Software sizing
- Cost estimation model
- COCOMO Model
- Software Documentation

3.1 Introduction :

Software project planning actually encompasses all estimation, risk analysis, scheduling, and SQA/SCM planning. However, in the context of set of resources, planning includes estimate – your effort to determine how much money, effort, resources, and time it will take to develop a specific software–based system or product.

3.2 Work Breaks Down Structure (WBS) :

Work Breakdown Structure is a tool project managers use to break projects down into manageable pieces. The Work Breakdown Structure is an essential tool to set the project scope. It forms the agreement between you and your client on what is included and what is not included in your end deliverable. Creating a Work Breakdown Structure requires a substantial amount of energy, time and people, but in the end is not rocket science. The Project Management Body of Knowledge defines the work breakdown structure as a "deliverable oriented hierarchical decomposition of the work to be executed by the project team." The work breakdown structure visually defines the scope into manageable chunks that a project team can understand, as each level of the work breakdown structure provides further definition and detail.



Work Breakdown Structure

A work breakdown structure starts with the project as the top–level deliverable and is further decomposed into sub–deliverables as shown in above figure.

The project team creates the project work breakdown structure by identifying the major functional deliverables and subdividing those deliverables into smaller systems and sub–deliverables. The following guidelines should be considered when creating a work breakdown structure:

- ✓ The top level represents the final deliverable or project
- ✓ Sub–deliverables contain work packages that are assigned to an organization's department or unit
- ✓ All elements of the work breakdown structure don't need to be defined to the same level
- ✓ The work package defines the work, duration, and costs for the tasks required to produce the sub–deliverable
- ✓ Work packages should not exceed 10 days of duration
- ✓ Work packages should be independent of other work packages in the work breakdown structure
- ✓ Work packages are unique and should not be duplicated across the work breakdown structure

❑ **Check Your Progress – 1 :**

1. What is Work Breakdown Structure ?

   a. Division of software development process

   b. Tool used for breaking the whole project into manageable pieces

   c. Task allocation

   d. None of these

3.3 Software Sizing :

Sizing measures are needed to make valid comparisons across (or within) systems. Without a software sizing measure, productivity cannot be computed. There are only two software sizing measures widely used today — Lines of Code (LOC or KLOC) and Function Points (FP). Though each is a sizing measure, they actually measure different things and have very different characteristics.

Lines of Code is a measure of the size of the system after it is built. It is very dependent on the technology used to build the system, the system design, and how the programs are coded. The major disadvantages of LOC are that systems coded in different languages cannot be easily compared and efficient code is penalized by having a smaller size. Capers Jones stated at a talk to the Chicago Quality Assurance Association on November 22, 1996 that anyone using LOC is "committing profession malpractice." Despite these problems, LOC is still frequently used by very reputable and professional organizations.

In contrast to LOC, Function Points is a measure of delivered functionality that is relatively independent of the technology used to develop the system. FP is based on sizing the system by counting external components (inputs, outputs, external interfaces, files and inquiries.) While FP addresses many of the problems inherent in LOC and has developed a loyal following, it has its own set of advantages and disadvantages.

❑ **Check Your Progress – 2 :**

1. What is the software sizing measures ?

   a. Function points (FP) b. Lines of code (LOC)

   c. Both a & b d. None of these

3.4 Milestones Baseline :

Baseline is an agreed description of the attributes of a product, at a point in time, which serves as a basis for defining change. A baseline may be a single work product, or set of work products that can be used as a logical basis for comparison.

The main goal of baseline is to reduce the software project's vulnerability. It is generally used in configuration management. While milestone is a point which shows how far the project is progressed?

❑ **Check Your Progress – 3 :**

1. Baseline is used in _____.

   a. Project Planning b. Requirement Gathering

   c. Configuration Management d. Quality Management

3.5 Cost Estimation :

Software costing should be carried out objectively with the aim of accurately predicting the cost of developing the software. If the project cost has been computed as part of a project bid to a customer, a decision then has to be made about the price quoted to the customer. Classically, price is simply cost–plus profit. However, the relationship between the project cost and the price to the customer is not usually so simple.

Software pricing must take into account broader organizational, economic, political and business considerations. Therefore, there may not be a simple relationship between the price to the customer for the software and the development costs. Because of the organizational considerations involved, project pricing should involve senior management, as well as software project managers.

Three main approaches to estimation :

- **Empirical**
- **Heuristic**
- **Analytical**

- **Empirical techniques :** It is a technique based on past experience.
- **Heuristic techniques :** This technique is assumed to be the characteristics that can be estimated to express mathematical expression.
- **Analytical techniques :** This derives the required results starting from certain simple assumptions.

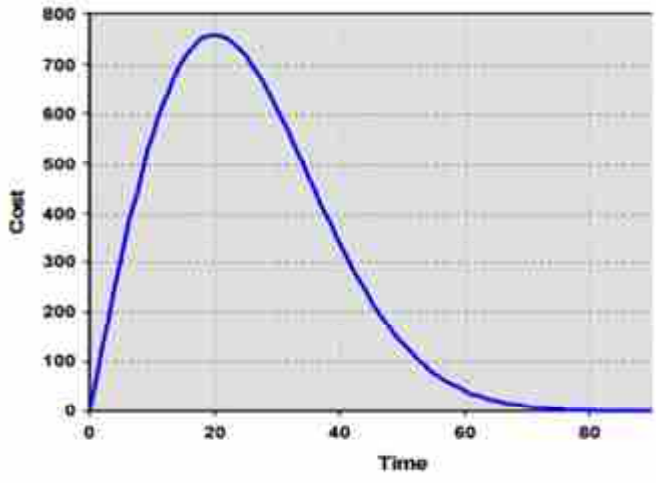
❑ **Check Your Progress – 4 :**

1. Which of the following cost estimation approach is based on past experience ?

   a. Heuristic b. Empirical c. Analytical d. None of these

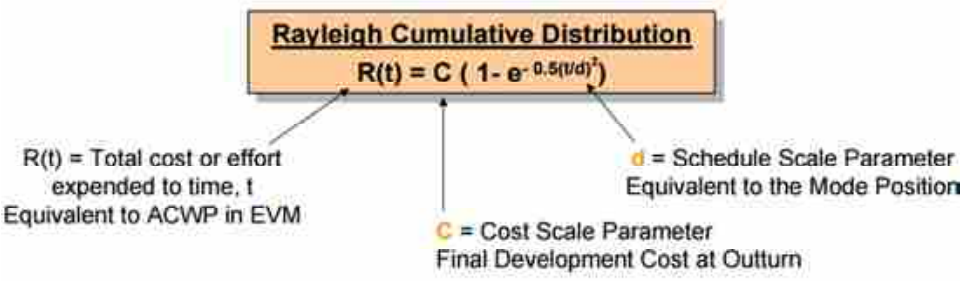
3.6 Ray Leigh Curve :

The classic S–shape shown in below figure can be describe as cost histories of items and shows certain development programs aspects.



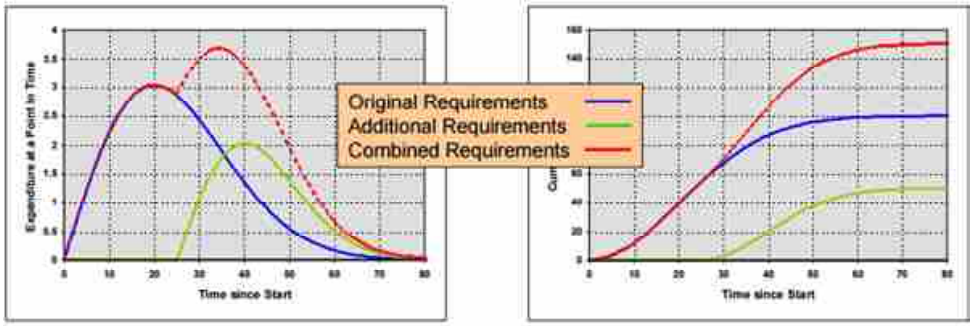
Curve

The Rayleigh Distribution models show development program's effort which ramp–up the peak value and over certain time frame.



Distribution Formula

A two Rayleigh curves when combined describes the major project requirements or variants that comes all through the development. If additional requirements are obtained before peak, then it is possible to model the total expenditure by using one Rayleigh Curve as shown in below figure.



Two Curves Showing Combination

❑ **Check Your Progress – 5 :**

1. Why do we use Ray Leigh Curve ?
   a. Development program's effort which rump–up the peak value and over certain time frame
   b. To measure cost
   c. To measure risk
   d. None of these

3.7 LOC and FP Estimation :

- **LOC Estimation :**

LOC is the Line of Code, which is program length used as predictor of program features like effort and ease of maintenance. The LOC measure is used to measure size of the software. There are many versions of LOC such as :

DSI which is Delivered Source Instructions used in COCOMO'81 as KDSI and explains as :

✓ Source lines which deliver portion of product like test drivers and support software's that are obtained by project staff by creating applications generators,

✓ Has single instructions in one line of code or card image,

✓ In this the instructions are declared with no Comments.

- **Advantages of LOC :**

✓ Simple to measure

- **Disadvantage of LOC :**

✓ As defined on code, it is hard to measure size of specification with it.

✓ It is limited to particular size, namely length and has no functionality or complexity.

✓ It has great line of code due to worst software design.

✓ It depends on certain language.

✓ Hard for users to understand.

- **FP Estimation :**

FP which is Function Points is basic data from which the productivity metrics could be solved. Such type of data is used in two ways :

✓ In form of an estimation variable used to size every element of software,

✓ In form of baseline metrics gathered from past projects used in conjunction with estimation variables to get cost and effort projections.

It is noted that the idea of this is to find and count number of different functions like :

✓ External inputs which are normally in shape of file names

✓ External outputs that can be reports, messages etc.

✓ Queries which can be interactive in response

✓ External files or interfaces which are files that are shared with software systems

✓ Internal files that are not seen outside system

It is noted that such above functions individually assessed for complexity and weight age which varies from simple to complex internal files.

- **Advantages of FP :**

✓ It is not specific to code

✓ It has no dependency on language

✓ In the data is seen early in project as detailed specification.

✓ It is more accurate

- **Drawbacks of FP :**

✓ It requires subjective counting

✓ It is very difficult to automate and hard to calculate

✓ It does not considered quality of output

✓ It is more towards traditional data processing applications

✓ Effort prediction by unadjusted function is worse with addition of TCF

❑ **Check Your Progress – 6 :**

1. What are the advantages of using Function Point (FP) ?

   a. It is not specific to code

   b. It is used to measure the size of software

   c. It depends on certain language

   d. None of these

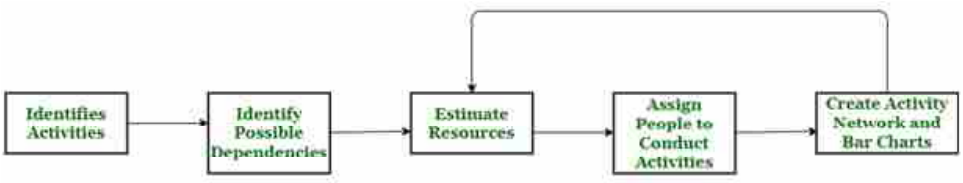
3.8 Documenting the Schedule :

Documentation of software includes written text that joins computer software. It explains how it operates or how to use it, and may mean different things to individuals in different roles. Documentation is an important part of software engineering. Types of documentation include:

- **Requirements** – Statements that identify attributes capabilities, characteristics, or qualities of a system. This is the foundation for what will be or has been implemented.
- **Architecture/Design** – Overview of software. Includes relations to an environment and construction principles to be used in design of software components.
- **Technical** – Documentation of code, algorithms, interfaces, and APIs.
- **End user** – Manuals for the end–user, system administrators and support staff.
- **Marketing** – How to market the product and analysis of the market demand.

We see that Project Scheduling includes :

- ✓ Split project into tasks and estimate time and resources required to complete each task.
- ✓ Organize tasks concurrently to make optimal use of workforce.
- ✓ Minimize task dependencies to avoid delays caused by one task waiting for another to complete.
- ✓ Dependent on project managers intuition and experience.



Project Scheduling Process

Scheduling for certain projects can be analyzed based on :

- ✓ End–date for release of computer–based system
- ✓ Releasing of effort by software organization in particular time frame
- ✓ Based on end–date which is normally set by software engineering organization

Problems involved in Scheduling :

- ✓ In analyzing difficulty of problems requires heavy cost.
- ✓ In this, the productivity does not remain in proportion as compared to people involved on particular work.
- ✓ In this, the project gets delayed, if people get added in between the project.
- ✓ It doesn't allow contingency in planning.

❑ **Check Your Progress – 7 :**

1. What is software documentation ?

   a. Text that accompanies computer software

   b. Requirement document

   c. Document of project process schedule

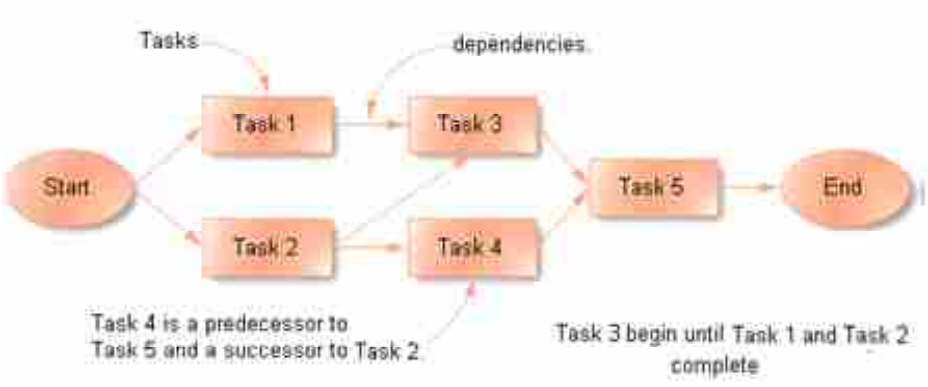
   d. None of these

3.9 Developing the Activity Network :

Activity Network is a graphical way to analyze tasks, dependencies and the critical path in a project. In this, nodes show tasks and dependencies by way of lines which connects particular job those boxes.

It is seen that an Activity Network Diagram which can be represented as arrow diagram is particularly applied to find time sequences of events that appears to be the pivotal to objectives. Such will help the team members to analyze specific event sequences which are driven by time requirements for objective achievement. These are useful in case when a project carries multiple activities that are required for simultaneous management working. Such types of network diagram are mostly applied in engineering and construction project management. It is seen that the Critical Path Analysis draws on particular methodology will find and standardize certain management activities.

Activity Network Diagram normally helps to identify many efficient sequences of events which are required to complete any project. It allows to create a realistic project schedule by showing graphically the total time required to complete project sequence in which the tasks carry out at particular time.



Activity Network Diagram

In above figure, in the Activity Network diagram, it is shown that interdependencies exist among tasks by way of boxes and arrows. In this, the arrows will point in task box which appears from its earlier tasks that must be completed before the tasks finally start. It is seen in the above figure that the arrows pointing out of a task box will enter into its previous tasks that cannot start till the work gets finished off.

❑ **Check Your Progress – 8 :**

1. What is Activity Network ?
   a. Network of various activities used in software development
   b. Graphical way to analyze tasks, dependencies and the critical path in a project
   c. Both of these
   d. None of these

3.10 Empirical Relationships :

In software, empirical relational system translates as relations of formal relational system that explains three relations :

- **Inequalities :** In this, partial orders on sets of entities gets translate into inequalities that exist among measures.
- **Equalities :** It explains about equivalence relations which get translated in equalities.
- **Special Value Assignment :** It is the measure of size of empty program body which is corrected to value zero.

In software, we see that one should start from measurement goal and explains empirical relational system which exists for external attribute of interest. It is easy to find sensible measures of external attributes that involves measure of cost and time or reliability and defects.

❑ **Check Your Progress – 9 :**

1. What is meant by Equalities in empirical relationship ?
   a. It is the measure of size of empty program body which is corrected to value zero.
   b. Partial orders on set of entities gets translate into inequalities that exist among measures.
   c. It explains about equivalence relations which get translated in equalities.
   d. None of these

3.11 Effort Distribution :

Software plays important role in systems acquisition and development which involves large and complex systems. Such type of systems involves accurate estimates of costs which act as important part of program management. It is noted that the bulk of software development cost exits because of human effort and estimation methods required.

Effort distribution is an exercise which is performed in software development projects in order to produce cost estimation report which is normally required to get the percentage of every phase. Such type of percentage of effort distribution was obtained by collection of empirical data in case of large and small developments.

Effort distribution in software comprises of following phases which finds percentage of effort required :

- Requirements gathering

- Analysis
- Development
- Testing

We see that effort estimation accuracy is normally applied without any adjustments which made for differences that focuses on scope and quality as examined while estimating the effort and system working. To improve effort estimation, a special terminology for software effort estimation is required that carries certain guidelines :

1. You should not mix estimation of effort with planning, budgeting or pricing
2. In case of assessing estimation accuracy, you have to make sure that the estimate and real effort results in comparison.

❑ **Check Your Progress – 10 :**

1. What is Effort Distribution ?
   a. Efforts put in the software development
   b. Division of work among them members
   c. It is an exercise used to produce cost estimation report which required to get the percentage of every phase
   d. None of these

3.12 Empirical Estimation Techniques – COCOMO :

It is noted that with Software cost estimation, we can predict the amount of effort that is needed to create a software system. Software cost estimation results in continuous activity that starts initially with software life cycle and continues all through the lift time. We see that there are many ways in which you can estimate the software cost by following Boehm's classification system that results in:

- Expert Judgment
- Algorithmic Estimation
- Analogy Based Estimation

COCOMO which is Constructive Cost Models is a form of an algorithmic method which depends on mathematical models which generates cost estimate which appears as function of number of variables considered as main cost factors.

COCOMO models is shown by Barry Boehm in year 1981 that uses equations and parameters obtained from his early practice and research estimation of projects. It has code–size S which is given in thousand LOC which results in person month. With his invention, Barry Boehm suggested following COCOMO models:

Simple COCOMO :

This is his initial model that has formula

Effort = a × (K LOC)b

In this :

- S is code–size
- a, b are complexity factors

Such type of model uses three sets of a, b depending on complexity of software. It is noted that basic COCOMO model is simple and easy since it does not involve many cost factors and can only be applied to do rough estimation.

Intermediate COCOMO :

This model is called as nominal effort estimation model which is obtained with power function with a, b along with coefficient a different from basic COCOMO.

❑ **Check Your Progress – 11 :**

1. Which is also called as Nominal effort estimation model ?
   a. Basic COCOMO Model
   b. Intermediate COCOMO Model
   c. Detailed COCOMO Model
   d. None of these

3.13 Let Us Sum Up :

In this unit we have learnt that Work Breakdown Structure is a tool which is used by project managers in order to break projects into standard manageable pieces. It is noted that sizing measures are required in order to have valid comparisons across systems.

It is seen that software costing be done with an objective of accurately predicting cost of development of software. Software documentation is a text which accompanies computer software and operates different things to people in different roles

Activity Network Diagram is an arrow diagram which is mostly applied to find time sequences of events which appears to be objectives. COCOMO is Constructive Cost Models which is an algorithmic method that depends on mathematical models which generates cost estimate that appears as function of number of variables which serves as main cost factors.

3.14 Answers for Check Your Progress :

❑ **Check Your Progress 1 :**

1. (b)

❑ **Check Your Progress 2 :**

1. (c)

❑ **Check Your Progress 3 :**

1. (c)

❑ **Check Your Progress 4 :**

1. (b)

❑ **Check Your Progress 5 :**

1. (a)

- **Check Your Progress 6 :**

  1. (a)

- **Check Your Progress 7 :**

  1. (a)

- **Check Your Progress 8 :**

  1. (b)

- **Check Your Progress 9 :**

  2. (c)

- **Check Your Progress 10 :**

  1. (c)

- **Check Your Progress 11 :**

  1. (b)

3.15 Glossary :

1. **Activity Network –** Activity Network is a graphical way to analyze tasks, dependencies and the critical path in a project.
2. **COCOMO Model –** COCOMO (Constructive Cost Models) is a form of an algorithmic method which depends on mathematical models which generates cost estimation.
3. **Software Documentation –** Documentation of software include written text that joins computer software. It explains how it operates or how to use it ? or may mean different things to people in different roles.

3.16 Assignment :

1. Write a short note on COCOMO model.

3.17 Activities :

1. Study Empirical relationship in detail.

3.18 Case Study :

Do you think software documentation is required in the Software development ? Justify your answer with the help of suitable example.

3.19 Further Reading :

1. W. Shewhart, Statistical Method from the Viewpoint of Quality Control, Dover, 1986.
2. W.E. Deming, Out of the Crisis, SPC Press, 1982; reprinted in paperback by MIT Press, 2003.
3. T. Gilb, Software Metrics, Little, Brown, and Co., 1976.
4. R. Zultner, "The Deming Approach to Quality Software Engineering," Quality Progress, vol. 21, no. 11, 1988, pp. 58–64.
5. W.H. Dana, The X–15 Lessons Learned, tech. report, NASA Dryden Research Facility, 1993
6. L. Aiken and S. West: Multiple Regression: Testing and Interpreting Interactions, Sage Publications, 1991.

Unit 04

SOFTWARE PROJECT MAINTENANCE

UNIT STRUCTURE

4.0 Learning Objectives

4.1 Introduction

4.2 Software Configuration Management

4.2.1 Why do we need Configuration Management ?

4.2.2 Element of Configuration Management

4.2.3 Participant of Software Configuration Management

4.3 Software Maintenance Processes

4.4 Project Planning

4.5 Documentation Standards

4.6 Version Control

4.7 Let Us Sum Up

4.8 Answers for Check Your Progress

4.9 Glossary

4.10 Assignment

4.11 Activities

4.12 Case Study

4.13 Further Readings

4.0 Learning Objectives :

After learning this unit, you will be able to understand :

- Software configuration management
- Software maintenance process
- Project planning
- Documentation standard
- Version control

4.1 Introduction :

Software maintenance is the change of software after delivery to correct faults, to improve performance or other attributes, or to adjust the software to a changed environment. It stands for all the modifications done after the delivery of software. There is numeral of reasons, why changes are required; some of them are as follows:

Market Conditions – Rules, which changes over the time like taxation and newly announced restrictions like, how to maintain accounting, may initiation need for modification.

Client Requirements – Over the time, customer may ask for new features or functions in the software.

Host Modifications – If any of the hardware and/or platform (such as operating system) of the target host changes, software changes are required to keep flexibility.

Organization Changes – If there is any business level change at client end, such as reduction of organization strength, getting another company, organization offering into new business, need to change in the original software may arise.

4.2 Software Configuration Management :

Configuration management is a discipline which will find out coordinate, approves or disapprove and implement changes that appears in object that are mainly applied to construct and maintain software systems.

Configuration management is an integral part of software development as it results in special consideration in product line which can be seen as multidimensional version of configuration management problem for a special system. Configuration management for product lines is complex for single systems as it has qualities like parallel development, distributed engineering, release management, change management, configuration and workspace management that carry tools, processes and environments.

4.2.1 Why do we need Configuration Management ?

The primary reasons for Implementing Technical Software Configuration Management System are:

- ✓ There are multiple people working on software which is continually updating
- ✓ It may be a case where multiple versions, branches, authors are involved in a software config project, and the team is geographically distributed and works concurrently
- ✓ Changes in user requirement, policy, budget, and schedule need to be accommodated.
- ✓ Software should able to run on various machines and Operating Systems
- ✓ Helps to develop coordination among stakeholders
- ✓ SCM process is also beneficial to control the costs involved in making changes to a system

Any change in the software configuration Items will affect the final product. Therefore, changes to configuration items need to be controlled and managed.

4.2.2 Elements of Configuration Management :

It is noted that Configuration Management System carries following elements :

Component Elements : It results as tools which are coupled inside file management system that gives access to and management of each SCI.

Process Elements : It shows collection of procedures and tasks which explains effective approach to change management for all stakeholders.

Construction Elements : It is a set of tolls which automates construction of software which ensures set of assembled components.

Human Elements : It is applied by team as it carries set of tools and process features which encompass other elements.

Software Configuration Management deals with different technical difficulties of project plan. It is found that in software organization, good implementation of software configuration management improves productivity with increase of coordination among which occur among team members. With use of Software Configuration Management, the confusion which exists due to miscommunication among members gets wiped out. Such type of management system control basic elements like :

- Software objects
- Program code
- Test data
- Test output
- Design documents
- User manuals

4.3.3 Participant of Software Configuration Management :

Following are the key participants in SCM

Participant of Software Configuration Management

1. Configuration Manager :

✓ Configuration Manager is the head who is Responsible for identifying configuration items.

✓ CM ensures team follows the SCM process.

✓ He / She needs to approve or reject change requests.

2. Developer :

✓ The developer needs to change the code as per standard development activities or change requests. He is responsible for maintaining configuration of code.

✓ The developer should check the changes and resolves conflicts.

3. Auditor :

- ✓ The auditor is responsible for SCM audits and reviews.
- ✓ Need to ensure the consistency and completeness of release.

4. Project Manager :

- ✓ Ensure that the product is developed within a certain time frame.
- ✓ Monitors the progress of development and recognizes issues in the SCM process.
- ✓ Generate reports about the status of the software system.
- ✓ Make sure that processes and policies are followed for creating, changing, and testing.

5. User :

- ✓ The end user should understand the key SCM terms to ensure he has the latest version of the software.

Software Configuration Management System has many advantages :

- It helps in lowering of redundant work.
- It helps in reducing problems related to configuration.
- It helps in building team coordination.
- It helps in managing tools that are required in building configuration.
- It makes sure that all bugs get traced out from its source.

❑ **Check Your Progress – 1 :**

1. What are the elements of Configuration Management System ?

a. Component Element b. Process Element

c. Human Element d. All of Above

4.3 Software Maintenance Processes :

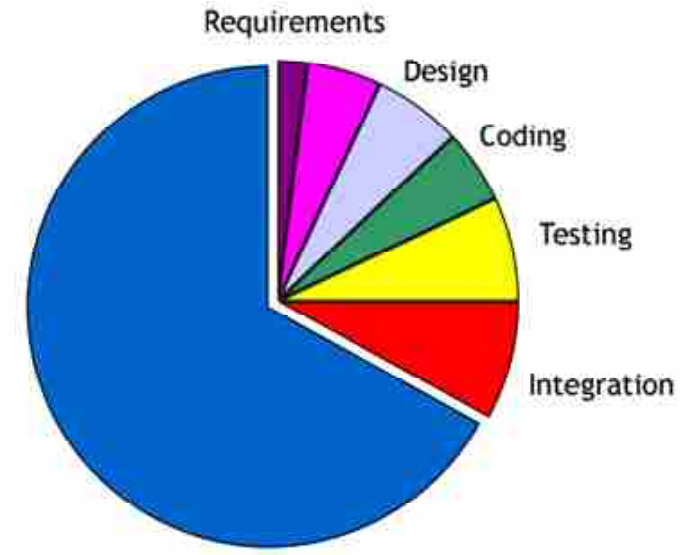
Software maintenance is normally required in order to support many key features and product lines which are applied in daily operational cycles. With Software maintenance, normally bugs get fixed which are reactive to errors and omissions.

The software maintenance processes comprise of :

- ✓ Implementation of process that carries software preparation and transition activities related to conception and creation of maintenance plan
- ✓ Problem and modification analysis process which execute when application serves as responsibility of maintenance group.
- ✓ Process acceptance of modification which confirm modified work with individual who submit request to ensure modification of solution.
- ✓ Exceptional migration process features which is not part of daily maintenance.
- ✓ In case when an event is not present daily this serves as retirement of software.

The steps involved in Software project maintenance includes :

- Requirements
- Design
- Coding
- Testing
- Integrating



Steps in Software Project Maintenance

Software Maintenance Categories

- **Corrective :** Corrective maintenance aims to correct any remaining errors regardless of where they may cause specifications, design, coding, testing, and documentation, etc.
- **Adaptive :** It refers to changes and updates applied to keep the software up–to date and changed to the ever–changing world of technology and business environment.
- **Perfective :** It refers to changes and updates done in order to keep the software usable over long period of time. It includes new features, new user requirements for purifying the software and improve its reliability and performance. Modifications to improve the performance, changeability, or efficiency of the system.
- **Preventive :** It refers to changes and updates to prevent future problems of the software. It aims to join problems, which are not important at this moment but may cause serious issues in future.

❑ **Check Your Progress – 2 :**

1. What are the main categories of software project management ?

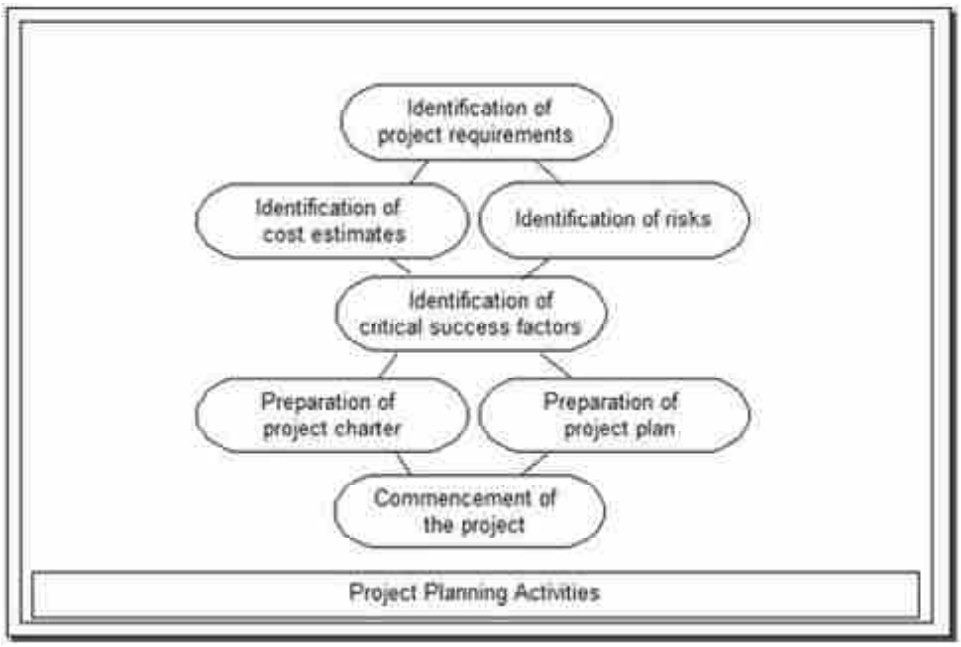
   a. Corrective b. Adaptive c. Perfective d. All of Above

4.4 Project Planning :

The project planning process involves a set of interrelated activities followed in an orderly manner to implement user requirements in software and includes the description of a series of project planning activities and individual(s) responsible for performing these activities. In addition, the project planning process comprises the following.

- ✓ Objectives and scope of the project
- ✓ Techniques used to perform project planning
- ✓ Effort (in time) of individuals involved in project
- ✓ Project schedule and milestones
- ✓ Resources required for the project
- ✓ Risks associated with the project.

Project planning process comprises several activities, which are essential for carrying out a project systematically. These activities refer to the series of tasks performed over a period of time for developing the software. These activities include estimation of time, effort, and resources required and risks associated with the project.



Project Planning Activities

Project planning process consists of the following activities.

- **Identification of Project Requirements :** Before starting a project, it is essential to identify the project requirements as identification of project requirements helps in performing the activities in a systematic manner. These requirements comprise information such as project scope, data and functionality required in the software, and roles of the project management team members.
- **Identification of Cost Estimates :** Along with the estimation of effort and time, it is necessary to estimate the cost that is to be incurred on a project. The cost estimation includes the cost of hardware, network connections, and the cost required for the maintenance of hardware components. In addition, cost is estimated for the individuals involved in the project.
- **Identification of Risks :** Risks are unexpected events that have an adverse effect on the project. Software project involves several risks (like technical risks and business risks) that affect the project schedule and increase the cost of the project. Identifying risks before a project begins helps in understanding their probable extent of impact on the project.

- **Identification of Critical Success Ractors :** For making a project successful, critical success factors are followed. These factors refer to the conditions that ensure greater chances of success of a project. Generally, these factors include support from management, appropriate budget, appropriate schedule, and skilled software engineers.
- **Preparation of Project Charter :** A project charter provides a brief description of the project scope, quality, time, cost, and resource constraints as described during project planning. It is prepared by the management for approval from the sponsor of the project.
- **Preparation of Project Plan :** A project plan provides information about the resources that are available for the project, individuals involved in the project, and the schedule according to which the project is to be carried out.
- **Commencement of the Project :** Once the project planning is complete and resources are assigned to team members, the software project commences.

Once the project objectives and business objectives are determined, the project end date is fixed. The project management team prepares the project plan and schedule according to the end date of the project. After analyzing the project plan, the project manager communicates the project plan and end date to the senior management. The progress of the project is reported to the management from time to time. Similarly, when the project is complete, senior management is informed about it. In case of delay in completing the project, the project plan is re–analyzed and corrective actions are taken to complete the project. The project is tracked regularly and when the project plan is modified, the senior management is informed.

❑ **Check Your Progress – 3 :**

1. What is the first step in software project planning ?
   a. Identification of Project Requirement
   b. Identification of Cost Estimate
   c. Identification of Risk
   d. None of Above

4.5 Documentation Standards :

Documentation standards in a software project are important because documents are the only tangible way of representing the software and the software process. Standardized documents have a consistent appearance, structure and quality, and should therefore be easier to read and understand.

There are three types of documentation standards:

Documentation Process Standards : These standards define the process that should be followed for document production.

Document Standards : These standards govern the structure and presentation of documents.

Document Interchange Standards : These standards ensure that all electronic copies of documents are compatible.

Documentation process standards define the process used to produce documents (example here). This means that you set out the procedures involved

in document development and the software tools used for document production. You should also define checking and refinement procedures to ensure that high–quality documents are produced.

Document process quality standards must be flexible and able to cope with all types of documents. For working papers or electronic memos, there is no need for explicit quality checking. However, for formal documents, that is, those that will be used for further development or released to customers, you should use a formal quality process.

Document standards should apply to all documents produced during a software development project. Documents should have a consistent style and appearance, and documents of the same type should have a consistent structure. Although document standards should be adapted to the needs of a specific project, it is good practice for the same "house style? to be used in all of the documents produced by an organization.

❑ **Check Your Progress – 4 :**

1. How many types of documentation standards are there ?

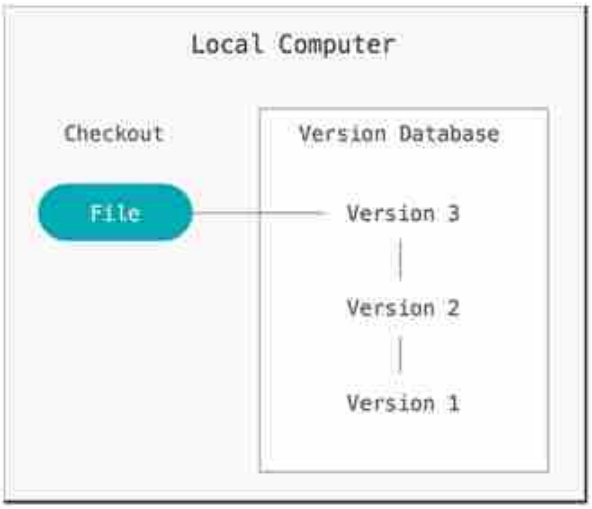
   a. One b. Four c. Three d. Five

4.6 Version Control :

Version control is a system that records changes to a file or set of files over time so that you can recall specific versions later. For the examples in this book, you will use software source code as the files being version controlled, though in reality you can do this with nearly any type of file on a computer.

Version control systems are essential for any form of distributed, collaborative development. Whether it is the history of a wiki page or large software development project, the ability to track each change as it was made, and to reverse changes, when necessary, can make all the difference between a well–managed and controlled process and an uncontrolled "first come, first served ? system. It can also serve as a mechanism for due diligence for software projects.

Many people's version–control method of choice is to copy files into another directory (perhaps a time–stamped directory, if they're clever). This approach is very common because it is so simple, but it is also incredibly error prone. It is easy to forget which directory you're in and accidentally write to the wrong file or copy over files you don't mean to.



Version Control

Developers may wish to compare today's version of some software with yesterday's version or last year's version. Since version control systems keep track of every version of the software, this becomes a straightforward task. Knowing the what, who, and when of changes will help with comparing the performance of particular versions, working out when bugs were introduced (or fixed), and so on. Any problems that arose from a change can then be followed up by an examination of who made the change and the reasons they gave for making the change.

❑ **Check Your Progress – 5 :**

1. What is version control used ?

   a. It keeps track of every version of the software

   b. It helps in comparing updates

   c. It is helpful in recording changes

   d. All of Above

4.7 Let Us Sum Up :

In this unit we have learnt that software maintenance results in alteration of software product which deliver correct faults in order improve performance or attributes. It is found that configuration management will find coordinate, approves or disapprove and implement changes which appears in object which is mainly applied to construct and maintain software systems.

Software maintenance is required which supports certain features and product lines which are applied in daily operational cycles. Project planning is a calculation on process of project completion in required timeframe that certainly defines stages and required resources. Version control is a system which will highlight changes to file or set of files over time in order to collect specific versions.

4.8 Answers for Check Your Progress :

❑ **Check Your Progress 1 :**

1. (d)

❑ **Check Your Progress 2 :**

1. (d)

❑ **Check Your Progress 3 :**

1. (a)

❑ **Check Your Progress 4 :**

1. (c)

❑ **Check Your Progress 5 :**

1. (d)

4.9 Glossary :

1. **Software Configuration Management** – Software Configuration management is a discipline which will implement changes that appears in object that is mainly applied to construct and maintain software.

2. **Version Control –** Version control is a system that records changes to a file or set of files over time so that you can recall specific versions later.
3. **Project Planning –** It involves calculation that deals in full process of project completion in definite timeframe which are normally with certain defined stages, and designated resources.

4.10 Assignment :

1. Write short note on software configuration management.

4.11 Activities :

1. Read more about version control.

4.12 Case Study :

Do you think Documentation standards help in improving software quality, Comment ?

4.13 Further Reading :

1. W. Shewhart, Statistical Method from the Viewpoint of Quality Control, Dover, 1986.
2. W.E. Deming, Out of the Crisis, SPC Press, 1982; reprinted in paperback by MIT Press, 2003.
3. T. Gilb, Software Metrics, Little, Brown, and Co., 1976.
4. R. Zultner, "The Deming Approach to Quality Software Engineering," Quality Progress, vol. 21, no. 11, 1988, pp. 58–64.
5. W.H. Dana, The X–15 Lessons Learned, tech. report, NASA Dryden Research Facility, 1993

BLOCK SUMMARY :

In this block, you have learnt and understand about the basic of Project management and Extreme Programming techniques. The block gives an idea on the study about how to manage Software Development process with concept on Software Sizing and Empirical Relationships. You have been well explained on the concepts of Project Planning and basic documentation standards.

The block detailed about the basic of Tailoring techniques with involvement in terms of software modeling. The concept related to Ray Leigh Curve and its advantages and measures on software projects are well explained to you. You will be demonstrated practically about Empirical Estimation Techniques related to COCOMO technique.

BLOCK ASSIGNMENT :

❖ **Short Questions :**

1. Explain Prototyping Model.
2. Explain Spiral Model.
3. Explain Project Communication Management.
4. What is Project Execution and Monitoring ?
5. What is Software Development ?
6. Explain version control ?
7. Explain Software Configuration Management.
8. What is extreme programming ?
9. What is software documentation ?
10. Explain Software Maintenance Process.

❖ **Long Questions :**

1. Explain Software Development Life Cycle.
2. Write a note on Sub–Team needed in Software Engineering Projects.
3. Write difference between waterfall model and incremental model.
4. Explain in detail COCOMO Model and its advantage.

❖ **Enrolment No. :** ☐

1. How many hours did you need for studying the units ?

Unit No.	1	2	3	4
No. of Hrs.				

2. Please give your reactions to the following items based on your reading of the block :

Items	Excellent	Very Good	Good	Poor	Give specific example if any
Presentation Quality	☐	☐	☐	☐	_____
Language and Style	☐	☐	☐	☐	_____
Illustration used (Diagram, tables etc)	☐	☐	☐	☐	_____
Conceptual Clarity	☐	☐	☐	☐	_____
Check your progress Quest	☐	☐	☐	☐	_____
Feed back to CYP Question	☐	☐	☐	☐	_____

3. Any other Comments

.....

.....

.....

.....

.....

.....

.....

.....

Dr. Babasaheb Ambedkar Open University Ahmedabad

BCAR-402

Software Engineering

BLOCK 2 : SOFTWARE REQUIREMENT, DESIGN, QUALITY MANAGEMENT & SOFTWARE TESTING

UNIT 5 SOFTWARE REQUIREMENT

UNIT 6 SOFTWARE DESIGN

UNIT 7 SOFTWARE QUALITY MANAGEMENT

UNIT 8 SOFTWARE TESTING TECHNIQUES

SOFTWARE REQUIREMENT, DESIGN, QUALITY MANAGEMENT & SOFTWARE TESTING

Block Introduction :

In this block, we will detail about the basic of software requirement. Software requirement play an important role in software development life cycle. To gather requirement analyst will going to interact with number user of all levels. Analyst will apply different type of requirement initiation techniques to find requirements.

In this block, we will detail about the basic of software design, strategies, software user interface design, design complexity and software implementation which include all criteria of design from basic to advance as design play an important role.

In this block, we will detail about the basic of software quality and idea about different variables associated with it. The block will focus on the study and concept of software testing and associated process. You will give an idea on software quality control along with necessity of quality in software project.

User acceptance testing is final phase of software testing process where normally software users perform testing of software so as to handle the required work as per specifications. In software development testing the software and quality management are important aspects. Performance testing is a non-functional technique which is done in order to locate the system parameters which could be related to responsiveness and stability under different workload.

After studying this block, you will make to learn and understand about the basic of testing of software techniques along with detailed explanation on Capability Maturity Model. The concept related to Review and Walkthrough with respect to software configuration management are well detailed. You will be demonstrated about User acceptance testing and its standards.

Block Objectives :

After learning this block, you will be able to understand :

- Requirement Engineering and it's process
- Requirement initiation process and techniques
- Characteristics of software requirement
- Basic of Software Design
- Strategies of Software Design
- Software user interface Design
- Complexity of Software Design & Software Implementation
- Understanding about Software Quality needs
- Basic of CMM modeling
- Features of software testing

Block Structure :

Unit 5 : Software Requirement

Unit 6 : Software Design

Unit 7 : Software Quality Management

Unit 8 : Software Testing Techniques

Unit

05 SOFTWARE REQUIREMENT

UNIT STRUCTURE

5.0 Learning Objectives

5.1 Introduction

5.2 Requirement Engineering

5.3 Requirement Engineering Process

- **5.3.1 Feasibility Study**
- **5.3.2 Requirement Gathering**
- **5.3.3 Software Requirement Specification (SRS)**
- **5.3.4 Software Requirement Validation**

5.4 Requirement Initiation Process

5.5 Requirement Initiation Techniques

- **5.5.1 Interview**
- **5.5.2 Questionnaires**
- **5.5.3 Observation**
- **5.5.4 Document Review**

5.6 Software Requirement Characteristics

5.7 Software Requirements

- **5.7.1 Functional Requirements**
- **5.7.2 Non–Functional Requirements**

5.8 User Interface Requirements

5.9 Software System Analyst

- **5.9.1 Role of System Analyst OR What a System Analyst Does ?**
- **5.9.2 Attributes of a Good System Analyst OR Qualities of System Analyst**

5.10 Let Us Sum Up

5.11 Answers for Check Your Progress

5.12 Glossary

5.13 Assignment

5.14 Activities

5.15 Case Study

5.16 Further Readings

5.0 Learning Objectives :

After learning this unit, you will be able to understand :

- Introduction about the Requirement
- Idea about Requirement Engineering and it's Process

- Detail idea about Software Requirement Specification
- Detail of Software Requirement Validation
- Requirement Initiation Process and Techniques
- Characteristics of Software Requirement
- Idea about Software System Analyst

5.1 Introduction :

The software requirements are explanation of required functionalities of the aim system. Requirements convey the opportunities of users from the software product. The requirements can be clear or hidden, known or unknown, expected or unexpected from client's point of view.

5.2 Requirement Engineering :

Requirements Engineering (RE) refers to the development of defining, documenting, and maintaining requirements in the engineering process. It provides the proper mechanism to understand what the customer needs, analyzing the need, and evaluating possibility, negotiating a practical solution, specifying the solution clearly, confirming the specifications and managing the requirements as they are changed into a working system. Therefore, requirement engineering is the self–controlled application of confirmed principles, methods, tools, and notation to define a planned system's proposed performance and its related constraints.

Requirement modelling involves investigation and fact–finding to describe the current system and define the requirements for the new system such as **Outputs, Inputs, Processes, Performance, and Security**.

- **Output** refers to the electronic or printed information produced by the system.
- **Input** refers to the necessary data that enters in the system, either manually or in an automated manner.
- **Process** refers to the logical rules that are applied to transform the data into meaningful information.
- **Performance** refers to system characteristics such as speed, volume, capacity, availability, reliability.
- **Security** refers to hardware, software, and procedural controls that safeguard and protect the system and its data from internal or external threats.

❑ Check Your Progress – 1 :

1. Requirement engineering refers to process of _____.
   a. Defining Requirement b. Documenting Requirement
   c. Maintaining Requirement d. All of Above
2. _____ refers to printed information produced by the system.
   a. Input b. Security c. Output d. Process

5.3 Requirement Engineering Process :

It is a process of four–step process which is as follows :

- **Feasibility Study**
- **Requirement Gathering**
- **Software Requirement Specification**
- **Software Requirement Validation**

5.3.1 Feasibility Study

A system request must meet several tests to see whether it is meaningful to proceed further. This series of tests is called a feasibility study and it is a vital part of every system project.

Feasibility study uses four major standards to measure, or predicts a system's successes which are as follows : ***Operational Feasibility, Technical Feasibility, Economic Feasibility, and Schedule Feasibility.***

- **Operational Feasibility :**

A system that has operational feasibility is one that will be used effectively after it has been developed. If users have difficulty with a new system, it will not produce the expected benefits. Operational feasibility depends on several vital issues. For example, consider the following questions :

- ✓ Will the new system result in a workforce reduction ? If so, what will happen to affected employees ?
- ✓ Will the new system require training for users ? If so, is the company prepared to provide the necessary resources for training current employees ?
- ✓ Will users be involved in planning the new system right from the start ?
- ✓ Will customer experience adverse (poor) effects in any way, either temporarily or permanently ? Will any risk to the company's image or goodwill result ?
- ✓ Do any legal or ethical issues need to be considered ?

- **Technical Feasibility :**

A system request has technical feasibility if the organization has the resources to develop or purchase, install, and operate the system. When accessing technical feasibility, an analyst must consider the following points :

- ✓ Does the company have the necessary hardware, software, and network resources ? If not, can those resources be acquired without difficulty ?
- ✓ Does the company have the needed technical expertise ? If not, can it be acquired ?
- ✓ Does the proposed platform have enough capacity for future needs ? If not, can it be expanded ?
- ✓ Will the hardware and software environment be reliable ? Will it integrate with other company information systems, both now and in future ? Will it interface properly with external system operated by customers and suppliers ?
- ✓ Will the combination of hardware and software supply enough performance ?

- ✓ Will the system be able to handle future transaction volume and company growth ?

- **Economic Feasibility :**

A system request has economic feasibility if the projected benefits of the proposed system balance the estimated costs involved in acquiring, installing, and operating it. Costs can be one time or continuing, and can acquire at various times during project development and use. When measuring costs, companies usually consider the total cost of ownership (TCO), which includes ongoing support and maintenance cost, as well as acquisition costs.

To determine TCO, the analyst needs to estimate costs in each of the following areas :

- ✓ People, including IT staff and users
- ✓ Hardware and equipment
- ✓ Software, including in–house development as well as purchases from vendors
- ✓ Formal and informal training
- ✓ Licenses and fees
- ✓ Consulting expenses
- ✓ Facility costs
- ✓ The estimated cost of not developing the system or postponing the project

In addition to costs, you need to access tangible and intangible benefits to the company.

1. Tangible Benefits :

Tangible benefits are benefits that can be measured in dollars. Tangible benefits result from a decrease in expenses, an increase in revenues, or both.

Examples of tangible benefits include the following :

- ✓ A new scheduling system that reduces overtime.
- ✓ An online package tracking system that improves service and decreases the need for clerical staff.
- ✓ A sophisticated inventory control system that cuts excess inventory and eliminates production delays.

2. Intangible Benefits :

Intangible benefits are difficult to measure in dollars but also should be identified.

Examples of intangible benefits include the following :

- ✓ A user–friendly system that improves employee job satisfaction.
- ✓ A sales tracking system that supplies better information for marketing decisions.
- ✓ A new website that enhances the company's image.

- **Schedule Feasibility :**

Schedule Feasibility is defined as the probability of a project to be completed within its scheduled time limits, by a planned due date. When accessing schedule feasibility, a system analyst must consider the interaction

between time & cost. For example, speeding up a project schedule might make a project feasible, but much more expensive.

Other issues that relate to schedule feasibility include the following :

- ✓ Has management established a certain time–table for the project ?
- ✓ Will a project manager be appointed ?

5.3.2 Requirement Gathering :

If the feasibility report is positive to task the project, next phase starts with gathering requirements from the user. Analysts communicate with the client and end–users to know their ideas on what the software should deliver and which features they need to include in the software.

5.3.3 Software Requirement Specification (SRS) :

After the requirements are collected from various end–users, system analyst creates a document called SRS. It describes how the proposed software will interact with hardware, external interfaces, speed of operation, response time of system, portability of software across various platforms, maintainability, speed of recovery after crashing, Security, Quality, Limitations etc.

*The requirements received from client are written in normal language. It is the duty of the system analyst to document the requirements in technical language so that they can be known and used by the software development team.

There should be following features in SRS :

- ✓ Requirements of user are stated in normal language.
- ✓ Technical requirements are expressed in structured language.
- ✓ Design explanation should be written in Pseudo code.
- ✓ Format of Forms and GUI screen prints.
- ✓ Conditional and mathematical notations for DFDs etc.

5.3.4 Software Requirement Validation :

The stated requirement in the document should be validated after the requirement specification and developed. User might ask for illegal, unrealistic solution or experts may understand the requirements inaccurately. This results in huge increase cost.

Requirements can be checked against following circumstances :

- ✓ If they can be practically executed
- ✓ If they are valid and as per functionality of software
- ✓ If there are any doubts
- ✓ If they are complete
- ✓ If they can be verified

❑ Check Your Progress – 2 :

1. Requirement engineering process include _____.

   a. Feasibility Study b. SRS

   c. Requirement Gathering d. All of Above

2. SRS Stands for _____.
   a. Software Requirement System
   b. Software Requirement Specification
   c. Software Requirement Structure
   d. None of Above
3. Write a detailed note on Feasibility Study.

5.4 Requirement Initiation Process :

Requirement initiation process can be represented using the below diagram :

- **Requirements Gathering :** The developers discuss with the client and end users and know their hopes from the software.
- **Organizing Requirements :** The developers arrange the requirements in order of importance, urgency and accessibility.
- **Negotiation & Discussion :** If requirements are unclear or there are some conflicts in requirements of various investors, it is then negotiated and conversed with the investors. Requirements may then be arranged and practically compromised.
- **Documentation :** All formal and informal, functional and non–functional requirements are documented and made available for next phase.

❑ **Check Your Progress – 3 :**

1. Write a short note on Requirement Initiation Process.

5.5 Requirement Initiation Techniques :

Requirements Initiation is the process finding the requirements for proposed software by communicating with client, end users, system users, and others who have a stake in the software system development.

There are various ways which are as follows :

5.5.1 Interview :

An interview is a planned meeting during which you obtain information from another person. You must have the skills needed to plan, conduct, and

document interviews successfully. The interviewing process consists of these seven steps :

1. Determine the people to interview.
2. Establish objectives for the interview.
3. Develop interview questions.
4. Prepare for the interview.
5. Conduct the interview.
6. Document the interview.
7. Evaluate the interview.

1. Determine the people to interview :

To get an accurate picture of the system, you must select the right people to interview and ask them the right questions. During the system analysis phase, you might need to interview people from all levels of the organization.

Group interviews can save time and provide an opportunity to observe interaction between the participants. Group interviews also can present problems. One person might control the conversation, even when questions are addressed specifically to others. Organization level also can present a problem, as the presence of upper management in an interview can prevent lower–level employees from expressing themselves openly.

2. Establish objectives for the interview :

After deciding on the people to interview, you must establish objectives for the session. First you should determine the general areas to be discussed, and then list the facts you want to gather.

You also try to request ideas, suggestions, and opinions during the interview. The objectives of an interview depend on the role of the person being interviewed. Upper–level manager can provide the picture and help you to understand the system. Specific details about operations and business processes are best learned from people who actually work with the system on a daily basis.

3. Develop interview questions :

Creating a standard list of interview questions helps keep you on track and avoid unnecessary discussion. If you interview several people who perform the same job, a standard question list allows you to compare their answer to the same questions.

The interview should consist of several different kinds of questions : **open–ended questions** and **closed ended questions.**

o Open–Ended Questions :

– *"A question that requires the respondent to express viewpoint is called an open–ended interview".*

– Open–ended questions are useful when you want to understand a large process or draw out the interviewee's opinions, attitudes, or suggestions.

o Closed–Ended Questions :

– *"A question that requires a direct answer to a question is called closed–ended interview".*

– Closed–ended questions limit or restrict the response. You use closed–ended questions when you want information that is more specific or when you need to verify facts.

4. Prepare for the interview :

After setting the objectives and developing the questions, you must prepare for the interview. Careful preparation is essential because this is an important meeting and not just a casual chat. Schedule a specific day and time for meeting and place a reminder call to confirm the meeting.

5. Conduct the interview :

After determining the people to interview, setting your objectives, and preparing the questions, you should develop a specific plan for the meeting. When conducting the interview, you should begin by introducing yourself, describing the project, and explaining your interview objectives.

During the interview, ask questions in the order in which you prepared them, and give the interviewee sufficient time to provide thoughtful answers. Your primary responsibility during an interview is to listen carefully to the answers. Analysts sometimes hear only what they expect to hear. You must concentrate on what is said and notice any nonverbal communication that take place. This process is called **engaged listening.**

6. Document the interview :

You should write down a few notes to run your memory after the interview, you should avoid writing everything that is said. Too much writing distracts the other person and makes it harder to establish a good relationship.

After conducting the interview, you must record the information quickly. You should set away time right after the meeting to records the facts and evaluate the information. For that reason, try not to schedule back–to–back interviews.

7. Evaluate the interview :

In addition to recording the facts obtained in an interview, try to identify any possible partiality. Some interviewees might answer your questions in an attempt to be helpful even though they do not have the necessary experience to provide accurate information.

The system analyst will evaluate to summaries the information gathered during the interview and verify it with the user to get accurate and complete information.

5.5.2 Questionnaires :

In projects where it is desirable to obtain input from a large number of people, a questionnaire can be a valuable tool. A questionnaire, also called survey, is a document containing a number of standard questions that can be sent to many individuals.

Questionnaires are used to obtain information about workloads, reports received volumes of transactions handled, types of job duties, difficulties, and opinions of how the job could be performed better or more efficiently. A typical questionnaire starts with heading, which includes a title, a brief statement of purpose, the name and telephone number of the contact person, the deadline date for completion, and how and where to return the form.

Some additional ideas to keep in mind when designing your questionnaires :

- ✓ Keep the questionnaire brief and user friendly.
- ✓ Arrange the questions in a logical order.
- ✓ Use simple terms and wording.
- ✓ Limit the use of open–ended questions.
- ✓ Include a section at the end of the questionnaire for general comments.
- ✓ Test the questionnaire whenever possible in a small test group before finalizing it and distributing to a large group.

5.5.3 Observation :

The observation of current operating procedure is another technique. Seeing the system in action gives you additional viewpoint and a better understanding of system procedures. Personal observation also allows you to verify statements made in interviews and determine whether procedure really operate as they are described.

Through observation, you might discover that neither the system documentation nor the interview statements are accurate. Plan your observations in advance by preparing a check list of specific tasks you want to observe and questions you want to ask.

Consider the following issues when you prepared your list :

- ✓ Ask sufficient questions to ensure that you have complete understanding of the present system operation.
- ✓ Observe all the steps in a transaction and note the documents, inputs, outputs, and processes involved.
- ✓ Examine each form, record, and report.
- ✓ Talk to the people who receive current reports to see whether the reports are complete, timely, accurate, and in a useful form.

5.5.4 Document Review :

Document review can help you understand how the current system is supposed to work. Remember that system documentation is sometimes out of date. So, you should obtain copies of actual forms and operating documents currently in use. You also should review blank copies of forms, as well as sample of actual completed forms. You usually can obtain document samples during interviews with the people who perform the procedure. If the system uses a software package, you should review the documentation for that software.

❑ Check Your Progress – 5 :

1. A question that requires the respondent to express viewpoint is called _____.

   a. Questionnaire b. Open–Ended Question
   c. Interview d. Closed–Ended Question

2. _____ is used to obtain information from large number of people.

   a. Interview b. Document Review
   c. Observation d. Questionnaire

3. Write a detailed note on Interview Technique.

5.6 Software Requirement Characteristics :

Gathering software requirements is the basis of the whole software development project. Hence, they must be clear, accurate, and well–defined.

A complete Software Requirement Specifications must be :

- Clear
- Accurate
- Reliable
- Understandable
- Changeable
- Provable
- Ordered
- Traceable
- Dependable source

❑ **Check Your Progress – 5 :**

1. A complete software requirement specification must be _____.

   a. Clear b. Accurate c. Reliable d. All of Above

5.7 Software Requirements :

We should try to know what type of requirements may arise in the requirement initiation phase and what kinds of requirement are expected from the software system. Software requirements categorized in two categories which are as follows :

5.7.1 Functional Requirements

Requirements, that is related to functional part of software fall into this category. They describe functions and functionality within and from the software system.

Examples :

- ✓ Search option given to user to search from various invoices.
- ✓ User should be able to mail report to management.
- ✓ Users can be divided into groups and given separate rights.
- ✓ It should fulfill business rules and organizational functions.
- ✓ Software is developed keeping downward compatibility whole.

5.7.2 Non–Functional Requirements :

Requirements that are not related to functional part of software fall into this category. They are understood or predictable characteristics of software, which users make possibility of.

Non–functional requirements include –

- ✓ Accessibility
- ✓ Flexibility
- ✓ Performance
- ✓ Interoperability
- ✓ Storage
- ✓ Configuration
- ✓ Security
- ✓ Logging
- ✓ Cost
- ✓ Disaster recovery

❑ **Check Your Progress – 6 :**

1. Write a note on Software Requirement.

5.8 User Interface Requirements :

User Interface (UI) is a vital part of any software or hardware or hybrid system. Software is generally known if it is –

- ✓ Easy to use
- ✓ Faster response
- ✓ Successfully handling operational errors
- ✓ Providing simple yet reliable user interface

User acceptance depends upon how user uses the software. UI is the only way for users to observe the system. Well performing software must be prepared with attractive, clear, reliable, and responsive user interface. Else, the functionalities of software cannot be used in appropriate way. A system is said to be good if it provides means to use it efficiently.

User interface requirements are stated as follows :

- ✓ Content presentation
- ✓ Easy Navigation
- ✓ Simple interface
- ✓ Reactive
- ✓ Reliable UI elements
- ✓ Feedback mechanism
- ✓ Default settings
- ✓ Focused layout
- ✓ Strategical use of color and texture

- ✓ Deliver help information
- ✓ User centric approach
- ✓ Group based view settings

❑ **Check Your Progress – 7 :**

1. Write a note on User Interface Requirement.

5.9 Software System Analyst :

"A system analyst is a person who conducts a study, identify activities and determine the procedure to achieve the objective."

A system analyst researches problem, plans solutions, recommends software and systems, and manages development to meet business or other requirements. System analysts are link between vendors and information technology. They may be responsible for developing cost analysis, design considerations, and implementation time–lines.

5.9.1 Role of system analyst OR What a system analyst does ?

1. **Define Requirements :** The first and may be most difficult task of system analyst is problem definition. Business problems are quite difficult to define. It is also true that problems cannot be solved until they are correctly and clearly defined.
2. **Gathering data, facts, and opinion of user :** Initially system analyst does not know how to solve a specific problem. He must consult with managers, users and other data processing professional in defining problems and developing solutions. We use various methods for data gathering to get the correct solution of a problem. Also, he is responsible to define priorities according to requirements.
3. **System Analysis :** The analyst's sole responsibility is conducting systems studies to know relevant facts about business activity. Having gathered the data relating to a problem, the systems analyst then thinks of plan to solve it. He may not come up personally with best way of solving a problem but pulls together solution is achieved.
4. **Solving Problems :** System analyst coordinates the process of developing solutions. Since many problems have numbers of solutions, the system analyst must evaluate the advantage of such suggested solutions before recommending one to the management.
5. **Drawing up specification :** Systems analysts are often referred to a planner. A key part of the system analyst is to develop a plan to meet the management's aims. The specification must be accurate and detailed so that it can be used by system implementers. The specification must be non–technical so that users and managers can understand it.

6. **Designing System :** When the plan has been accepted, systems analyst is responsible for designing system so that management's goal could be achieved. System design is a time consuming, complex and precise work.
7. **Evaluation System :** System analyst often co–ordinates the testing procedures and helps in deciding whether or not the new system is meeting standards established in the planning phase.

5.9.2 Attributes of a good system analyst OR Qualities of system analyst :

System analyst must have the following attributes :

1. **Knowledge of Business functions :** A system analyst must know the environments in which he or she works.

   He must understand the management structure and the relationship between departments in organizations.

   A working knowledge of accounting, marketing and material management principles is a must.

   Since so many systems are built around these areas. He must familiar with his company's product and services and management's policies in areas concerning him.
2. **Knowledge of people :** Since system analyst work with others so closely, he or she must understand their needs and what motivates them to develop system properly.
3. **Knowledge of data processing principles/Computer system :** Most systems today are computer based. The systems analyst must fully aware about the potential and limitations of computers.
4. **Ability to Communicate :** As a coordinator, a system analyst must communicate properly with people of different levels within an organization. He should use non–technical language while communicating with other. System analyst must listen carefully to what others said and include the thoughts of others into the system development process.
5. **Flexibility :** System analysts must be flexible in their thinking since they often do not get their own way. Different sections of organization have conflicting needs and most systems are result of compromise. The analyst goal is to produce the system that will be the best for his organization. This required an open mind, and flexibilities in his ideas.
6. **An analytical mind :** System analyst often finds themselves with more data than they required. It requires an analytical mind to select pertinent data and concentrate on them in defining problems and forming solutions.
7. **Well educated with sharp mind :** System analysts have to work with people of all levels in every phase of business. They must know how to work with all of them and gain their confidence. Analyst must have sharp mind to learn quickly how people do their job and develop ways for them to it better.

❑ **Check Your Progress – 8 :**

1. What is System Analyst ? Discuss Role and Quality of System Analyst.

5.10 Let Us Sum Up :

In this unit we have learnt that during the software development how requirements are going to gather from the all level of the organization. As well feasibility study, requirement gathering, software requirement specification and validation criteria are going to be carry on in requirement engineering process.

We have also seen about different requirement initiation techniques using which requirements are going to collect. So, as a part of requirement initiation techniques interview, questionnaire, observation, and document review are conducted by the analyst.

Analyst is person who conducts a study, identify activities and determine the procedure to achieve the objective. So, there are various role that system analyst has to perform to meet the requirement and there should be analytical qualities in the analyst.

5.11 Answers for Check Your Progress :

❑ **Check Your Progress 1 :**

1. (d) 2. (c)

❑ **Check Your Progress 2 :**

1. (d) 2. (b) 3. (Refer 5.3.1)

❑ **Check Your Progress 3 :**

1. (Refer 5.4)

❑ **Check Your Progress 4 :**

1. (b) 2. (d) 3. (Refer 5.5.1)

❑ **Check Your Progress 5 :**

1. (d)

❑ **Check Your Progress 6 :**

1. (Refer 5.7)

❑ **Check Your Progress 7 :**

1. (Refer 5.8)

❑ **Check Your Progress 8 :**

1. (Refer 5.9)

5.12 Glossary :

1. **Requirement Engineering –** Requirement engineering (RE) refers to the development of defining, documenting, and maintaining requirements in the engineering process.

2. **Feasibility Study –** A system request must meet several tests to see whether it is meaningful to proceed further. This series of tests is called a feasibility study.
3. **SRS –** After the requirements are collected from various end–user, system analyst creates a document called SRS.
4. **Interview –** An interview is a planned meeting during which you obtain information from another person.
5. **Open–Ended Question –** A question that requires the respondent to express viewpoint is called an open–ended interview.
6. **Closed–Ended Question –** A question that requires a direct answer to a question is called closed–ended interview.
7. **Questionnaire –** A questionnaire, also called survey, is a document containing a number of standard questions that can be sent to many individuals.
8. **System Analyst –** A system analyst is a person who conducts a study, identify activities and determine the procedure to achieve the objective.

5.13 Assignment :

1. Explain all Requirement Initiation Techniques.

5.14 Activities :

1. Differentiate Functional Requirement and Non–Functional Requirement.

5.15 Case Study :

Justify, Interview requirement initiation technique is more flexible than the Questionnaire.

5.16 Further Reading :

1. Software Engineering : A Practitioner's Approach Book by Roger S. Pressman

Unit 06

SOFTWARE DESIGN

UNIT STRUCTURE

6.0 Learning Objectives
6.1 Introduction
6.2 Software Design Basic
- 6.2.1 Software Design Level
- 6.2.2 Modularization
- 6.2.3 Concurrency
- 6.2.4 Coupling and Cohesion
- 6.2.5 Design Verification

6.3 Software Design Strategies
- 6.3.1 Structured Design
- 6.3.2 Function Oriented Design
- 6.3.3 Object Oriented Design
- 6.3.4 Software Design Approaches

6.4 Software User Interface Design
- 6.4.1 Command Line Interface (CLI)
- 6.4.2 Graphical User Interface
- 6.4.3 User Interface Design Activities
- 6.4.4 GUI Implementation Tools
- 6.4.5 User Interface Golden Rules

6.5 Software Design Complexity
- 6.5.1 Halsted's Complexity Measures
- 6.5.2 Cyclomatic Complexity Measures
- 6.5.3 Function Point

6.6 Software Implementation
- 6.6.1 Structured Programming
- 6.6.2 Functional Programming
- 6.6.3 Programming Style
- 6.6.4 Software Documentation
- 6.6.5 Software Implementation Challenges

6.7 Let Us Sum Up
6.8 Answers for Check Your Progress
6.9 Glossary
6.10 Assignment
6.11 Activities
6.12 Case Study
6.13 Further Readings

6.0 Learning Objectives :

After learning this unit, you will be able to understand :

- Introduction about the Basic Software Design
- Idea about different Software Design Level and Design Strategies
- Detail of Software User Interface Design
- Software Design Complexity
- Software Implementation

6.1 Introduction :

Software design is a mechanism to convert user requirements into appropriate form, which helps the developers in software coding and implementation. It deals with by providing the client's requirement, as defined in SRS document, into a form, that is, easily implementable using programming language.

6.2 Software Design Basic :

Software design transfers the problem into solution, which is the first step in SDLC (Software Design Life Cycle). It attempts to specify how to satisfy the requirements stated in SRS.

6.2.1 Software Design Level :

There are three levels of software design which are as follows :

- **Interface Design**
- **Architectural Design**
- **Detailed Design**

- **Interface Design :**

Interface design is the specification of the interaction between a system and its environment. This phase proceeds at a high level of abstraction with respect to the inner workings of the system i.e., during interface design, the internal of the systems are completely ignored.

Attention is focused on the dialogue between the target system and the users, devices, and other systems with which it interacts. The design problem statement produced during the problem analysis step should identify the people, other systems, and devices which are collectively called agents.

Interface design should include the following details :

- ✓ Precise description of events in the environment, or messages from agents to which the system must respond.
- ✓ Precise description of the events or messages that the system must produce.
- ✓ Specification on the data, and the formats of the data coming into and going out of the system.
- ✓ Specification of the ordering and timing relationships between incoming events or messages, and outgoing events or outputs.

- **Architectural Design :**

Architectural design is the specification of the major components of a system, their responsibilities, properties, interfaces, and the relationships and interactions between them. In architectural design, the overall structure of the system is chosen, but the internal details of major components are ignored.

Issues in architectural design include :

- ✓ Gross decomposition of the systems into major components.
- ✓ Allocation of functional responsibilities to components.
- ✓ Component Interfaces
- ✓ Component scaling and performance properties, resource consumption properties, reliability properties, and so forth.
- ✓ Communication and interaction between components.

The architectural design adds important details ignored during the interface design. Design of the internals of the major components is ignored until the last phase of the design.

- **Detailed Design :**

Detailed design is the specification of the internal elements of all major system components, their properties, relationships, processing, and often their algorithms and the data structures.

The detailed design may include :

- ✓ Decomposition of major system components into program units.
- ✓ Allocation of functional responsibilities to units.
- ✓ User interfaces
- ✓ Unit states and state changes
- ✓ Data and control interaction between units
- ✓ Data packaging and implementation, including issues of scope and visibility of program elements
- ✓ Algorithms and data structures

6.2.2 Modularization :

Modularization is a method to divide software into multiple independent modules, which are capable of carrying out tasks independently. These modules work as basic concepts for the whole software. Designers are responsible to design modules so that it can be implemented separately and independently.

Modular design accidentally follows the rule of 'divide and conquer' strategy because many other benefits involved in the modular design of software.

- **Advantage of modularization :**

- ✓ Easier to manage smaller components.
- ✓ Program can be divided based on functional characteristics.
- ✓ Wanted level of concept can be taken in the program.
- ✓ Components can be re–used.
- ✓ Parallel execution can be made possible.
- ✓ Wanted from security feature.

6.2.3 Concurrency :

In the early days, software executed sequentially, means coded instruction executed one after another indicating only one part of program activated at any given time.

In software design, concurrency is executed by dividing the software into multiple independent components and executing them in parallel. In other words, concurrency provides ability to execute more than one part of software code in parallel to each other.

It is required for the programmers and designers to identify those modules, which can be made parallel execution.

6.2.4 Coupling and Cohesion :

When software is modularized, its tasks are divided into several modules based on some characteristics. However, modules considered as a single entity but, may refer to each other to work together. There are procedures by which module design quality and their interaction between them can be measured. These measures are called coupling and cohesion.

- **Cohesion :**

Cohesion is describing the degree of intra–dependability within elements of a module. The greater cohesion means better program design.

There are seven types of cohesion which are as follows :

- **Unplanned Cohesion –** An unplanned cohesion might be the result of breaking the software into smaller modules for the sake of modularization. It may serve confusion to the programmers and is generally not–accepted because it is unplanned.
- **Logical Cohesion –** When logically characterized elements are place together into a module, it is called logical cohesion.
- **Sequential Cohesion –** When elements of module are organized as they are processed in sequence at given time, it is called sequential cohesion.
- **Practical Cohesion –** When elements of module are grouped together, which are executed sequentially in order to perform a task, it is called practical cohesion.
- **Communicational Cohesion –** When elements of module are grouped together, which are executed sequentially and work on same data (information), it is called communicational cohesion.
- **Serial Cohesion –** When elements of module are grouped because the output of one element serves as input to another and so on, it is called serial cohesion.
- **Functional Cohesion –** It is measured to be the highest degree of cohesion, and it is highly expected. Elements of module in functional cohesion are grouped because they all contribute to a single well–defined function. It can also be reused.

- **Coupling :**

Coupling defines the level of inter–dependability among modules of software. It states at what level the modules restrict and interact with each other. The lower the coupling means the better the program.

There are five levels of coupling which are as follows :

- **Content Coupling –** When another module is able to access or modify or refer to the content directly, it is known as content coupling.
- **Global Coupling –** When one or more modules have read and write access to global data it is known as global coupling.
- **Control Coupling –** If one module decides the function of another module or modify the flow of execution it is known as control coupling.
- **Stamp Coupling –** When more than one module shares common data structure and work on different part of it, it is known as stamp coupling.
- **Data Coupling –** When two modules interact with each other by passing data it is known as data coupling. If a module provides data as parameter, then the receiving module should use all its components.

6.2.5 Design Verification :

The output of software design includes detailed description, documentation, and complete logic diagrams of all functional and non–functional requirements.

The next phase is software implementation so it is then necessary to verify the output before proceeding to the next phase. If design of the outputs is in formal symbolic form, then design tools should be used for verification else detailed design review can be used for verification and validation.

Reviewers can detect faults by structured verification method that might be produced by supervising some conditions. A good design review is important for good software design, accuracy, and quality.

❑ **Check Your Progress – 5 :**

1. _____ is the specification of the interaction between a system and its environment.

   a. Interface Design b. Architectural Design

   c. Detailed Design d. None of Above

2. Architectural design is specification of _____.

   a. Component of a system b. Responsibilities

   c. Properties & Interface d. All of Above

3. _____ is a method to divide a software into multiple independent modules

   a. Concurrency b. Modularization

   c. Coupling d. Cohesion

4. Write a note on Coupling & Cohesion.

6.3 Software Design Strategies :

Software design is a process to develop a thought of user requirements into software implementation. It takes requirements of user as challenges and goes to develop best solution. While the software is being developed, a plan is drawn to find the best possible design for implementing the proposed solution.

6.3.1 Structured Design :

Structured design is a process of develop a thought for finding solution of the problem with well–organized elements. As a benefit it gives better understanding of how the problem is being solved. It also makes it easy for designer to focus on the problem more correctly. In Structured design a problem is divided into small problems and each small problem is independently solved until the whole problem is solved and problems are solved by means of solution modules.

Structured design ensures that modules are well organized in order to get accurate solution. These modules are organized in hierarchy. All modules are communicating with each other.

A good structured design follows rules for communication between multiple modules, which are as follows :

- **Cohesion –** grouping of all functionally related elements.
- **Coupling –** communication between different modules.

A good structured design has high cohesion and low coupling arrangements.

6.3.2 Function Oriented Design :

The system includes of many smaller sub–systems known as functions in function–oriented design. These functions are capable of performing important task in the system. Some properties of structured design where divide and conquer methodology is used are inheriting in function–oriented design.

This design method divides the entire system into smaller functions, which provides means of idea by hiding the information and their operation. These functional modules can share information between themselves by information passing and using information available globally.

Another characteristic of functions is that when a program calls a function, the function changes the state of the program. Function oriented design works well where the system state does not matter and program work on input rather than on a state.

- **Design Process :**

✓ With the help of data flow diagram how data flows in entire system are seen.

✓ DFD shows how functions change data and state of the entire system.

✓ The entire system is logically divided into smaller parts known as functions based on its operation in the system.

✓ Each function is then defined at large.

6.3.3 Object Oriented Design :

Object Oriented Design (OOD) works on the entities and their characteristics instead of functions involved in the software. The entire concept of software solution spins around the engaged entities.

Important concepts of Object–Oriented Design are as follows :

Objects – Object is the basic run–time entities in an object–oriented system. As an object person, banks, company and customers can be there. Each entity has attributes and has some methods to perform on the attributes.

Classes – A class is the definition of the behavior and properties of one or more objects within the system. A class binds the data of an object to operations that it can perform. A class is a collection of objects of similar type.

Encapsulation – In OOD, the attributes and methods are bundled together is called encapsulation. As a attributes data variables and methods operation on the data are consider. As we seen encapsulation bundles information of an object together, but also restricts access of the data and methods from the outside world, it is called information hiding.

Inheritance – OOD permits similar classes to stand up in hierarchical style where the sub–classes can import, implement and re–use variables and methods from their immediate super classes it is called inheritance.

Polymorphism – OOD afford a mechanism where methods performing similar tasks with same name but different arguments it is called polymorphism.

- **Design Process :**

Software design process can be supposed as series of well–organized steps. However, it differs according to design approach function oriented or object oriented.

Design process may have the following steps :

- ✓ A solution design is created from requirement.
- ✓ Objects are recognized and grouped into classes on the basis of similarity in attribute features.
- ✓ Class hierarchy and relation between them is defined.
- ✓ Framework of the application is defined.

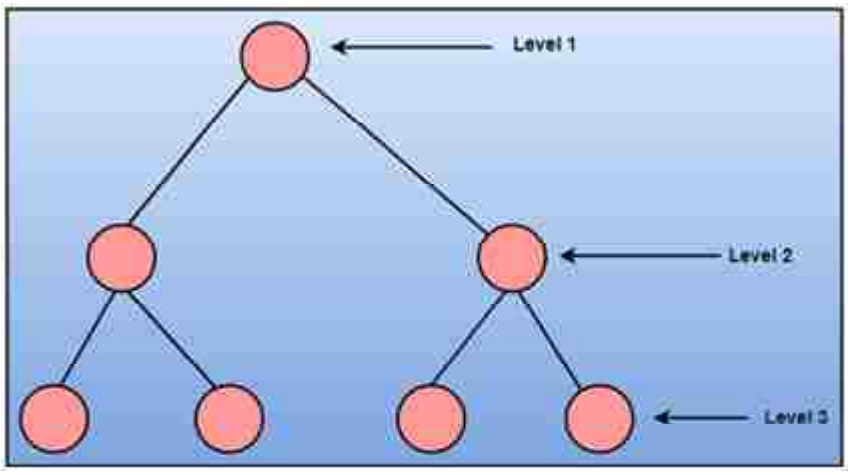
6.3.4 Software Design Approaches :

There are two general approaches for software designing which are as follows :

- **Top–Down Design :**

A system is composed of sub–system and it contains a number of components. Additionally, these sub–systems and components may have their own set of sub–systems and components, and creates hierarchical structure in the system.

Top–down design considers the whole system as single entity and then divides it to get more than one sub–system or component based on some characteristics. Each sub–system or component is then treated as a system and decomposed further. This process is carried on until the lowest level of system in the top–down hierarchy is achieved.



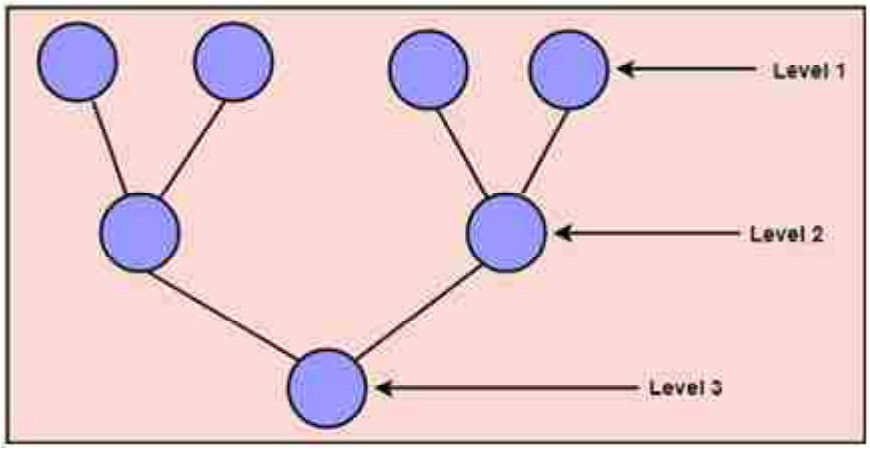
Top–Down Design

It starts with a generalized characteristic of system and keeps on defining the more specific part of it. When all the components are composed the entire system comes into existence.

Top–down design is more appropriate when the software solution needs to be planned from diagram and specific details are unidentified.

- **Bottom–up Design :**

The bottom–up design starts with most specific and basic components. It proceeds with combining higher level of components by using lower–level components. It keeps creating higher level components until the required system is not developed as single component.



Bottom–Up Design

Bottom–up policy is more appropriate when a system needs to be created from some existing system. Both, top–down and bottom–up approaches are not practical independently it is good to combine both.

❑ Check Your Progress – 2 :

1. OOD Stands for _____.

   a. Object Oriented Development b. Object Oriented Design

   c. Object Oriented Detail d. Object Oriented Document

2. Entities involved in the solution design are known as _____.

   a. Objects b. Classes c. Inheritance d. Encapsulation

3. An object is an instance of a _____.

   a. Inheritance b. Encapsulation c. Class d. Object

4. Write a note on Software Design Approaches.

6.4 Software User Interface Design :

User interface is the area of the application with which user is going to interact to work with software. With the help of user interface user can manipulate and control the software as well as hardware. As the today is a technological era you will find user interface in all digital technology like computers, mobile, music system, home appliances, cars, airplanes etc.

UI provides basic platform to users for human–computer interaction and is a part of software using which users looks into the software. As a part of UI there can be combination of text, graphics, audio, and video depending on the basic hardware and software.

The software becomes more popular if its user interface is :

- ✓ Attractive
- ✓ easy to use
- ✓ Gives response in short time
- ✓ Clear to understand
- ✓ Reliable on all interfacing screens

UI is divided into two categories which are as follows :

- ✓ Command Line Interface
- ✓ Graphical User Interface

6.4.1 Command Line Interface (CLI) :

CLI is a tool of interaction with the computer, where you have to provide the text–based command to the computer. In other words, CLI provides a command prompt, the place where the user types the command and feeds to the system. The user needs to remember the syntax of command and its use. Earlier CLI were not programmed to handle the user errors effectively.

A command is a text–based reference to set of instructions, which are expected to be executed by the system. There are methods like macros, scripts that make it easy for the user to operate.

CLI uses less amount of computer resource as compared to GUI.

- **CLI Elements :**

A text–based command line interface elements are as follows :

- o **Command Prompt –** It is text–based notifier that shows the framework in which the user is working. It is produced by the software system.
- o **Cursor –** It is a small horizontal line or a vertical bar which represent position of character while typing. Cursor is typically found in blinking state. It moves as the user writes or deletes characters.

- **Command –** A command is an executable instruction and may have one or more parameters. After the execution of command an output is shown on the screen. When output is produced, command prompt is displayed on the next line.

6.4.2 Graphical User Interface :

Graphical User Interface (GUI) provides graphical way to the user to interact with the system. Using GUI available options, user interact with the software in a faster speed with accuracy and more effectively.

- **GUI Element :**

GUI offers a set of components to work with software or hardware. Every graphical component provides a way to work with the system. Elements of GUI system are as follows :

Window – It is referred to an area where contents of application are shown. Contents in a window can be shown in the form of icons or lists, if the window represents file structure and it is easy for a user to navigate in the file system. Windows can be minimized, resized or maximized to the size of screen. They can be moved anywhere on the screen. A window may contain another window of the same application, called child window.

- **Tabs –** If an application permits executing multiple cases of it, they appear on the screen as separate windows.
- **Menu –** Menu is collection of standard commands, grouped together and placed at top of the application window.
- **Icon –** An icon is small picture representing an associated application. As you clicked or doubled click on, the application window is opened. Icon displays programs installed on a system in the form of small pictures.
- **Cursor –** It refers to interacting devices such as mouse, touch pad; digital pen is represented in GUI as cursors. Cursors are also called pointers. It is used to select menus, windows and other application features.

- **Application Specific GUI Components :**

A GUI of an application contains following GUI elements :

- **Application Window –** Most application windows use the concepts provided by operating systems but many use their own customer created windows to hold the contents of application.
- **Dialogue Box –** It is a child window that includes message for the user and request for some action to be perform. For Example : Application generates a dialogue to get confirmation from user to delete a file.

- **Text–Box** – It is an area where user types to enter text–based data.

- **Buttons** – It is used to submit inputs to the software.

- **Radio–button** – It is the element which is used to provide option to the user from which user can select any one option.

- **Check–box** – This function is similar to list–box. When an option is selected, the box is marked as checked user can also select multiple option.

- **List–box** – It provides list of available items for selection. More than one item can be selected.

6.4.3 User Interface Design Activities

There are a number of activities executed for designing user interface. The process of GUI design and implementation is similar to SDLC. Any model can be used for GUI implementation among Waterfall, Iterative or Spiral Model.

A model used for GUI design and development should fulfill these GUI specific steps.

- **GUI Requirement Gathering –** The designers should have list of all functional and non–functional requirements of GUI. Requirement is gathering from user and their existing software.
- **User Analysis –** The designer identifies the users who are going to use the software. The target audience matters as the design details change according to the knowledge and competency level of the user. If user is aware of technical things, advanced and complex GUI can be combined. For a beginner user, more information is included on how–to–use the software.
- **Task Analysis –** Designers identifies what task is to be done by the software solution. Here in GUI, it does not matter how it will be done. Tasks can be represented in hierarchical manner Tasks provide goals for GUI presentation. Flow of information between sub–tasks controls the flow of GUI contents in the software.
- **GUI Design and Implementation –** It refers to implements into code and insert the GUI with working software in the background.
- **Testing –** GUI testing can be done on usability, compatibility, and user acceptance.

6.4.4 GUI Implementation Tools :

There are many tools available using which the designers can implement full GUI on a mouse click. Some tools can be fixed into the software environment. GUI implementation tools provide collection of controls. Customization is also possible so designers can change the code accordingly.

There are different sections of GUI tools according to their different use and platform.

Example

Mobile GUI, Computer GUI, Touch–Screen GUI etc.

Example of GUI Tools

- Visual Studio, Android Wavemaker, LucidChart, FLUID

6.4.5 User Interface Rules :

There are some rules for GUI design which are as follows :

- **Consistency –** It refers to consistent sequences of actions should be required in similar situations. Identical terms should be used in prompts, menus, and help screens.
- **Enable users to use short–cuts –** The user should able to use shortcuts, function–keys, hidden commands during the interaction to reduce the number of interactions.
- **Offer feedback –** There should be feedback system for every user action. The feedback gives satisfactory result of the user on the software system design.
- **Dialog to produce closing –** Sequences of actions should be organized into groups with a starting, middle, and end.
- **Simple error handling –** If an error is made, the system should be able to detect it and offer simple, mechanisms for handling the error.
- **Permit easy reversal of actions –** This feature releases nervousness because the user aware about errors can be undone. It encourages study of unaware options.

❑ **Check Your Progress – 3 :**

1. CLI Stands for _____.

   a. Command Line Interface b. Command Line Instruction

   c. Command Line Information d. Command Line Implementation

2. GUI Stands for _____.

   a. Graphical User Instruction b. Graphical User Implementation

   c. Graphical User Information d. Graphical User Interface

3. Write a note on Command Line Interface.

4. Write a note on Graphical User Interface.

6.5 Software Design Complexity :

The word complexity refers to the events where there is a multiple interrelated links and extremely complex structure. As the software design is understood, various element and connection between those elements are step by step develop to be huge, which becomes too tough to understand at once.

Software design complexity is difficult to measure without using complexity metrics and measures. There are three important software complexity measures which are as follows :

6.5.1 Halsted's Complexity Measures :

In 1977, Mr. Maurice Howard Halstead introduced metrics to measure software complexity. Halstead's metrics depends upon the actual implementation of program and its measures, which are computed directly from the operators and operands from source code, in static manner. It allows to evaluate testing time, vocabulary, size, difficulty, errors, and efforts for C/C++/Java source code.

According to Halstead, "A computer program is an implementation of an algorithm considered to be a collection of tokens which can be classified as either operators or operands". Halstead metrics think a program as sequence of operators and their associated operands.

He defines various indicators to check complexity of module. Following table states the parameters and the meanings :

Parameter	Meaning
n1	Number of unique operators
n2	Number of unique operands
N1	Number of total occurrence of operators
N2	Number of total occurrence of operands

When we select source file to view its complexity details in Metric Viewer, the following result is seen in Metric Report :

Metric	Meaning	Mathematical Representation
n	Vocabulary	n1 + n2
N	Size	N1 + N2
V	Volume	Length * Log2 Vocabulary
D	Difficulty	(n1/2) * (N1/n2)
E	Efforts	Difficulty * Volume
B	Errors	Volume / 3000
T	Testing time	Time = Efforts / S, where S=18 seconds.

6.5.2 Cyclomatic Complexity Measures :

To perform some tasks to execute in order every program includes statements and decision statement which decides which statement need to be executed. The decision–making statement changes the flow of the program.

If we compare two programs of same size, the one with more decision–making statements will be more complex as the control of program jumps frequently.

McCabe, in 1976, proposed Cyclomatic Complexity Measure to quantify complexity of given software. It is graph based model that is work on decision–making concepts of program such as if–else, do–while, repeat–until, switch–case and goto statements.

Process to make flow control graph :

- ✓ Divide program in smaller blocks, enclosed by decision–making concepts.
- ✓ Create nodes representing each of these nodes.
- ✓ Connect nodes as follows :

If control can branch from block i to block j

Draw an arc

From exit node to entry node

Draw an arc.

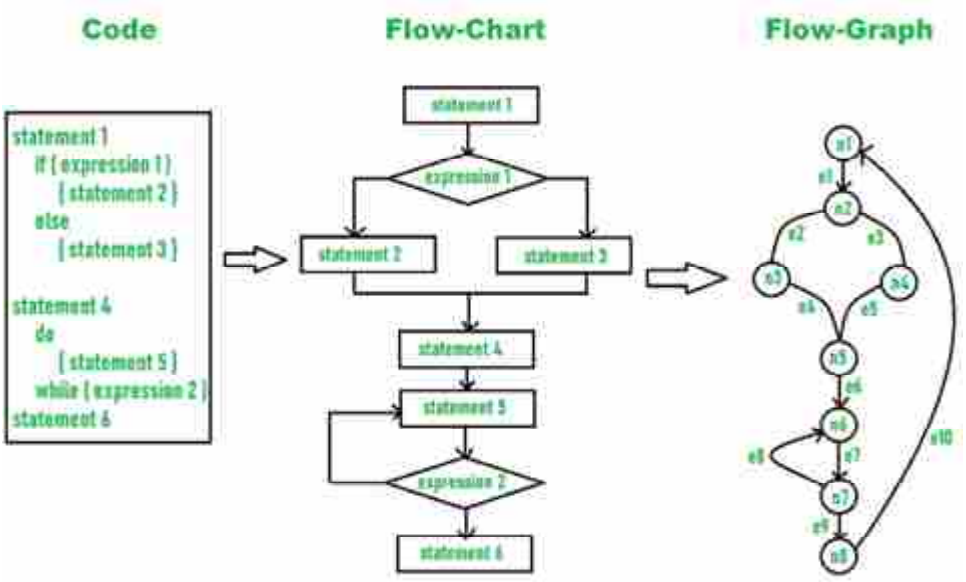
To calculate Cyclomatic complexity of a program module, we use the formula –

$V(G) = e - n + 2$

Where :

e is total number of edges

n is total number of nodes



The Cyclomatic complexity of the above module is

e = 10

n = 8

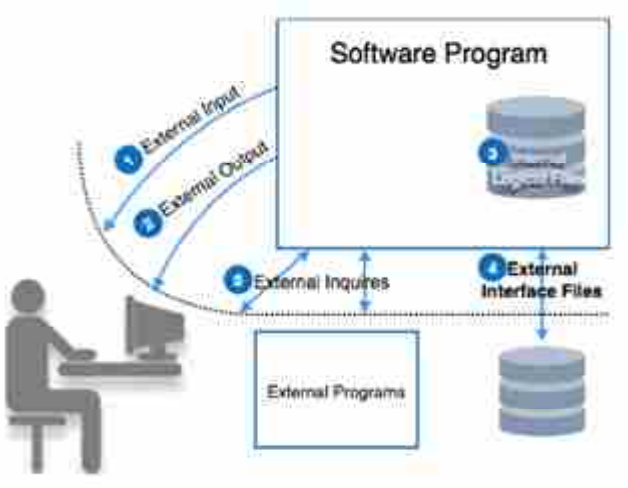
Cyclomatic Complexity = 10 – 8 + 2 = 4

According to P. Jorgensen, Cyclomatic Complexity of a module should not exceed 10.

6.5.3 Function Point :

It is broadly used to measure the size of software. Function Point focusses on functionality of the system. Features and functionality of the system are used to measure the software complexity.

Function point counts on five parameters, named as External Input, External Output, Logical Internal Files, External Interface Files, and External Inquiry. To consider the complexity of software each parameter is further categorized as simple, average or complex.



- **External Input :**

In system input from the outside of the system is known as external input. Uniqueness of input is measured, as no two inputs should have same formats. These inputs can either be data or control parameters.

Simple – if input count is low and affects less internal files

Complex – if input count is high and affects more internal files

Average – if–between simple and complex.

- **External Output :**

All output provided by the system are known as external output. Output is measured unique if their output format and/or processing are unique.

Simple – if output count is low

Complex – if output count is high

Average – if–between simple and complex.

- **Logical Internal Files :**

Every software system maintains internal files in order to maintain its functional information and to function properly. These files hold logical data of the system, it may contain both functional data and control data.

Simple – if number of record types are low

Complex – if numbers of record types are high

Average – if–between simple and complex.

- **External Interface Files :**

Software system may need to share its files with some external software or it may need to pass the file for processing or as parameter to some function. All these files are considered as external interface files.

Simple – if number of record types in shared file are low

Complex – if numbers of record types in shared file are high

Average – if–between simple and complex.

- **External Inquiry :**

An inquiry is a combination of input and output, where user sends some data to inquire about as input and the system responds to the user with the output of inquiry processed.

Simple – if query needs low processing and yields small amount of output data

Complex – if query needs high process and yields large amount of output data

Average – if–between simple and complex.

Each of these parameters in the system is specified weightage according to their class and complexity. The table below mentions the weightage given to each parameter :

Parameter	Simple	Average	Complex
Inputs	3	4	6
Outputs	4	5	7
Enquiry	3	4	6
Files	7	10	15
Interfaces	5	7	10

The above table produces raw Function Points. These function points are adjusted according to the environment complexity. System is described using fourteen different characteristics :

- ✓ Data communications
- ✓ Distributed processing
- ✓ Performance objectives
- ✓ Operation configuration load
- ✓ Transaction rate
- ✓ Online data entry,
- ✓ End user efficiency
- ✓ Online update
- ✓ Complex processing logic
- ✓ Re–usability
- ✓ Installation ease
- ✓ Operational ease
- ✓ Multiple sites
- ✓ Desire to facilitate changes

These characteristics factors are then rated from 0 to 5, as mentioned below :

- ✓ No impact
- ✓ Related
- ✓ Reasonable
- ✓ Average
- ✓ Significant
- ✓ Essential

❑ **Check Your Progress – 4 :**

1. Write a note on Halsted's Complexity Measures.

2. Write a note on Cyclomatic Complexity Measures.

3. Write a note on Function Point.

6.6 Software Implementation :

We will learn about programming methods, documentation and challenges in software implementation.

6.6.1 Structured Programming :

Structured programming inspires the developer to use functions and loops instead of using simple jumps in the code, so bringing simplicity in the code and improving its efficiency. Structured programming also helps programmer to reduce coding time and organize code properly.

Structured programming defines how the program shall be coded. It uses three main concepts which are as follows :

1. **Top–down analysis –** In top–down analysis, the problem is divided into small parts where each part has some meaning. Each problem is solved separately and solving steps are clearly defined.
2. **Modular Programming –** While programming, the code is divided into smaller group of instructions which is known as modules. Modular programming based on the understanding of top–down analysis. Jumps are prohibited and modular format is encouraged in structured programming.
3. **Structured Coding –** In reference with top–down analysis, structured coding sub–divides the modules into further smaller units of code in the order of their execution. Structured programming controls the flow of the program, whereas structured coding uses control structure to organize its instructions in definable patterns.

6.6.2 Functional Programming :

Functional programming is style of programming language, which uses the concepts of mathematical functions. A function in mathematics always gives the same result on receiving the same argument.

Functional programming uses the following concepts :

- **First class and High–order functions –** It refers to functions which have ability to accept another function as argument or they return other functions as results.
- **Pure functions –** It refers to functions that do not include negative updates, they do not affect any I/O or memory and if they are not in use, they can easily be removed without hampering the rest of the program.
- **Recursion –** It refers to programming technique where a function calls itself and repeats the program code in it unless some pre–defined condition matches. Recursion is the method of creating loops in functional programming.
- **Strict evaluation –** It is referring to a method of evaluating the expression passed to a function as an argument. Functional programming has two types of evaluation methods, strict (eager) or non–strict (lazy). Strict evaluation always evaluates the expression before invoking the function. Non–strict evaluation does not evaluate the expression unless it is needed.

6.6.3 Programming Style :

Programming style is set of coding rules followed by all the programmers to write the code. When multiple programmers work on the same project, they need to work with the program code written by some other developer.

A proper programming style contains using function and variable names relevant to the planned task, using well–placed indentation, and including comment in code for the reader comfort. This makes the program code readable and understandable by all, which in turn makes debugging and error solving easier.

- **Coding Guidelines :**

Practice of coding style differs with organizations, operating systems and language of coding itself. The following coding elements may be defined under coding guidelines of an organization :

- **Naming conventions –** It refers to how to name functions, variables, constants and global variables.
- **Indenting –** It refers to the space left at the beginning of line, usually single tab.
- **Whitespace –** It is generally omitted at the end of line.
- **Operators –** It refers to the rules of writing mathematical, assignment and logical operators. For example, assignment operator '=' should have space before and after it, as in "x = 2".
- **Control Structures –** It refers to the rules of writing decision statements and control flow statement such as if–then–else, case–switch, while–until and in nested fashion.

- **Line length and wrapping –** It refers to how many characters are there in one line, mostly a line is 80 characters long. Wrapping defines how a line should be wrapped, if is too long.
- **Functions –** It refers to how functions should be declared and invoked, with and without parameters.
- **Variables –** It refers to how variables with different data types are declared and defined.
- **Comments –** The comments included in the code describe what the code actually does and it is the important component of coding.

6.6.4 Software Documentation :

Software documentation is a vital part of software process. A document provides information to know about software process and about how to use the software.

A well–maintained documentation should involve the following documents :

- **Requirement Documentation :**

As the requirement are gather from various stakeholders by analyst it uses as key tool for software designer, developer, and the test team to carry out their respective tasks. It includes all the functional, non–functional and behavioral description of the software.

Information of this document can be gathered from client using the fact–finding techniques, or from historical data of the software. It is works as a basis of the software to be developed and is majorly used in verification and validation phases.

- **Software Design Documentation :**

These documentations include all the necessary information, which are needed to build the software. It contains : (a) Architecture of High–level software, (b) Detail of Software design, (c) Data flow diagrams, (d) Database design

These documents work as source for developers to implement the software. However, these documents do not give any details on how to code the program; they give all necessary information that is required for coding and implementation.

- **Technical Documentation :**

These documentations are managed by the developers and actual coders which represent information about the code. While writing the code, the programmers also mention objective of the code, who wrote it, where will it be required, what it does and how it does, what other resources the code uses, etc.

This documentation increases the understanding among various programmers working on the same code. It enhances re–use capability of the code. It makes debugging easy and traceable.

- **User Documentation :**

This documentation is different from all the above, this explains how the software product should work and how it should be used to get the wanted results.

These documentations may include software installation procedures, how–to guides, user–guides, uninstallation method and special references to get more information like license updations etc.

6.6.5 Software Implementation Challenges :

There are some challenges faced by the development team while implementing the software are as follows :

- **Code–reuse –** Programming interfaces of present–day languages are very sophisticated and are equipped huge library functions. Still, to bring the cost down of end product, the organization management prefers to re–use the code, which was created earlier for some other software.
- **Version Management –** As the technology enhances day by day every time new software is issued to the customer, developers have to maintain version and configuration related documentation. This documentation needs to be highly accurate and available on time.
- **Target–Host –** The software program, which is being developed in the organization, needs to be designed for host machines at the customers end.

But at times, it is impossible to design software that works on the target machines.

❑ **Check Your Progress – 5 :**

1. Write a note on Structured Programming.

2. Write a note on Functional Programming.

3. Write a note on Programming Style.

4. Explain Software Documentation.

5. Explain Software Implementation Challenges.

6.7 Let Us Sum Up :

In this unit we have learnt Basic Software Design, Idea about different Software Design Level and Design Strategies. As part of Software Design Interface Design, Architectural Design and Detailed Design play an important role.

We have also seen about the modularization is a method to divide a software into multiple independent modules, which are capable of carrying out tasks independently. Software design is a process to develop a thought of user requirements into software implementation.

We also learnt about User interface is the area of the application with which user is going to interact to work with software. With the help of user interface user can manipulate and control the software as well as hardware.

We also discussed about the Software Implementation in which structured programming, functional programming, programming style, and software documentation are going to be consider and make implementation successful.

6.8 Answers for Check Your Progress :

❑ **Check Your Progress 1 :**

1. (a) 2. (d) 3. (b) 4. (Refer 6.2.4)

❑ **Check Your Progress 2 :**

1. (b) 2. (a) 3. (c) 4. (Refer 6.3.4)

❑ **Check Your Progress 3 :**

1. (a) 2. (d)

3. (Refer 6.4.1) 4. (Refer 6.4.2)

❑ **Check Your Progress 4 :**

1. (Refer 6.5.1) 2. (Refer 6.5.2) 3. (Refer 6.5.3)

❑ **Check Your Progress 5 :**

1. (Refer 6.6.1) 2. (Refer 6.6.2) 3. (Refer 6.6.3)

4. (Refer 6.6.4) 5. (Refer 6.6.5)

6.9 Glossary :

1. **Interface Design –** Interface design is the specification of the interaction between a system and its environment.
2. **Architectural Design –** Architectural design is the specification of the major components of a system, their responsibilities, properties, interfaces, and the relationships and interactions between them.
3. **Detailed Design –** Design is the specification of the internal elements of all major system components, their properties, relationships, processing, and often their algorithms and the data structures.
4. **Cohesion –** Cohesion is describing the degree of intra–dependability within elements of a module.
5. **Coupling –** Coupling defines the level of inter–dependability among modules of software.
6. **Command Prompt –** It is text–based notifier that shows the framework in which the user is working.
7. **Command –** A command is an executable instruction and may have one or more parameters.
8. **Icon –** An icon is small picture representing an associated application.
9. **Menu –** Menu is collection of standard commands, grouped together and placed at top of the application window.
10. **Cursor –** It is a small horizontal line or a vertical bar which represent position of character while typing.

6.10 Assignment :

1. Write a detailed note on Graphical User Interface.

6.11 Activities :

1. Explain GUI Implementation Tool.

6.12 Case Study :

Discuss Halsted's Complexity Measures and Cyclomatic Complexity Measures.

6.13 Further Reading :

1. Software Engineering : A Practitioner's Approach Book by Roger S. Pressman
2. Braude, Formal Inspections in Software Quality Assurance, 1998
3. Beck, A.C.; Mattos, J.C.B.; Wagner, F.R.; Carro, L. CACOPS : General Purpose Configurable Power Simulator, 2003.

Unit 07

SOFTWARE QUALITY MANAGEMENT

UNIT STRUCTURE

7.0 Learning Objectives

7.1 Introduction

7.2 Software Quality

7.3 Verification & Validation (V & V)

7.4 Quality Control

7.5 Inspection

7.6 Walkthrough and Review

7.7 Why Standards ?

7.8 Software Quality Metrics or Parameters

7.9 Five levels of Capability Maturity Model (CMM)

7.10 Let Us Sum Up

7.11 Answers for Check Your Progress

7.12 Glossary

7.13 Assignment

7.14 Activities

7.15 Case Study

7.16 Further Readings

7.0 Learning Objectives :

After learning this unit, you will be able to understand :

- The quality management process
- Quality management activities
- Explain role of standards in quality management
- Concept of software metric, predictor metrics and control metrics
- Idea about measurement in assessing software quality

7.1 Introduction :

Software quality management is related with ensuring the required level of quality which Software should have in order to work with its product. Appropriate quality standards and procedures will make sure that such quality standard in software needs to be adopted. It is moreover going well with quality culture that appears due to creation and maintenance of certain standards with responsibility.

Quality Management Activities :

- **Quality Assurance :** It is related to establishing organizational quality standards and procedures.

- **Quality Planning :** This will adopt and select applicable quality standards and procedures for particular software project.
- **Quality Control :** This will encompass quality standards and procedures that are followed at the development time by development team.

7.2 Software Quality :

In software, quality is related to different variables that depend on external and internal quality factors. In case of external quality, it shows that the user experiences come in consideration in terms of running and handling software matters, while internal quality concern with designing of software by use of codes which appears to be at backhand and is not visible to end–user. It is noted that external quality is main aspect to the user, while internal quality is basically for software developer.

Software quality appears as a result from use and working of end user. It is noted that software quality cannot be reactive action to external defects. The quality in software is related with working and development right from scratch using certain design and development principles and methodologies which has to focus on three prime factors :

- **Testability**
- **Coverage**
- **Flexibility**

First table shows criteria to be followed during External Quality, while Second table shows criteria to be followed during internal quality.

	User	Developer	Measurable
External Qualities			
Feature	X		Yes
Speed	X	X	Yes
Space	X	X	Yes
Network Usage	X	X	Yes
Stability	X	X	Yes
Robustness	X	X	Somewhat
Ease–of–Use	X	X	Subjective
Determinism	X	X	Yes
Back–Compatibility	X		Yes
Security	X		Difficult
Power Consumption	X		Difficult

[External Qualities]

	User	Developer	Measurable
Internal Qualities			
Test Coverage		X	Yes
Testability		X	Hard
Portability		X	Somewhat
Thread–Safeness		X	Hard
Conciseness		X	Somewhat
Maintainability		X	Hard
Documentation		X	Subjective
Legibility		X	Subjective
Scalability		X	Somewhat

[Internal Qualities]

It is seen that many quality criteria are objective that can be measured accordingly, while some quality criteria are subjective that are obtained with more arbitrary measurements. We see that quality in software means that a product should meet all necessary requirements which can be :

- ✓ Problem for software systems
- ✓ Shows discrepancy among customer quality and software developer requirements
- ✓ Difficult to specify in a definite way
- ✓ Incomplete and inconsistent

It seems that quality management is not simply related with lowering defects but also with several other features and qualities. There are certain Quality Standards which should be kept in mind during development :

- ✓ Product standards with features using components related to style of program or skills.
- ✓ Process standards with features highlighting process implementation
- ✓ Encompassing best practices to avoid previous mistakes
- ✓ Designing as per organizational priorities with interference of new team members

With this we see that there is certain standard which have to be adopted and followed in order to keep the quality standard high in software which are :

- ✓ Functional suitability
- ✓ Reliability
- ✓ Operability
- ✓ Quality performance
- ✓ Security
- ✓ Compatibility
- ✓ Maintainability
- ✓ Transferability

In a quality model, there appears following characteristics features such as :

Effectiveness

- ✓ Efficiency
- ✓ Satisfaction
- ✓ Safety
- ✓ Usability

The table below shows the responsibility of Management related to quality.

Management Responsibility	Quality System
Control of non–conforming products	Design Control
Handling, Storage, Packaging and Delivery	Purchasing
Purchaser–Supplied Products	Product Identification and Traceability
Process Control	Inspection and Testing
Inspection and Test Equipment	Inspection and Test Status
Control Review	Corrective Action
Document Control	Quality Records
Internal Quality Audits	Training
Servicing	Statistical Techniques

[Management responsibility regarding standard]

❑ **Check Your Progress – 1 :**

1. Internal qualities are linked with :

   a. Software b. Coding c. Programming d. Development

7.3 Verification & Validation (V & V) :

Verification and Validation is a methodology that will find and assess correctness and standard of quality in software all the way in software life cycle. This methodology focuses on ensuring required parameters to be adopted which software should meet that lead to correct objectives and helps in managing risk.

Verification is a process of finding the development phase work products of software development lifecycle so as to check whether the verification is done in right track at time of creating final product. Validation on the other hand is also a process which finds out whether the software is meeting the required business needs during the final product.

Such method is good which when initiated at the time of acquisition process and all the way in life cycle software development. In this, the level is a range of values which shows :

- ✓ Complexity in software
- ✓ Criticality
- ✓ Risk

- ✓ Level of safety
- ✓ Level of security
- ✓ Required performance
- ✓ Reliability
- ✓ Project unique characteristics

This methodology makes use of following techniques in order to verify software at the time of development process :

- ✓ Peer Reviews
- ✓ Documentation inspections
- ✓ Requirements/design/code reading
- ✓ Test witnessing
- ✓ Installation audits

It is noted that both Verification and Validation tasks is different during the development lifecycle phase. Verification and validation are done in every phase of lifecycle such as :

In Planning :

- ✓ Contract Verification
- ✓ Finding the Concept document
- ✓ Working with risk analysis

In Requirement :

- ✓ Finding software needs
- ✓ Analyze and locate interfaces
- ✓ Framing of test plans in system
- ✓ Framing of Acceptance test plan

In Design :

- ✓ Finding design for software
- ✓ Analyze User Interface

Framing of Integrated test plan :

- ✓ Framing of Component test plan
- ✓ Framing of test design

In Implementation :

- ✓ Finding source code
- ✓ Finding of documents
- ✓ Framing of test cases
- ✓ Finding the test procedure
- ✓ Finalization of Components test cases

In Testing :

- ✓ Working on systems test case
- ✓ Finding of acceptance test case
- ✓ Updating traceability metrics
- ✓ Framing of Risk analysis

In Installation and checkout :

- ✓ Auditing the installation and configuration
- ✓ Implement the final test installation from candidate
- ✓ Framing of final test report

In Operation :

- ✓ Finding new constraint
- ✓ Evaluating change proposed

In Maintenance :

- ✓ Finding of anomalies
- ✓ Estimation of migration
- ✓ Estimation of retrial features
- ✓ Estimation of proposed change
- ✓ Finding production issues

Benefits of Verification and Validation :

- ✓ With this method, we can locate leads much before in order to create better solution instead of fixing them.
- ✓ Finding the solution using correct problem against software needs.
- ✓ Shows objective evidence of software and system fulfillment in terms of quality standards.
- ✓ Shows process improvements along with feedback on quality of development process and products

❑ **Check Your Progress – 2 :**

1. Verification is related to :

   a. Finished Product

   b. Final Delivery of Product

   c. In between Product Development

   d. None of Above

7.4 Quality Control :

Quality Control in Software carries various activities which ensure quality in software products. The idea behind quality control in software is to check whether the project follows its standards processes and procedures. Also, it checks for software project that produces required deliverable products which can be internal and external.

Every Organisation defines its own internal quality standards, processes and procedures and develops the product based on certain criteria and parameters. The process of assurance from stakeholders presents standards and procedures that are normally concern in terms of quality control including verification.

- Design
- Code
- Deployment Plan
- Test Plan

- Test Cases
- Unit Testing
- Integration Testing
- System Testing
- Acceptance Testing

Software Quality Control is restricted to Review or testing phases of Software Development Life Cycle. Its main idea is to make sure that the products should meet particular specifications or requirements.

It is seen that process of Software Quality Control is managed by Software Quality Assurance which is oriented towards prevention while Software Quality Control is oriented towards detection. It is a wide practice which is applied for assuring quality of products or services where constant effort enhances the quality practices in an organization that lead to regular improvements and enhancement of product. In many Organisations, process programmers allocate for enhancing processes and procedures along with quality assurance team.

Moreover, we can see that Software Quality Control is concern with :

- Measures and controls of quality in software product being developed.
- Routine checking of quality to make sure about errors in software.
- Identification and addressing of product errors and defects.
- Development of final product without intervention of errors.
- Testing activities related to product.

❑ **Check Your Progress – 3 :**

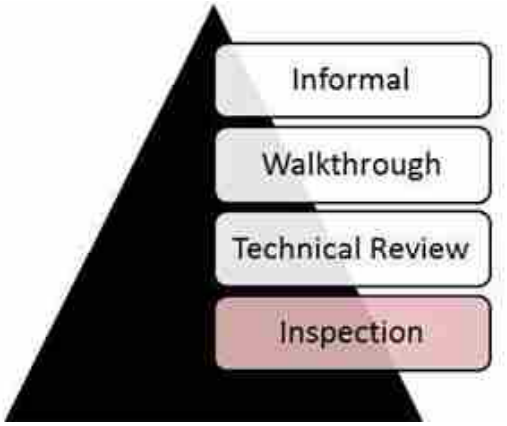
1. Software Quality is related to :

   a. Following Instruction b. Software Standard

   c. Software Parameters d. All of Above

7.5 Inspection :

Software inspection or code inspection relates to design and documentation along with source code. Inspection is a kind of reviews or a strategy which is adopted during static testing phase of software.



[Testing Phases]

Inspection is done by trained staff which does peer examination of product. It is formal and worked using checklists and rules. It serves as review

process that uses entry and exit features and rules. Inspection requires pre–meeting preparation which prepares a report which is shared with developer for correct actions.

Code inspections are good test method that takes more time and could locate 80% of contained faults. A proper code inspection takes days and require tools to browse symbols so as to find places where required. Proper inspections can be applied for almost all work products in the software life cycle. Initially it requires time but statistical evaluations show whole life cycle of software development which save resources and improves quality of product.

There are certain elements involved in inspection process :

- Explicit entry
- Exit criteria
- Individual preparation by inspectors
- Roles of moderator, reader, producer, and recorder
- Training for moderators
- Use of checklist
- Limitation to identify and classify defects
- Needs successful completion of rework for proper inspection
- Formal data collection, reporting, and analysis

The idea behind inspection is to :

- Helps in improving quality of document under inspection
- Remove defects efficiently
- Improve product quality
- Create understanding by exchanging information
- Prevent occurrence of defects

❑ **Check Your Progress – 4 :**

1. Software Inspection could be carried out by :

   a. New Joined b. New Developer

   c. Trained Quality Checker d. None of Above

7.6 Walkthrough and Review :

Review is concern with work product that examines the defects by an individual who may not be a developer. Work Product is important deliverable which is designed at the time of requirements, designing, coding or testing phase of software development. It is seen that reviews seems to be the best ways in order to make sure the quality requirements which gives maximum return on investment. It helps in finding defects which make sure about product compliance to specifications, standards or regulations.

Walkthrough is a procedure where developer describes product and look on the comments send from the end user. This program explains the user about the product information rather than entering into product rectification part. It is normally done at the time of testing which can be :

- Formal process
- Led by developer

- Process of guiding participants by document as per the process
- Required for non–software professional who have no idea about process of software development.

The basic idea behind walkthrough is to :

- Show documents inside and outside software discipline to have information on topics under documentation.
- Explain or transfer knowledge by evaluating contents of document
- Achieve understanding and taking feedback.
- Examine and discuss validity of proposed solutions

❑ **Check Your Progress – 5 :**

1. Walkthrough is a procedure that explain about :

a. Product b. Process c. Issues d. All of Above

7.7 Why Standards ?

It is not that there could be issues related to quality and process while developing software product. For formalization of product, many software companies are nowadays coming with their own standards that carry effective key features which are favorable for effective quality management. Standards could be international, national, organizational or related to any project.

It is seen that product standards explains about the characteristics where every components exhibit a common programming style. Whereas process standards define how the software process should be performed.

There are many reasons for Quality Standards in Software :

- Ensuring best practice by avoiding repetitive mistakes
- Involving certain quality assurance checking standard compliance
- Makes new join to understand better about the quality produced and standards

Quality Standards involves :

- Practitioners which could be an Engineer to understand the rationale of particular standard
- Reviewing of standards with proper utilization.
- Certain tool which supports quality with less of clerical work.

There are certain attributes related to Quality standards :

Safety	Understandability	Portability
Security	Testability	Usability
Reliability	Adaptability	Reusability
Resilience	Modularity	Efficiency
Robustness	Complexity	Learnability

❑ **Check Your Progress – 6 :**

1. Standard could be in terms of :

a. Software b. Project c. Plan d. All of Above

7.8 Software Quality Metrics or Parameters :

Measurement allows an organization in improving the process, planning, tracking and controlling of software projects and thereby proper assessing of quality. It is a measure of particular attributes of process, project and product which is used to calculate software metrics. Metrics are analyzed and provides a dashboard to management on overall progress of process, project and product. Generally, the validation of metrics is regular phenomena where spanning of multiple projects takes place. Type of metrics used normally account for checking of quality requirements that can be achieved during software development process. In quality assurance process, metric is required to revalidate all time when applied.

Software Metrics is of classes :

Process Metrics :

- It measures the efficiency of processes
- It stretches on quality achieved as a result of managed process
- It is a reusable data mostly applied for prediction
- It is a form of defect removal efficiency

Project Metrics :

- It will assess the status of projects
- It helps in tracking of risk
- It finds the problem areas
- It balances the work flow
- Examples are :
- Effort/time per SE task
- Defects detected per review hour
- Scheduled vs. actual milestone dates
- Distribution of effort on SE tasks

Product Metrics :

- It will measure predefined product attributes
- It stresses on quality of deliverables
- Examples are : Code or design complexity, Program size, Defect count, Reliability

It is found that software quality metrics is a subset of software metrics which stresses on quality of product, process and project. It is closely linked with process and product metrics as compared to project metrics. It carries project parameters like number of developers with skill levels, schedule, size, and affecting product quality of an Organisation. It is broadly divided into :

- End–product quality metrics
- In–process quality metrics

Quality metrics is used at the time of software design and for non–embedded systems. In case of embedded system, community is not ready to use certain new software technique which tends to lose major evolution of software design methodologies. Using quality metrics in software product at

the time of software designing will help in evaluating several levels which can be a solution confirming all requirements with better architecture, good reusing conditions and flexibility.

It is found that Hewlett–Packard on using Software Quality Metrics follows five quality parameters which are in terms of :

- Functionality
- Usability
- Reliability
- Performance
- Serviceability

It is noted that for many most software quality assurance systems, common software metrics which checks for improvement relates to :

- Source lines of code
- Cyclical complexity of code
- Function point analysis
- Bugs per line of code
- Code coverage
- Number of classes and interfaces
- Cohesion and coupling among modules

Common software metrics include :

- Bugs per line of code
- Code coverage
- Cohesion
- Coupling
- Cyclomatic complexity
- Function point analysis
- Number of classes and interfaces
- Number of lines of customer requirements
- Order of growth
- Source lines of code
- Robert Cecil Martin's software package metrics

Software Quality Metrics focus on the process, project and product. By analyzing the metrics, the organization, the organization can take corrective action to fix those areas in the process, project or product which are the cause of the software defects.

❑ **Check Your Progress – 7 :**

1. Standard could be in terms of :

   a. Distribution of effort b. Design Complexity

   c. Program Size d. Reliability

7.9 Five levels of Capability Maturity Model (CMM) :

Capability Maturity Model also CMM is a model of process maturity for software development. It is an evolutionary model of progress of company's abilities in order to frame software. In software Development Company there are standards for processes of development, testing and application which work with certain rules for appearance of final program code, components, interfaces, etc.

The CMM model describes five–level path showing organized and systematically arranged processes. It was framed by Software Engineering Institute which happens to be an R&D center of U.S. Department of Defense.

This model is similar to ISO 9001 with one ISO 9000 series of standards that was specified by International Organization for Standardization. ISO 9000 standards show prominent quality system for manufacturing and service industries and deals with development and maintenance of software. As seen, ISO 9001 specifies least acceptable quality level for software processes, while CMM uses framework for continuous process improvement and is more functional as compared to ISO standard.

[Level of CMM Model]

The CMM model defines five levels of organizational maturity :

Initial Level : This is the basic level which compares with next levels. It a company the initial level conditions are not fit for any development of good software; hence the outcome depends on higher authorities or on manager's approach and programmers' experience. So, this level depends on higher authorities.

Repeatable Level : It is the next level where project management technologies take part which includes project planning and management. It results in experience and standards of particular employees for producing software's with exercise on special quality management.

Defined Level : This level employs in designing and framing of standards used at the processing of software development and maintenance. These are documented along with a standard that uses effective technologies. It comprises of superior quality management department which creates building and maintenance standards which can only be possible with advanced training in order to achieve better quality. In this level, degree of organizational dependence on qualities of developer decreases with no possibility of rolling back of process to previous level in certain critical situations.

Managed Level : This level carries quantitative indices as software and process which originates in an organization. In this, high project management is obtained with the result of decrease of digression in various project indices.

On the other hand, good variations in process efficiency results in different from random variations in mastered areas.

Optimizing Level : This level is engaged with improvement related to procedures in existing processes along with evaluation of efficiency for newly innovative technologies. The idea behind this level in an organisation is to permanently improve the existing processes that anticipate possible errors and defects. With this, there result decrease in the costs of software development by way of using and developing previously used components.

❑ **Check Your Progress – 8 :**

1. Which among following levels depends on Higher Authority Decisions ?

   a. Initial Level b. Repeatable Level

   c. Defined Level d. Managed Level

7.10 Let Us Sum Up :

In this unit we have learnt that software quality is related to various variables which rely on external and internal quality factors. The external quality is about user experiences in terms of running and handling software matters, while internal quality is designing of software using codes.

The idea about verification and validation involves methods which finds and assess correctness and quality standards in software in software life cycle. Quality Control in Software relates to activities that make sure about quality in software products which checks for project standardize processes and procedures.

Software inspection or code inspection relates to design and documentation along with source code. Review is work product that examines defects by an individual while walkthrough is a procedure where developer describes product and comment on end user comments. Capability Maturity Model is a process maturity for software development which shows progress of company's abilities to frame software.

7.11 Answers for Check Your Progress :

❑ **Check Your Progress 1 :**

1. (b)

❑ **Check Your Progress 2 :**

1. (c)

❑ **Check Your Progress 3 :**

1. (d)

❑ **Check Your Progress 4 :**

1. (c)

❑ **Check Your Progress 5 :**

1. (a)

❑ **Check Your Progress 6 :**

1. (d)

❑ **Check Your Progress 7 :**

1. (a)

❑ **Check Your Progress 8 :**

1. (a)

7.12 Glossary :

1. **Software Quality –** Activities that depends on external and internal quality factors.
2. **External Quality –** Parameters on which user experiences depends in terms of running and handling software.
3. **Internal Quality –** Parameters concern with designing of software using codes.
4. **Verification –** To cross check or very document before start of software project or development.
5. **Validation –** It is methods that locates and assess correctness and quality standards in SDA cycle.
6. **Quality Control –** It is measure related to quality standard of software.

7.13 Assignment :

1. Explain about verification and validation in terms of software ?

7.14 Activities :

1. Explain about CMM model.

7.15 Case Study :

Discuss various standards adopted by companies to maintain quality in software ?

7.16 Further Reading :

1. McConnell's, Software Project Survival, 2011
2. Braude, Formal Inspections in Software Quality Assurance, 1998
3. Beck, A.C.; Mattos, J.C.B.; Wagner, F.R.; Carro, L. CACOPS : General Purpose Configurable Power Simulator, 2003.
4. Chatzigeorgiou, A.; Stephanides, G. Evaluating Performance of Procedural Programming Software Technologies, 2002.

Unit
08

SOFTWARE TESTING TECHNIQUE

UNIT STRUCTURE

8.0 Learning Objectives

8.1 Introduction

8.2 Software Validation & Verification

8.3 Manual VS Automated Testing

8.4 Testing Approaches

- **8.4.1 Black–Box Testing**
- **8.4.2 White–Box Testing**

8.5 Testing Levels

- **8.5.1 Unit Testing**
- **8.5.2 Integration Testing**
- **8.5.3 System Testing**
- **8.5.4 Acceptance Testing**
- **8.5.5 Regression Testing**

8.6 Function Test Plan

8.7 Process of Testing

8.8 Testing Documentation

- **8.8.1 Before Testing**
- **8.8.2 While Being Tested**
- **8.8.3 After Testing**

8.9 Grey Box Testing

8.10 Non–Functional Testing

8.11 Testing Artifacts

8.12 Let Us Sum Up

8.13 Answers for Check Your Progress

8.14 Glossary

8.15 Assignment

8.16 Activities

8.17 Case Study

8.18 Further Readings

8.0 Learning Objectives :

After learning this unit, you will be able to understand :

- Software Validation & Verification
- Testing Approaches
- Testing Level

- Function Test Plan
- Process of Testing
- Testing Documentation
- Non–Functional Testing
- Testing Artifacts

8.1 Introduction :

Software testing is the procedure of assessment a software product to detect differences between given input and expected output. Testing assesses the quality of the product. Software testing is a process that should be done during the development process.

In other words, software testing is a verification and validation process. There are two fundamentals of software testing which are as follows :

- Black box testing
- White box testing

- **Black box Testing :** Black box testing is a testing technique that ignores the internal mechanism of the system and focuses on the output generated against any input and execution of the system. It is also called functional testing.
- **White box Testing :** White box testing is a testing technique that takes into account the internal mechanism of a system. It is also called structural testing and glass box testing.

8.2 Software Validation & Verification :

Software testing includes Validation and Verification which are as follows :

- **Software Validation :**

Validation refers to process where it is going to check about specified requirement of user is fulfill or not. It is carried out at the end of the SDLC. If developed software fulfill user's requirement then it is validated.

✓ Validation ensures that software is as per the user requirements.

✓ Validation answers the question – "Are we developing the product which attempts all that user needs from this software ?".

✓ Validation underlines on user requirements.

- **Software Verification :**

Verification is the procedure of confirming that the software is fulfilling the business requirement and is developed by following the specification and methodologies.

✓ Verification ensures the software being developed is as per the design specifications.

✓ Verification answers the question– "Are we developing this product by firmly following all design specifications ?"

✓ Verifications focus on the design and system specifications.

There are some targets of the test which are as follows :

- **Errors –** It is referring to coding errors made by developers. As well there is a difference in output of software and required output, is known as an error.
- **Fault –** When there is an error exist fault occurs. A fault, also called as a bug, is a result of an error which can cause system to fail.
- **Failure –** failure is refers to the inability of the system to perform the required task. Failure occurs when fault exists in the system.

❑ **Check Your Progress – 1 :**

1. _____ ensures that software is as per the user requirements.

   a. Verification b. Validation c. Both a and b d. None of Above

2. _____ ensures that software is as per the user requirements.

   a. Verification b. Validation c. Both a and b d. None of Above

3. _____ is refers to the inability of the system to perform the required task.

   a. Errors b. Fault c. Failure d. All of Above

8.3 Manual VS Automated Testing :

Testing can be performed either manually or automated which are as follows :

- **Manual –** In manual testing software tester makes test cases for various segments and level of code, executes the test cases and provides the results to the manager. It is time consuming process as well tester need to check whether proper test cases are used or not.
- **Automated –** In automated testing, testing procedure is done with the help of automated testing tools. Automated test tools overcome the limitations of manual testing.

Simple testing like webpage can be opened in Internet explorer or not this can be done with manual testing very easily. But web–server can take the load or not then it is impossible to test manually. There are various software and hardware tools are available which helps tester in checking load testing, stress testing, regression testing.

❑ **Check Your Progress – 2 :**

1. Write a note on Manual VS Automated Testing.

_____

_____

_____

_____

_____

8.4 Testing Approaches :

There are two approaches with which tests can be conducted which are as follows :

1. **Functionality testing**
2. **Implementation testing**

In functionality testing, functionality is tested but implementation in not considering it is known as black–box testing. On the other hand, not only functionality is tested but implementation is also analyzed it is known as white–box testing. For a perfect testing complete tests are performed where every possible value of input and output is tested.

8.4.1 Black–Box Testing :

Black box testing involves looking at the specifications and does not require examining the code of a program. Black box testing is done from customer's viewpoint. Black box testing is the testing where inputs and outputs are defined but process is not defined.

Black box test is convenient to administer because they use the complete finished product and do not require any knowledge of its construction.

- **Why Black Box Testing :**

Black box testing helps in the overall functionality verification of the system under test.

- **Black box testing is done based on requirements :** It helps in identifying any incomplete, inconsistent requirement as well as any issues involved when the system is tested as a complete entity.
- **Black box testing addresses the stated requirements as well as implied requirements :** Not all requirements are stated clearly, but are considered implicit. For example, inclusion of dates, page header, and footer may not be clearly stated in the report generation requirements specification.
- **Black box testing includes the end user viewpoint :** Since we want to test the behavior of a product from an external viewpoint, end–user views are an important part of black box testing.
- **Black box testing handles valid and invalid inputs :** It is a natural for users to make errors while using a product. Hence, it is not sufficient for black box testing to simply handle valid inputs. Testing from end user view includes testing for these error or invalid conditions. This ensures that product behaves as expected in a valid situation and does not hang or crush when provided with an invalid input. These are called positive or negative test cases.

- **When to do Black Box Testing**

- ✓ Black box testing activities require involvement of the testing team from the beginning of the software project life cycle, regardless of the software development life cycle model chosen for the project.
- ✓ Testers can get involved right from the requirements gathering and analysis phase for the system under test.
- ✓ Test scenarios and test data are prepared during the test construction phase of the test life cycle, when the software is in the design phase.
- ✓ Once the code is ready and delivered for testing, test execution can be done.
- ✓ All the test scenarios developed during the construction phase are executed.

8.4.2 White–Box Testing :

White box testing is a way of testing the external functionality of the code by examining and the testing the program code that realizes the external functionality. This is also known as *clear box, or glass box or open box* testing.

White box testing is the testing where process is defined but input and output are not defined. White box testing considers the code, structure, flow of internal design of program. A number of defects come about because of incorrect translation of requirements and design into program code.

Some other defects are created by programming errors and programming language characteristics.

- **Static Testing :**

Static testing is type of testing which requires only the source code of the product, not the binaries or executables. Static tasting does not involve executing programs on computers but involves select people going through the code to find out whether

- ✓ The code works according to the functional requirement;
- ✓ The code has been written in accordance with the design development in the project life cycle;
- ✓ The code for any functionality has been missed out;
- ✓ The code handles errors properly.

o Static testing by Humans :

These methods rely on the principle of humans reading the program code to detect errors rather than computer executing the code to find errors.

This process has several advantages.

- ✓ Sometimes humans can find errors that computers cannot.
- ✓ By making multiple humans read and evaluate the program, we can get multiple views and therefore have more problems identified honest than a computer could.
- ✓ A human evaluation of the code can compare it against the specifications or design and thus ensure that it does what is intended to do.
- ✓ A human evaluation can detect many problems at one go and can even try to identify the root causes of the problems.

There are multiple methods to achieve static testing by humans which are as follows :

1. **Desk checking of the code :** Desk checking normally done manually by the author of the code, desk checking is a method to verify the portions of the code for correctness. Such verification is done by comparing the code with the design or specifications to make sure that the code does what it is supposed to do and effectively.
2. **Code walkthrough :** This method and formal inspection are group–oriented methods. Walkthroughs are less formal than inspections. The line drawn in formalism between walkthrough and inspections is very thin and varies from organization to organization.
3. **Code review / Code inspection :** Code review – also called Fagan Inspection – is a method, normally with a high degree of formalism.

The focus of this method is to detect all faults, violations, and other side–effects.

❑ **Check Your Progress – 3 :**

1. _____ is the testing where inputs and outputs are defined but process is not defined.

   a. White Box Testing b. Black Box Testing

   c. Unit Testing d. System Testing

2. _____ is the testing where process is defined but input and output are not defined.

   a. White Box Testing b. Black Box Testing

   c. Unit Testing d. System Testing

3. Explain white box testing in detail.

8.5 Testing Levels :

As the software development is carry on parallelly testing procedure is also performed so the current phase is tested, validated and verified before the move to the next phase, so testing is performed at various levels of SDLC.

To ensure that there are no any hidden bugs in the software testing is perform separately. Software is tested on various levels which are as follows :

8.5.1 Unit Testing

This initial part of structural testing corresponds to some quick checks that a developer performs before exposing the code to more extensive code coverage testing or code complexity testing.

This can happen by several methods which are as follows :

1. Initially, the developers can perform certain clear tests, knowing the input variables and the corresponding expected output variables. This can be a quick test that checks out any clear mistakes.
2. For modules with complex logic or conditions, the developer can build a "debug version" of the product by putting intermediate print statement and making sure the program is passing through the right loops and iterations the right number of times.
3. Another approach to do the initial test is to run the product under a debugger or an Integrated Development Environment (IDE). These tools allow single stepping of instructions, setting break points at any function or instruction, and viewing the various system parameters or program variable values.

8.5.2 Integration Testing :

A system is made up of multiple components or modules that can comprise hardware and software. Integration is defined as the set of interactions

among components. Testing the interaction between the modules and interaction with other systems externally is called integration testing.

Integration testing starts when two of the product components are available and end when all components' interfaces have been tested. The final round of integration involving all components is called Final Integration Testing (FIT), or system integration.

- **Integration Testing as A Type of Testing :**

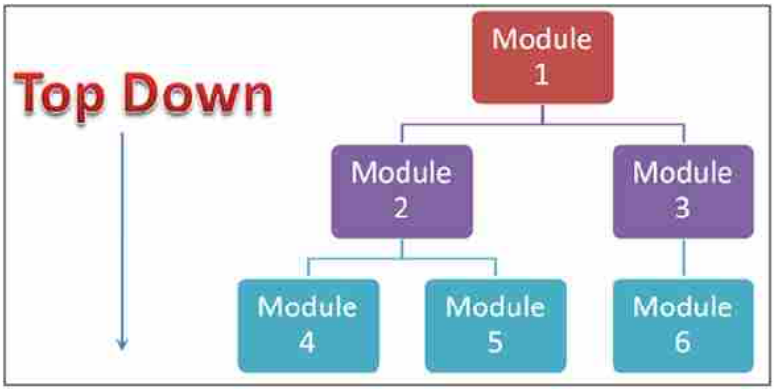
Integration testing means testing of interfaces. When we talk about interfaces, there are two types of interfaces that have to be kept in mind for proper integration testing. They are *internal interfaces* and *exported* or *external interfaces*.

Internal interfaces are those that provide communication across two modules within a project or product, internal to the product, and not exposed to the customer or external developers.

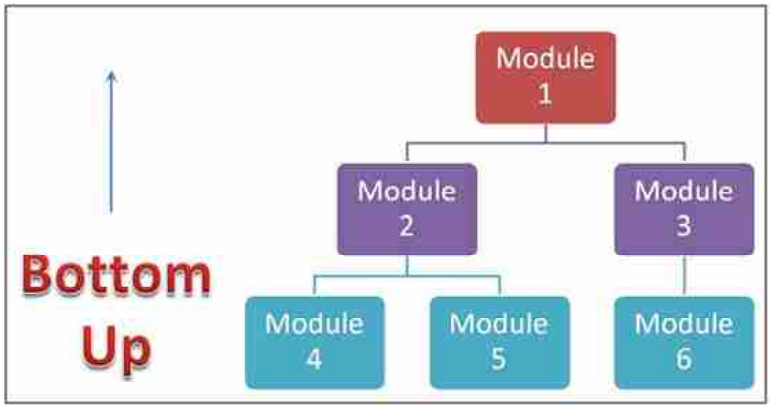
Exported interfaces are those that are visible outside the product to third party developers and solution providers.

There are several methods of integration testing which are as follows :

1. **Top–Down Integration :** Integration testing involves testing the topmost component interface with other components in same order as you navigate from top to bottom, till you cover all the components. In Top to down approach, testing takes place from top to down following the control flow of the software system.



2. **Bottom–Up Integration :** Bottom–up integration is just opposite of top–down integration, where the components for a new product development become available in reverse order, starting from the bottom. In the bottom–up strategy, each module at lower levels is tested with higher modules until all modules are tested. It takes help of Drivers for testing.



8.5.3 System Testing :

System testing is defined as a testing phase conducted on the whole integrated system, to assess the system arrangement with its specified requirements. It is done after unit, component, and integration testing phases.

A system is complete set of integrated components that together deliver product functionality and features. A system can also be defined as a set of hardware, software and other parts that together provide product features and solutions.

System testing is the only phase of testing which tests the both *functional* and *non–functional* aspects of the product. On the functional side, system testing focuses on real–life customer usage of the product and solutions. On the non–functional side, system brings in different testing types also called quality factors, some of which are as follows :

1. **Performance/Load testing :** To evaluate the time taken or response time of the system to perform its required functions in comparison with different versions of same product or a different competitive product is called performance testing.
2. **Scalability testing :** A testing that requires huge amount of resource to find out the maximum capability of the system parameters is called scalability testing.
3. **Reliability testing :** To evaluate the ability of the system to perform its function repeatedly for a specified period of time is called reliability testing.
4. **Stress testing :** Evaluating a system beyond the limits of the specified requirement to ensure the system does not break down unexpectedly is called stress testing.
5. **Interoperability testing :** This testing is done to ensure that two or more products can exchange information, use the information, and work closely.
6. **Localization testing :** Testing conducted to verify that the localized product works in different languages is called localization testing.

o Why is System Testing Done ?

An independent test team normally does system testing. The behavior of the complete product is verified during system testing. Tests that refer to multiple modules, programs, and functionality are included in system testing.

System testing helps in identifying as many defects as possible before the customer finds them in deployment. System testing is conducted with an objective to find product level defects and in building confidence before the product is released to the customer.

To summarize, system testing is done for the following reasons.

- ✓ Provide independent viewpoint in testing.
- ✓ Bring in customer perspective in testing.
- ✓ Provide a "fresh pair of eyes" to discover defects not found earlier by testing.
- ✓ Test product behavior in a complete and realistic environment.
- ✓ Test both functional and non–functional aspects of the product.

- ✓ Build confidence in the product.
- ✓ Analyze and reduce the risk of releasing the product.
- ✓ Ensure all requirements are met and ready the product for acceptance testing.

8.5.4 Acceptance Testing :

Acceptance testing is a phase after system testing that is normally done by the customers. A customer defines a set of test cases that will be executed to qualify and accept the product. These test cases are executed by customers themselves to quickly judge the quality of product before deciding to buy the product. Acceptance test cases failing in a customer site may cause the product to be rejected and may mean financial loss or may mean rework or product involving effort and time.

o Acceptance Criteria :

There are basically three acceptance criteria which are as follows :

1. Product Acceptance :

During the requirements phase, each requirement is associated with acceptance criteria. It is possible that one or more requirements may be mapped to form acceptance criteria. Whenever there are changes to requirements, the acceptance criteria are accordingly modified and maintained. Testing for faithfulness to any specific legal or contractual terms is included in the acceptance criteria.

2. Procedure Acceptance :

Acceptance criteria can be defined on the procedures followed for delivery. An example of procedure acceptance could be documentation and release media.

Some examples of acceptance criteria of this nature are as follows :

- ✓ User, administration and troubleshooting documentation should be part of the release.
- ✓ Along with binary code, the source code of the product with build script to be delivered in a CD.
- ✓ A minimum of 20 employees are trained on the product usage prior to deployment.
- – These procedural acceptance criteria are verified/tested as part of acceptance testing.

3. Service Level Agreements :

Service level agreements can become part of acceptance criteria and are generally part of a contract signed by the customer and product organization. The important contract items are taken and verified as part of acceptance testing.

For example, time limits to resolve those defects can be mentioned part of SLA such as

- ✓ All major defects that come up during first three months of deployment need to be fixed free of cost;
- ✓ Downtime of the implemented system should be less than 0.1%;
- ✓ All major defects are to be fixed within 48 hours of reporting.

8.5.5 Regression Testing :

- **What is Regression Testing ?**

Regression testing is a type of testing which ensures that a recent code change has not harmfully affected current features of the software. In this testing executed test cases re–executed full or partial to ensure existing functionalities work proper.

- **Types of Regression Testing :**

There are two types of regression testing which are as follows :

1. Regular Regression Testing :

A regular regression testing is done between test cycles to ensure that the defect fixes that are done and the functionality that were working with the earlier test cycle continue to work. A regular regression testing is performed to verify that the build has NOT broken any other parts of the application by the recent code changes for defect fixing or for enhancement.

2. Final Regression Testing :

A final regression testing is done to validate the final build before release. A "final regression testing" is performed to validate the build that hasn't changed for a period of time. This build is deployed or shipped to customers.

- **When to do Regression Testing ?**

Whenever changes happen to software, regression testing is done to ensure that these do no harmfully affect the existing functionality. Regression testing is done between test cycles to find out if the software delivered is as good as or better than the builds received in the past. As testing involves large number of resources, a quick testing is needed to assess the quality of build and changes to software.

It is necessary to perform regression testing when :

✓ A reasonable amount of initial testing is already carried out.

✓ A good number of defects have been fixed.

✓ Defect fixes that can produce side–effects are taken care of.

Regression testing may also be performed periodically, as a pro–active measure.

❑ **Check Your Progress – 4 :**

1. Write a detail note on Integration Testing.

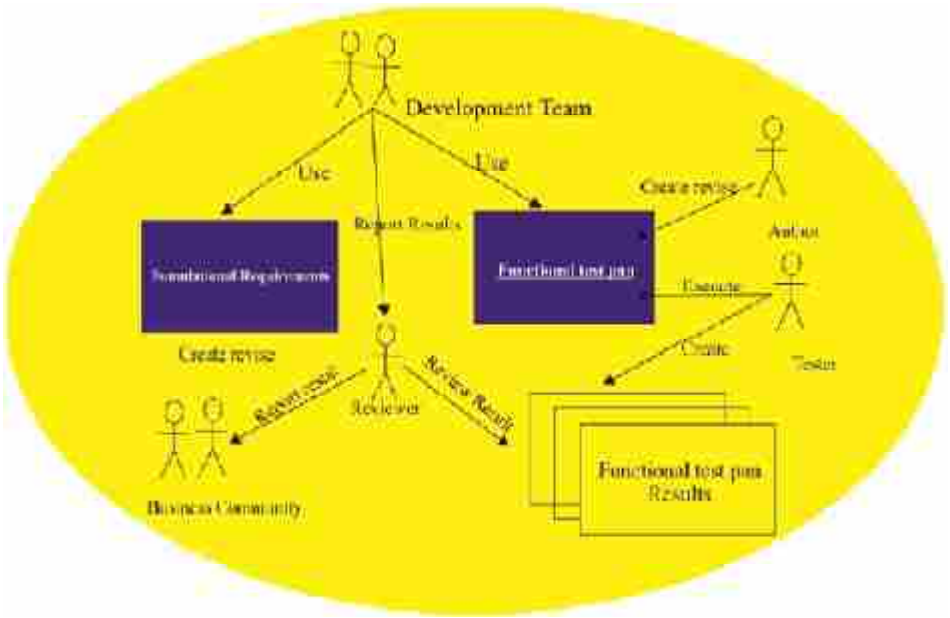
2. Write a detail note on System Testing.

3. Explain Acceptance Testing in detail.

4. Write a note on Regression Testing.

8.6 Function Test Plan :

By functional test plan we do not mean with testing of underlying implementation of application components, but it relates to testing of application from the customer's angle. It is related with how the application is meeting business requirements and specifications. In this, a functional test plan is prepared which serves as dynamic document that must regularly reflect continuous changing of business requirements. So, even when the document is completed, it will most likely need to change.



[Structure of Functional Test Plan]

We see that the participants interactions with functional test plan takes place. The developer or concern owner will create and revise test plan which will cover :

- Test cases
- Expected results
- Pre and posttest setup

In this, the tester will execute test plan with the use of actions which is highlighted in every test case and records as per the results in separate

document which serves as duplicate copy of function test plan with actual test results. The Reviewer compares actual test results with expected results and calculates whether the test case passed or failed the functional requirements.

The functional test plan consists of following sections :

- **Update History :** In the update history section, there appear lists names, dates and update descriptions of document.
- **Document Purpose :** It contains list of functions that are being tested by test plan. It should carry use–case diagram which give different perspective of functions that are to be examined.
- **Pre–test Setup :** The pre setup section carries instructions where test cases require persistence state to run through the steps. In this, the instructions will guide on how to run scripts to generate persistent data which is stored or created using another application.
- **Test Cases :** It carries list of test cases in which each test case verifies set of clear–defined functions of use–case. It will check for valid user's accessibility to the system and also declares that valid users cannot access system without entering valid password.

1. Functional Test Plan Scope	
In Scope	**Out of Scope**
In Scope *List functions that are tested.*	**Out of Scope** *List functions that are not tested.*

1. Functional Test Plan Assumptions and Constraints
Functional Test Plan Assumptions
Assumptions *List the functional test plan assumptions.*

Functional Test Plan Constraints
Constraint *List the functional test plan constraints.*

1. Functional Test Team Roles & Responsibilities		
Name	**Roles**	**Responsibilities**
Name *List name of people involved in functional testing.*		
Name *Add more rows if needed.*		

1. Functional Test Entry Criteria	
ID	**Criteria**
4.1	**Entry Criteria** *factors that must be present to enable the start of the functional test. Example: Testing environment/ data is available.*

1. Functional Test Cases	
ID	**Test Cases**
5.1	**Test Cases** *Identify the test cases along with expected results.* *Example:* *Test Procedure:* *Login with a corporate user account.* *Username: abc* *Password: abc* *Expected Results:* *An error will be displayed for the wrong credentials.*

1. Functional Test Results				
ID	**Test Cases**	**Pass/Fail**	**Tested By**	**Date Tested**
6.1	**Test Cases** *Name the test case.* *Example:* *Test Procedure:* *Login with a corporate user account.* *Username: abc* *Password: abc* *Expected Results:* *An error will be displayed for the wrong credentials.*			mm/dd/yyyy
	Test Case *Add more rows if needed.*			

[Layout of Function Test Plan]

The layout of the function test plan is shown in above figure. It is seen that functional test planning is a process and plan which serves as good way of communication with members of project team, testers, peers, managers and stakeholders. Such type of communication will allow test plan to manipulate project team and influence of project team on test plan mostly in areas of organization wide testing policies and motivations, test scope, objectives and testing areas, project and product risks, resource considerations and constraints along with testability of item under test.

Further the functional test plan will help in managing changes in information. As the project evolves and situations change, we adapt our plans. By updating the plan at major milestone helps us to keep testing aligned with project needs. As we run the tests, we make final adjustments to our plans based on the results.

❑ Check Your Progress – 5 :

1. Write a note on Function Test Plan.

8.7 Process of Testing :

Software testing strategy is a road map which is used and followed by developing team in order to maintain quality and serves as a bridge between customer road map which shows steps to be conducted at the time of testing. The customer road map carries steps that are planned which can be undertaken and judged in terms of effort and resources. Hence any testing strategy should use test planning, test case design, test execution and resultant data collection and evaluation.

The Fundamental Test Process comprises five activities :

- **Planning**
- **Specification**
- **Execution**
- **Recording**
- **Checking for Test Completion**

1. Test Planning :

The initial idea is to have a good plan. All good testing is based upon good test planning. For this, an overall test strategy should be planned along with project test plan. The Test Planning activity will generate test plan which is specific to level of testing. Such test level will specifically test plans which shows how test strategy and project test plan need to be applied to that level of testing and state any exceptions to them.

2. Test Specification :

In this, the fundamental test process shows activity related to as designing of test cases with certain techniques which to be followed at the time of planning. It is noted that for every test case, there should be specific objective with initial state of software along with input sequence and expected outcome. In this, specification can be considered as separate tasks :

- Identify test conditions
- Design test cases
- Build test cases

3. Test Execution :

The idea of this is to work with every test case which can be done either manually or with test execution automation tool. Here the way in which test cases are executed is significant where most important test cases should get executed first. Generally, important test cases find serious faults and may also those that stress on most important parts of system.

4. Recording :

Recording of test is done in parallel with Test Execution. For this, we need to record versions of software under test along with specification and for each test case, the outcome is recorded along with test coverage levels.

In this, the current situation should be compared with expected and any discrepancy found should be noted and logged and further analyzed to establish position of fault. The fault may lie in the environment set–up or be the result of using the wrong version of software under test.

5. Checking for Completion :

This will check the records against completion criteria which gets laid off for test plan. On failing of such criteria requires previous specification stage in order to specify test cases to meet completion features.

❑ **Check Your Progress – 6 :**

1. Write a note on Process of Testing.

8.8 Testing Documentation :

Testing documents are prepared at different phases which are as follows :

8.8.1 Before Testing :

Before testing is starts with preparation of test cases, it requires documents for reference which are as follows :

- **SRS document –** It is a document which define Functional Requirements of the software.
- **Test Policy document –** This defines how far testing should take place before releasing the product.
- **Test Strategy document –** It refers to detail characteristics of test team, responsibility matrix and rights/responsibility of test manager and test engineer.
- **Traceability Matrix document –** This is SDLC document, which is related to requirement gathering process. As new requirements come, they are added to this matrix.

8.8.2 While Being Tested :

The following documents may be required while testing is started and is being done :

- **Test Case document –** It is referring to list of tests required to be conducted. It contains Unit test plan, Integration test plan, System test plan and Acceptance test plan.
- **Test description –** It is referred to detailed description of all test cases and procedures to execute them.
- **Test case report –** It is referred to document that contains test case report as a result of the test.
- **Test logs –** It is referred to document contains test logs for every test case report.

8.8.3 After Testing

The following documents may be generated after testing :

- **Test summary –** It is referred to collective analysis of all test reports and logs. It summarizes and completes if the software is ready to be launched. The software is released under version control system if it is ready to launch.

❑ **Check Your Progress – 7 :**

1. Write a note on Testing Documentation.

8.9 Grey Box Testing :

Grey Box Testing is software testing method that results from mixture of Black Box Testing and White Box Testing methods. It is seen that in case of Black Box Testing, internal structure of item that gets tested is known to the tester while in White Box Testing, internal structure in known. The testing using Grey Box shows that the internal structure is partially known which gives access to internal data structures and algorithms with the idea to design test cases with testing at user or black box level.

Since the software program is like semitransparent impression to the tester, so this type of software testing is known as Grey Box Testing. It is noted that when codes for two units are studied for designing test cases where actual tests get carried out with exposed interfaces, then such testing procedure is called to be Grey Box Testing.

There are certain advantages of Grey Box testing :

✓ It allows tester to know about the functionality of application along with its architecture and data flow.

✓ It lowers the overhead of long drawn functional and non–functional types of testing cycles.

✓ It allows the developer to fix errors because of the ability of speeding up of development cycle with complete rudimentary unit testing.

❑ **Check Your Progress – 8 :**

1. What is Grey Box Testing ?

   a. Black–Box Testing

   b. Acceptance Testing

   c. Combination of Black Box and White Box Testing

   d. Stress Testing

8.10 Non–Functional Testing :

Non–functional testing in software is related with non–functional requirements and is mainly designed to find the readiness of system based on certain criteria that are not included by functional testing. It is seen that non–functional testing allow to measure and compare testing results of non–functional attributes of software systems. It can be seen as testing of an application or system against client's requirement or performance. There are certain examples of non–functional testing which are :

- Performance
- Security

- Usability
- Reliability and Dependability
- Endurance
- Localization and Internationalization
- Ergonomics
- Installation and Configuration
- Compatibility and Interoperability
- Maintainability and Availability

[Non–Functional Testing Example]

❑ **Check Your Progress – 9 :**

1. What is included in non–function testing ?

   a. Performance b. Testing c. Endurance d. All of Above

8.11 Testing Artifacts :

Test Artifacts is a scenario where a product what has developed in various phases of software testing life cycle being shared with owner/user. Such artifacts get prepared by software test team which takes final sign off from owner/user just to make sure that there exists no communication gap among customer and testing team members. These artifacts help in tracking the changes which occurs during the testing process.

There are certain deliverable test artifacts such as :

- **Test Strategy :** It is a static document which is written by higher officials for testing team in order to write objectives of test stages and techniques which they have used or applied. Such strategies form the basis for creation of standard document set and explain communication about test processes.
- **Test Plan :** It is form of document which explains about the scope, approach, resources and schedule of required test activities. Such type of document is normally prepared by Test Lead that identifies test items, features to be tested, testing tasks, resources and necessary risks factors which could occur in the process. The idea of this plan is share information to owner/user about the scope, responsibilities, timeline and deliverables for project under test.
- **Test Scenario :** Test scenario appears to be a small statement which verifies particular area of application and is prepared after reviewing

functional needs. It is noted that a particular area or module of application can have one or many scenarios based on complexity.

- **Test Case :** It is a single set of conditions which a tester follows in order to check whether the feature or function of software what is created is as per business requirement or not. The test case can be written by two ways which help the tester to verify functionalities of object inside software application.
  - ✓ Positive Test Cases are written on system to check the application behavior on positive test data input.
  - ✓ Negative Test Cases are written on system to check the system behavior on negative test data input.
- **Traceability Matrix :** This is also a document which is prepared and maintained in table format that correlates with two documents and shows relationship that exists among them. It is a part of software testing which is normally applied to link between detailed requirements and test design with end product as Requirement Traceability Matrix which checks whether all present project requirements are covered by designed test cases or not.
- **Test Report :** Test report shows the testing results which is defined in Test Plan. The test report gives information to customers about stability of current development so that they can redesign or implement immediate changes depending on the defects.

❑ **Check Your Progress – 10 :**

1. What are the test artifacts ?

   a. Test Plan b. Test Strategy c. Test Scenario d. All of Above

8.12 Let Us Sum Up :

In this unit we have learnt that software testing involves evaluation of software item which will find differences among given input and expected output. It is seen that testing of software be carried out at beginning of development that helps in reducing overhead cost and consumes less time in rework to generate error–free product.

It is noted that functional test plan relates to how application is meeting business requirements and specifications which is prepared as dynamic document that reflect continuous changing of business requirements.

It is found that software testing strategy provides a road map for the software developer, the quality assurance organization, and the customer– a road map that describes the steps to be conducted as part of testing, when these steps are planned and then undertaken, and how much effort, time, and resources will be required.

8.13 Answers for Check Your Progress :

❑ **Check Your Progress 1 :**

1. (b) 2. (a) 3. (c)

❑ **Check Your Progress 2 :**

1. (refer 8.3)

❑ **Check Your Progress 3 :**

1. (b) 2. (a) 3. (refer 8.4.2)

❑ **Check Your Progress 4 :**

1. (refer 8.5.2) 2. (refer 8.5.3)
3. (refer 8.5.4) 4. (refer 8.5.5)

❑ **Check Your Progress 5 :**

1. (refer 8.6)

❑ **Check Your Progress 6 :**

1. (refer 8.7)

❑ **Check Your Progress 7 :**

1. (refer 8.8)

❑ **Check Your Progress 8 :**

1. (c)

❑ **Check Your Progress 9 :**

1. (d)

❑ **Check Your Progress 10 :**

1. (d)

8.14 Glossary :

1. **Software Quality** – Activities that depends on external and internal quality factors.
2. **Verification** – To cross check or very document before start of software project or development.
3. **Quality Control** – it is measure related to quality standard of software.
4. **Testing** – It is related to software testing that enables the programmer to use necessary specification to develop a project.

8.15 Assignment :

1. Write a detailed note on Testing Artifacts.

8.16 Activities :

1. Give detailed analysis of Non–Functional Testing.

8.17 Case Study :

Concept related to Software Testing.

8.18 Further Reading :

1. Braude, Formal Inspections in Software Quality Assurance, 1998
2. Beck, A.C.; Mattos, J.C.B.; Wagner, F.R.; Carro, L. CACOPS : General Purpose Configurable Power Simulator, 2003.

BLOCK SUMMARY :

In this block, we have learnt about the basic of software requirement. Software requirement play an important role in software development life cycle. Detail about the basic of software design, software design strategies, software user interface design, software design complexity and software implementation which include all criteria of design from basic to advance as design play an important role.

We discussed about the basic of software quality and idea about different variables associated with it. the basic of testing of software techniques along with detailed explanation on Capability Maturity Model. The concept related to Review and Walkthrough with respect to software configuration management are well detailed. You will be demonstrated about User acceptance testing and its standards.

User acceptance testing is final phase of software testing process where normally software users perform testing of software so as to handle the required work as per specifications. In software development testing the software and quality management are important aspects. Performance testing is a non–functional technique which is done in order to locate the system parameters which could be related to responsiveness and stability under different workload.

BLOCK ASSIGNMENT :

❖ **Short Questions :**

1. Write a note on Requirement Engineering.
2. Explain Software Design Level.
3. Write a note on Coupling and Cohesion.
4. Explain Command Line Interface.
5. Explain golden rules of User Interface.
6. Write a note on Software Quality.
7. Write a note on CMM.
8. Write a note on Software Validation & Verification.
9. Explain Integration Testing.
10. Explain System Testing.
11. Explain Acceptance Testing with its criteria.
12. Write a note on Regression Testing.
13. Write a note on Process of Testing.

❖ **Long Questions :**

1. Write a detailed note on Feasibility Study.
2. Explain Requirement Initiation Technique.
3. Explain role and attributes of good system Analyst.
4. Explain Graphical User Interface in detail.
5. Explain Software Implementation.
6. Explain Software Quality Metrics and Parameters.
7. Explain Black Box and White Box Testing.
8. Explain Function Test plan in detail.

❖ **Enrolment No. :** ☐

1. How many hours did you need for studying the units ?

Unit No.	1	2	3	4
No. of Hrs.				

2. Please give your reactions to the following items based on your reading of the block :

Items	Excellent	Very Good	Good	Poor	Give specific example if any
Presentation Quality	☐	☐	☐	☐	_____
Language and Style	☐	☐	☐	☐	_____
Illustration used (Diagram, tables etc)	☐	☐	☐	☐	_____
Conceptual Clarity	☐	☐	☐	☐	_____
Check your progress Quest	☐	☐	☐	☐	_____
Feed back to CYP Question	☐	☐	☐	☐	_____

3. Any other Comments

.....

.....

.....

.....

.....

.....

.....

.....

Dr. Babasaheb Ambedkar
Open University Ahmedabad

BCAR-402

Software Engineering

BLOCK 3 : SOFTWARE RISK ANALYSIS & MANAGEMENT

UNIT 9 SOFTWARE RISK ANALYSIS

UNIT 10 SOFTWARE RISK MANAGEMENT – I

UNIT 11 SOFTWARE RISK MANAGEMENT – II

SOFTWARE RISK ANALYSIS & MANAGEMENT

Block Introduction :

Risk is expected in business organization at the time of taking projects. It is seen that project manager requires to make sure that risks are kept to lower level. Contingency Planning defines function of Risk Management on certain project. It is required in order to cater in case when initial level of risk tackling strategy fails to succeed.

In this block, we will detail about the basic of risk involved in companies and role of HR professional in minimizing risk. The block will focus on the study and concept of Software Risk analysis along with software project development process. You will give an idea on how human resources will minimize risk during software projects.

In this block, you will make to learn and understand about the basic of Contingency Planning techniques. The concept related to risk identification and minimization of risk in SDLC are well explained to you. You will be demonstrated practically about various phases in which the development occurs.

Block Objectives :

After learning this block, you will be able to understand :

- Risk Analysis in Project Management
- Detail of Risk Identification
- Idea about Qualitative Risk Analysis
- Idea about Quantitative Risk Analysis
- About Software Risk Management
- Basic of Risk Identification
- Idea about Analysis and Planning
- Features of Contingency Plans
- Concept about Human Resource and Risk Management
- Generalization of HR Executive and Risk Control

Block Structure :

Unit 9 : Software Risk Analysis

Unit 10 : Software Risk Management – I

Unit 11 : Software Risk Management – II

Unit

09 SOFTWARE RISK ANALYSIS

UNIT STRUCTURE

9.0 Learning Objectives

9.1 Introduction

9.2 Risk Analysis in Project Management

9.3 Risk Identification

9.4 Qualitative Risk Analysis

9.5 Quantitative Risk Analysis

9.6 Let Us Sum Up

9.7 Answers for Check Your Progress

9.8 Glossary

9.9 Assignment

9.10 Activities

9.11 Case Study

9.12 Further Readings

9.0 Learning Objectives :

After learning this unit, you will be able to understand :

- Risk Analysis in Project Management
- Identification of Risk
- Analysis of Qualitative Risk
- Analysis of Quantitative Risk

9.1 Introduction :

There is two parts of Risk : chance of something going wrong, and the negative interpretations if it does.

It is hard to identify the risk however, let alone to prepare for and manage. And, if you are success by an importance that you had not planned for, costs, time, and reputations could be on the line. Similarly, overestimating or overreacting to risks can create panic, and do more harm than good.

This makes Risk Analysis an important tool. It can help you to identify and understand the risks that you could face in your duty. In turn, this helps you to manage the risks, and reduce their effect on your plans.

❑ Check Your Progress – 1 :

1. What is Risk ?

9.2 Risk Analysis in Project Management :

It is referring to a procedure that helps you to identify and manage possible problems that could challenge key business creativities or projects. However, it can also be applied to other projects outside of business, such as organizing events or even buying a home!

To carry on a Risk Analysis, you must find the possible threats, then estimate their likely impacts if they were to happen, and finally estimate the possibility that these threats will arrive.

Risk Analysis can be complex, because you need to draw on detailed information such as project plans, financial data, security protocols, marketing forecasts, and other relevant information. However, it is an important planning tool, and that can save time, money, and reputations.

- **When to use Risk Analysis ?**

Risk analysis is useful in many situations which are as follows :

- ✓ When you are planning projects, that help you to do in advance and deactivate possible problems.
- ✓ When you are deciding whether or not to move forward with a project.
- ✓ When you are improving safety and managing possible risks in the workplace.
- ✓ When you are preparing for events like technology failure, theft, staff sickness, or natural disasters.
- ✓ When you are planning for changes in your environment, like new competitors coming into the market, or changes to government policy.

- **How to use Risk Analysis ?**

To carry out a risk analysis, follow these steps :

1. Identify Threats :

The first step in Risk Analysis is to identify the existing and possible threats. These can come from various different sources. For Example,

- ✓ **Human –** There may be possible risk by human like Illness, death, injury, or other loss of a main individual.
- ✓ **Operational –** Interruption to supplies and operations, loss of access to important resources, or failures in distribution.
- ✓ **Reputational –** There may be loss of customer or employee confidence, or damage to market reputation.
- ✓ **Procedural –** Failures of accountability, internal systems, or controls, or from fraud.
- ✓ **Project –** Going over budget, taking too long on key tasks, or experiencing issues with product or service quality.
- ✓ **Financial –** Business failure, stock market fluctuations, interest rate changes, or non–availability of funding.
- ✓ **Technical –** Due to advances in technology, or from technical failure.
- ✓ **Natural –** Weather, natural disasters, or disease.
- ✓ **Political –** Changes in tax, public opinion, government policy, or foreign influence.

- ✓ **Structural** – Dangerous chemicals, poor lighting, falling boxes, or any situation where staff, products, or technology can be harmed.

To carry out detailed analysis you can use a number of different methods which are as follows :

- ✓ Create a list of threats as above and check any of these threats are relevant.
- ✓ Run through a list such as the one above to see if any of these threats are relevant.
- ✓ Think about the systems, processes, or structures that you use, and analyze risks to any part of these. What weaknesses can you spot within them ?
- ✓ Ask others who might have different views. If you are leading a team, ask for input from your people, and consult others in your organization, or those who have run similar projects.

2. Estimate Risk :

Once you have recognized the threats, you need to calculate both the probability of these threats being understood, and their possible impact.

One way of doing this is to make estimate of the probability of the event occurring, and then to multiply this by the amount it will cost you to set things right if it happens. This gives you a value for the risk :

Risk Value = Probability of Event x Cost of Event

As a simple example, imagine that you've identified a risk that your rent may increase substantially.

You think that there's an 80 percent chance of this happening within the next year, because your landlord has recently increased rents for other businesses. If this happens, it will cost your business an extra 500,000 over the next year.

So, the risk value of the rent increase is :

0.80 (Probability of Event) x 500,000 (Cost of Event) = 400,000 (Risk Value)

❑ **Check Your Progress – 2 :**

1. The first step in Risk Analysis is to identify the existing and possible _____.

   a. Risk b. Treats c. Risk Analysis d. Risk Probability

2. Threats can come from _____ sources.

   a. Human b. Financial c. Technical d. All of Above

3. What is Risk Analysis in Project Management ?

_____

_____

_____

_____

_____

9.3 Risk Identification :

Risk identification is the process of determining risks that could potentially prevent the program, enterprise, or investment from achieving its objectives. It includes documenting and communicating the concern.

The objective of risk identification is the early and continuous identification of events that, if they occur, will have negative impacts on the project's ability to achieve performance or capability outcome goals. They may come from within the project or from external sources.

There are multiple sources of risk. For risk identification, the project team should review the program scope, cost estimates, schedule (to include evaluation of the critical path), technical maturity, key performance parameters, performance challenges, stakeholder expectations vs. current plan, external and internal dependencies, implementation challenges, integration, interoperability, supportability, supply–chain vulnerabilities, ability to handle threats, cost deviations, test event expectations, safety, security, and more.

In addition, historical data from similar projects, stakeholder interviews, and risk lists provide valuable insight into areas for consideration of risk.

Risk identification is an iterative process. As the program progresses, more information will be gained about the program (e.g., specific design), and the risk statement will be adjusted to reflect the current understanding. New risks will be identified as the project progresses through the life cycle.

Risk identification enables businesses to develop plans to minimize harmful events before they arise. The objective of this step is to identify all possible risks that could harm company operations, such as lawsuits, theft, technology breaches, business downturns, or even a Category 5 hurricane.

Safety management professionals must understand that risk identification is not a one–time process. Instead, the process should be rigorous, thoughtful, and ongoing.

- **Ways to Identify Risks :**

There are many ways to identify an organization's risks, however, some of the more common examples include brainstorming, thinking pessimistically, and seeking employee feedback.

1. **Brainstorming :** Risk managers may find that brainstorming the probability of various catastrophic events with other company stakeholders, such as managers and certain C–level staff, can help identify new threats.
2. **Thinking Negatively :** Careers in safety management often entail planning for the worst while expecting the best. Although doubt isn't often encouraged in the workplace, taking time to ponder "what is the worst possible thing that could happen to the company" may be helpful in identifying risks.
3. **Seek Employee Feedback :** Upper–level management's perspective of an organization's risks can be starkly different from the perspective that employees hold. Employees may encounter new risks in their day–to–day activities that may not have otherwise been encountered. For example, insufficient training on a piece of operating equipment may be placing staff at risk of injury. As such, employees are an invaluable source of first–hand information.

❑ **Check Your Progress – 3 :**

1. What are the ways to identify risk ?

   a. Brainstorming b. Thinking Negatively

   c. Seek Employee Feedback d. All of Above

9.4 Qualitative Risk Analysis :

It is referring to the procedure of evaluating the probability of a risk occurring and the impact it would have on a project if it happened.

- **Types of Analysis :**

As there is different demand of the project in the market there is various types of qualitative risk analysis. Employee experience and available resource are useful into the decision of identifying the project's risk.

There are five types of qualitative risk analysis which are as follows :

1. **Probability Environment**
2. **Bow–Tie Analysis**
3. **Delphi Method**
4. **SWIFT Analysis**
5. **Pareto Principle**

1. Probability Environment :

Probability is the standard process of finding risk seriousness. Risk is often different in size, but it is necessary to take action against risk. It gives practical method to give priority to the seriousness by considering the occurrence and compared to the impact of the risk.

By risk priority you will define major things from where risk is defined. This helps you in identifying how to handle the risk based on its projecting drivers.

2. Bow–tie Analysis :

Bow–tie analysis starts by considering risk event and it is going to plan in two directions. On t =he left side all the reasons of possible event are going to list and on right side all consequences of possible event are going to list. So, this is the most practical method to identify the risk reasons.

3. Delphi Method :

In Delphi method experts reply to number of rounds of questionnaires and these responses are combined and shared with the group after each round.

This method is used to identify risk and impact of the risk in risk management. The experts are asked to form an opinion on how expected the risk is to occur, and the consequence of its occurrence. These responses are combined and reviewed by the experts until a solution is achieved.

4. SWIFT Analysis :

SWIFT stands for "Structured What–If Technique", it is a systematic, team–based method for identifying risk analysis. Teams investigate how changes from an approved plan, may affect a project through a series of "What if" considerations.

5. Pareto Principle :

It is well known as "80/20 Rule", it is helps in defining risk which is most effective. It is considered as 8020 because 80% is based on initiate understood success and 20% based on efforts.

Risk managers use Pareto analysis as a tool for fast finding the most critical 20% of risks that will effectively mitigate 80% of the impact.

The task for risk managers knows how to successfully cut each risk. Large projects may require multi–attribute increments for business different priorities like security data, and operational or agreement policies.

But once you get the idea about where to look and what to look at will help you to improve in on the most important 20%. This offers a crucial leg up in managing the threats and weaknesses that have the potential to have the largest impact.

- **Qualitative Risk Analysis Process :**

When you are starting to identifying the risk at that time you are blank you have no ideas about how to start to define the risk, so at that time qualitative risk analysis is best method which is divide into smaller steps which are as follows :

1. **Finding Risks**
2. **Effect Analysis**
3. **Risk Handling**
4. **Review & Monitor**

1. Finding Risks :

Finding risk is the possibly most important part of qualitative risk analysis. If you will not find the risk on time then it becomes extremely challenging to manage them.

There is a simple trick to find risk, start thinking of the things which gives undefined effect on the project. Taking in to consideration by clear risk will help you to solve the risk in more linear way. Risk finding is all about quantity. So, reach out to as many people as you can to get a wide range of views.

There are various tools for Risk Identification which are as follows :

- ✓ Awareness plans
- ✓ Questionnaires
- ✓ Interviews
- ✓ Documentation review
- ✓ Checklist investigation
- ✓ SWOT Investigation

2. Effect Analysis :

Once you find the possible risks, the next step is to study the impact of the risk.

- ✓ Distinct the risks into threats and opportunities.
- ✓ Using qualitative risk analysis, estimate the impact of each risk on a scale from 1 to 5 or low/medium/high/extreme.

- ✓ Next, estimate the possibility of each risk occurring, using a similar scale.
- ✓ At the end, take those scores and combine them to create a total risk ranking.

Easiness is the major benefit of qualitative risk analysis because there is no statistical model that depends deeply on the quality of the data you use.

3. Risk Handling :

The next phase in the qualitative risk analysis is to apply actions to each risk. This can be processed in many numbers of ways depending on your industry or process. A simple example could show five options when it comes to risk action, but these are by no means final :

A. Accept

B. Reasonable

C. Exploit

D. Transfer

E. Avoid

A. Accept : If a risk has low effect and possibility, or the cost of stopping it is too high, sometimes it's more cost–effective to accept it.

B. Reasonable : Some risks have a high possibility, means you cannot avoid it. In order to reduce the impact of a risk when it becomes an issue, you can select practical way.

C. Exploit : Some risks can be exploited to the benefit of your project. Having the capacity to identify exploitable risks can be very useful and highlights the importance of seeking out experienced risk experts who can spot these chances.

D. Transfer : Risks with financial effects are a common example of risks that can be transferred to a third party. Insurance is designed to adopt a risk on your behalf, so you don't suffer as hard an effect if something goes wrong. Similarly, it is possible to transfer risk via a contract to a supplier or contractor.

E. Avoid : If you can't reasonable or transfer a risk, and that risk is too high to accept, the only way is to avoid it. Risks can be avoided by changing or removing certain possibility objects or changing the method.

Likelihood Planning :

If a risk becomes an issue, you need a plan, which are as follows :

- What to do
- Who come to be informed ?
- Who does what ?

Documenting a possibility plan saves time and money. When you know what to do in the event of an issue, you can reduce its effect by responding faster. The nature and detail of your possibility planning will depend on the nature of the risks themselves.

4. Review & Monitor :

Risk management is never ended, not even after the project has ended. As the project developments, it is important to keep up to date logs of risk.

At each phase of the project, risk possibility will vary. Some risks will disappear, while others might increase in possibility. Studying your risks regularly will help keep you on top of these changes.

After the project, a full reviewing is carry on which will provide important data and knowledge for future projects, making the next one more secure and helping to further your risk maturity.

❑ **Check Your Progress – 4 :**

1. The most common types of analysis are _____.

   a. Bow–Tie Analysis b. Delphi Technique

   c. SWIFT Analysis d. All of Above

2. _____ respond to several rounds of questionnaires.

   a. Delphi Technique b. Swift Analysis

   c. Probability d. Pareto Principle

9.5 Quantitative Risk Analysis :

Quantitative risk analysis is a numeric estimate of the overall effect of risk on the project objectives such as cost and schedule objectives. The results provide insight into the likelihood of project success and are used to develop contingency reserves.

- **Why Perform Quantitative Risk Analysis ?**
- **Better Overall Project Risk Analysis**

Individual risks are evaluated in the qualitative risk analysis. But the quantitative analysis allows us to evaluate the overall project risk from the individual risks plus other sources of risks.

- **Better Business Decisions :**

Business decisions are rarely made with all the information or data we desire. For more critical decisions, quantitative risk analysis provides more objective information and data than the qualitative analysis. Keep in mind : While the quantitative analysis is more objective, it is still an estimate. Wise project managers consider other factors in the decision–making process.

- **Better Estimates :**

A project manager estimated a project's duration at eight months with a cost of $300,000. The project actually took twelve months and cost $380,000. What happened ?

The project manager did a Work Breakdown Structure (WBS) and estimated the work. However, the project manager failed to consider the potential impact of the risks (good and bad) on the schedule and budget.

- **When to Perform Quantitative Risk Analysis :**

First, we identify risks. Then we can evaluate the risks qualitatively and quantitatively. Consider using Quantitative Risk Analysis for :

✓ Projects that require a Contingency Reserve for the schedule and budget.

✓ Large, complex projects that require Go/No Go decisions (the Go/No Go decision may occur multiple times in a project).

❑ **Check Your Progress – 5 :**

1. What is Quantitative Risk Analysis ?

9.6 Let Us Sum Up :

In this unit we have learnt that during the software development Planning defines function of Risk Management on certain project. We focused on the study and concept of Software Risk analysis along with software project development process. We have also seen about idea about the qualitative and quantitative risk analysis.

9.7 Answers for Check Your Progress :

❑ **Check Your Progress 1 :**

1. (refer 9.1)

❑ **Check Your Progress 2 :**

1. (b) 2. (d) 3. (refer 9.2)

❑ **Check Your Progress 3 :**

1. (d)

❑ **Check Your Progress 4 :**

1. (d) 2. (a)

❑ **Check Your Progress 5 :**

1. (refer 9.5)

9.8 Glossary :

1. **Risk :** Risk is made up of two parts : the probability of something going wrong, and the negative consequences if it does.
2. **Risk Analysis :** It is referring to a procedure that helps you to identify and manage possible problems that could challenge key business creativities or projects.
3. **Risk Identification :** Risk identification is the process of determining risks that could potentially prevent the program, enterprise, or investment from achieving its objectives.
4. **Qualitative Risk Analysis :** It is referring to the procedure of evaluating the probability of a risk occurring and the impact it would have on a project if it happened.
5. **Quantitative Risk Analysis :** Quantitative risk analysis is a numeric estimate of the overall effect of risk on the project objectives such as cost and schedule objectives.

9.9 Assignment :

1. Explain Risk Identification.

9.10 Activities :

1. Explain Qualitative Risk Analysis Process

9.11 Case Study :

Explain difference between qualitative and quantitative risk analysis.

9.12 Further Reading :

1. Bruegge, B. and A. H. Dutoit (2000). Object–Oriented Software Engineering : Conquering Complex and Changing Systems. Upper Saddle River, NJ, Prentice Hall.
2. Cockburn, A. (2001). Agile Software Development. Reading, Massachusetts, Addison Wesley Longman.
3. Gluch, D. P., "A Construct for Describing Software Development Risks," Software Engineering Institute, Pittsburgh, PA CMU/SEI–94–TR–14.

Unit 10

SOFTWARE RISK MANAGEMENT – I

UNIT STRUCTURE

10.0 Learning Objectives

10.1 Introduction

10.2 Software Risk Management Implementation

10.3 Planning Risk Responses

10.4 Monitoring and Controlling Risks

10.5 Let Us Sum Up

10.6 Answers for Check Your Progress

10.7 Glossary

10.8 Assignment

10.9 Activities

10.10 Case Study

10.11 Further Readings

10.0 Learning Objectives :

After learning this unit, you will be able to understand :

- Software Risk Management Implementation
- Planning Risk Responses
- Idea about Monitoring Risk
- Idea about Controlling Risks

10.1 Introduction :

This type of risk management shows conserve environment that is applied for taking practical decision making so as to analyses what goes wrong with development process. It shows the type of risks which is quite useful in dealing and which implements in such a way that will avoid risks.

Software risk management is a type of planning process which concerns with strategy that caters risk management and its process along with techniques using certain methods and tools which applies to support risk management process.

It is seen that risk management process normally requires corporate clearance about risk that serves as main consideration. It is found that in corporate organization, senior management supports project risk management activities through :

- ✓ Supplying of excess professionals, budget, schedules and tools
- ✓ Checking of required training which is to be implemented in finding, managing and lowering of software risk
- ✓ Giving training to project personnel in conducting risk management work.

❑ **Check Your Progress – 1 :**

1. What is Software Risk Management ?

10.2 Software Risk Management Implementation :

There are 9steps Software Risk Management Implementation which are as follows :

1. Define the end goal before starting :

It's impossible to begin any kind of project without a thorough understanding of where you're going. Doing so will lead to confusion, frustration, and wasted resources as the team moves in multiple directions at once without any noticeable results.

Since you've already gone through the process of selecting a risk management system, you know what issues need to be solved and where the system is needed. Formalize this knowledge by creating a document that defines exactly what your organization needs from the system and how this can be accomplished.

If you're going to use the risk management system in multiple areas, determine your priorities. These should be the areas with the most issues; highlighting these problems will allow the team to tackle them first.

In addition, define success for your risk management system. Are you aiming for a lower number of claims ? Would you like to see a reduction in costs ? Should your team reduce time spent on redundant tasks by 50% ? Whatever the goal, pre–defining success ensures you can measure the effectiveness of the system through implementation and going forward.

2. Set a timeline :

Implementing a risk management system is a complex process. It's important to understand exactly what is involved and what that means in terms of a timeline. The vendor and your team must find a balance : if an implementation is too quick, something may be missed; if the implementation takes too long, the team may lose faith in the system or become upset with the vendor.

Consider these stages in the implementation process :

- ✓ First, the risk management system must be set up. The vendor will need to import historical data and complete any necessary customization.
- ✓ The system must be tested to ensure it will work correctly throughout the organization.
- ✓ All users must be trained in the proper use of the system.

Project management is key when implementing a risk management system. Determine milestones that can be easily measured throughout the process to keep all stakeholders on track, and consider appointing a project champion who is responsible for seeing the implementation through.

3. Build a relationship with the vendor :

In many situations, the internal risk team views the vendor implementation team as external stakeholders who are only present for a few weeks or months. This is the wrong mindset. Risk management vendors have high levels of knowledge, insight, and resources that can help you manage both new and existing risks at any time.

By building a relationship with the vendor, you've widened your risk management network and increased the size of your risk management team. This can only benefit you as you seek to achieve your goals with the risk management system.

4. Be open to vendor suggestions :

Risk management systems are built a certain way for a reason. Vendors have extensive experience with the needs of organizations much like yours. You should always be open to their suggestions, especially if they're recommending a particular process.

Many teams fall into the trap of purchasing a risk management system only to use it in exactly the same way as their old system. For example, a team that switches from Excel spreadsheets may continue to manually add and report on data in the system, even when automation is possible. This mistake can be critical : the team continues to poorly utilize resources while extra resources are used to pay for the new system.

To avoid this problem, carefully consider all vendor suggestions on how their risk management system can truly improve your organization.

5. Customize where necessary :

While vendor suggestions and knowledge are valuable, sometimes they may not realistically fit into your organization or goals. Some aspects of an out–of–the–box system may not be right for you. In this case, some customization is ideal.

For example, consider your organization's hierarchy, the ideal usage of the system, and your reporting needs. Only you can determine exactly how a risk management system will best fit into these requirements.

6. Be flexible :

Adapting to changing circumstances is important when implementing a risk management system. Tasks may take more time than expected, there may be technical difficulties, or an employee may have a particularly hard time during training.

You must understand that difficulties like these are bound to happen and typically only involve a small adjustment. Being ready to re–prioritize or modify existing plans allows all stakeholders to feel comfortable through the implementation process, even if not everything goes as planned.

7. Involve users and decision–makers :

Another common mistake in the implementation of risk management systems is involving only decision–makers. While executives and top managers may be able to pick the system that best suits organizational goals, they aren't the ones that will be working inside the system every day.

Involving users from the beginning ensures that the entire risk team is onboard or even excited about the change. They can also provide valuable

insight into implementation : they may have needs or desires that decision–makers wouldn't know about and can reduce complications in the implementation process.

8. Communicate :

Any significant organizational change is likely to fail without regular and proper communication. When implementing a risk management system, there are two critical communication avenues : the vendor and employees.

No matter how robust their system, vendors cannot read your mind. You must explain your system, timeline, and security requirements as well as how involved you expect them to be in the implementation process. This will keep both teams on the same page and prevent frustrating back–and–forth conversation.

On the employee side, users need to be taught what to expect from the system. In some cases, users may feel that they are being replaced by the system; it is your job to reassure them that the system will actually make their jobs easier and more meaningful by streamlining complicated processes. Tell your employees what will change and how it will impact them individually, and make them aware of these changes well in advance. Educating them on the role they must play in the implementation of the risk management system will simplify the process.

9. Implement in stages :

While risk management systems often have extensive functionality, it can be overwhelming for a team to implement them all at once. This is frustrating to employees and can actually lower the chances of system success. Instead, choose the one area that is most in need of the system and start there. This allows the team to gradually become comfortable with the system and then expand their capabilities.

Using one small change as an example of the effectiveness of the system can also help win over resistant employees and prove that the system has value.

Risk management system implementation can seem like a daunting task. Following this advice will put you well on your way towards achieving your risk management goals.

❑ **Check Your Progress – 2 :**

1. _____ Step is not part of Software Risk Management Implementation.

   a. Set a timeline b. Be flexible

   c. Communicate d. Do not customize

2. Any significant organizational change is likely to fail without regular and proper _____.

   a. Communicate b. Implement in stages

   c. User Involvement d. All of Above

10.3 Planning Risk Responses :

Once the risks to the project have been identified, their probability and impact given a value, and an overall priority established, risk responses should be drawn up. For each response plan, trigger conditions should be identified.

- **When you need a Risk Response Plan :**

A proper risk management plan does not need to include response plans for all risks within the risk register. The risk register contains all risks that are significant enough to warrant tracking and monitoring. It is not feasible or necessary to develop response plans for everyone.

All risks fall within a range. On the one extreme it does not affect the project enough to warrant tracking and monitoring. In the middle, the event should be tracked and monitored but response plans do not need to be developed in advance. And on the other extreme, the risk is substantial enough that a response plan needs to be developed.

Generally, risk response plans are required for risks that are high in both probability and impact. For example, a nuclear power repair project might have response plans developed for radiation exposure events.

- **Four Risk Responses :**

There are four possible ways to deal with risk.

1. **Avoid :** Remove the threat or protect the project from its effect. Here is a list of common actions that can eliminate risks.
   - ✓ Change the scope of the project.
   - ✓ Extend the schedule to eliminate a risk to timely project completion.
   - ✓ Change project objectives.
   - ✓ Clarify requirements to eliminate ambiguities and misunderstandings.
   - ✓ Gain expertise to remove technical risks.
2. **Transfer :** This involves moving the impact of the risk to a third party. Direct methods might be through the use of insurance, warranties, or performance bonds. Indirect methods such as unit price contracts instead of lump sum, legal opinions, and so forth.
3. **Mitigation :** Reduce the probability or impact of the risk. This is not always possible and often comes with a price that must be balanced against the value of performing the mitigating action.
4. **Accept :** All projects contain risk. As a minimum, there is the risk that it does not accomplish its objective. Thus stakeholders, by definition, must accept certain risks. Accepting risk is a strategy like any other, and should be documented and communicated like any other strategy. Risk acceptance can be passive, whereby the consequences are dealt with after the risk occurs, or active, whereby contingencies (time, budget, etc.) are built in to allow for the consequences of the risk to the project.

The four risk response strategies can be applied to overall project risk as well.

❑ **Check Your Progress – 3 :**

1. _____ considered as a risk response.

   a. Avoid b. Transfer c. Mitigation d. All of Above

2. _____ is reduce the probability or impact of the risk.

   a. Mitigation b. Accept c. Transfer d. Avoid

10.4 Monitoring and Controlling Risks :

Project risk control and risk monitoring is where you keep track of about how your risk responses are performing against the plan as well as the place where new risks to the project are managed.

You must remember that risks can have negative and positive impacts. Positive risk is a risk taken by the project because its potential benefits outweigh the traditional approach and a negative risk is one that could negatively influence the cost of the project or its schedule.

The purpose of project risk control is to

- ✓ Identify the events that can have a direct effect in the project deliverables
- ✓ Assign qualitative and quantitative weight–the probability and consequences of those events that might affect the project deliverables
- ✓ Produce alternate paths of execution for events that are out of your control or cannot be mitigated
- ✓ Implement a continuous process for identifying, qualifying, quantifying, and responding to new risks

The main goals to risk monitoring and control :

- ✓ To confirm risk responses are implemented as planned
- ✓ To determine if risk responses are effective or if new responses are needed
- ✓ To determine the validity of the project assumptions
- ✓ To determine if risk exposure has changed, evolved, or declined due to trends in the project progression
- ✓ To confirm policies and procedures happen as planned
- ✓ To monitor the project for new risks
- ✓ To monitor risk triggers

Risk triggers are those events that will cause the threat of a risk to become a reality. For example, you have identified the fact that you only have one pump set available and the replacement takes six weeks to arrive. In the middle of your irrigation and recycling process tests, you discover that water pressure tends to fluctuate beyond pump tolerance levels. If you do not find a way to solve this problem, your risk will become a reality.

Make sure that for each identified risk, you must provide a response plan. It is not much help to you if the risk becomes a reality or issue and you do not have an alternate execution path or some other emergency procurement plan.

Main inputs to effectively monitor and control risks :

- ✓ Risk management plan
- ✓ Risk Register / Risk Tracker
- ✓ Risk response plan
- ✓ Project communications
- ✓ New risk identification
- ✓ Scope changes

Outputs of Risk Monitoring and Risk Control :

- ✓ Workaround plans
- ✓ Corrective / Preventive actions
- ✓ Change requests
- ✓ Risk response plan updates
- ✓ Risk database
- ✓ Checklist updates

❑ **Check Your Progress – 4 :**

1. _____ considered as an input to monitor and control risks.
   a. Scope Changes b. Risk Tracker
   c. New Risk Identification d. All of Above
2. _____ known as an output of risk monitoring and control.
   a. Change Request b. Risk Database
   c. Checklist updates d. All of Above

10.5 Let Us Sum Up :

In this unit we have learnt that Software risk management is concern with process orientation, methods and tools that helps in managing risks which appears in software project development process. We have also seen that there are various steps of Software Risk Management Implementation.

We also learnt about project risk control and risk monitoring is where you keep track of about how your risk responses are performing against the plan as well as the place where new risks to the project are managed. We have is seen that risk management process normally requires corporate clearance about risk that serves as main consideration.

10.6 Answers for Check Your Progress :

❑ **Check Your Progress 1 :**

1. (refer 10.1)

❑ **Check Your Progress 2 :**

1. (d) 2. (a)

❑ **Check Your Progress 3 :**

1. (d) 2. (a)

❑ **Check Your Progress 4 :**

1. (d) 2. (d)

10.7 Glossary :

1. **Software Risk Management** – Software risk management is concern with process orientation, methods and tools that helps in managing risks which appears in software project development process.
2. **Mitigation** – Reduce the probability or impact of the risk.

10.8 Assignment :

1. Explain 9 steps of Software Risk Management Implementation.

10.9 Activities :

1. Write a note on planning risk responses.

10.10 Case Study :

Explain about monitoring and controlling risks and also give difference.

10.11 Further Reading :

1. Bruegge, B. and A. H. Dutoit (2000). Object–Oriented Software Engineering : Conquering Complex and Changing Systems. Upper Saddle River, NJ, Prentice Hall.
2. Cockburn, A. (2001). Agile Software Development. Reading, Massachusetts, Addison Wesley Longman.
3. Gluch, D. P., "A Construct for Describing Software Development Risks," Software Engineering Institute, Pittsburgh, PA CMU/SEI–94–TR–14.

Unit 11

SOFTWARE RISK MANAGEMENT – II

UNIT STRUCTURE

11.0 Learning Objectives

11.1 Introduction

11.2 Human Resource and Risk Management

11.3 The HR Executive and Risk Control

11.4 Team Risk Management

11.5 Let Us Sum Up

11.6 Answers for Check Your Progress

11.7 Glossary

11.8 Assignment

11.9 Activities

11.10 Case Study

11.11 Further Readings

11.0 Learning Objectives :

After learning this unit, you will be able to understand :

- Software Risk Management
- Human Resource and Risk Management
- Idea about HR Executive
- Idea about Risk Control
- Detail of Team Risk Management

11.1 Introduction :

Risk is expected in business organization at the time of taking projects. It is seen that project manager requires making sure that risks are kept to lower level.

Risk management, in regards to human resources, doesn't stop once background checks, references and education confirmation is completed. The human resource department and the risk management department must continue to collaborate together to ensure employee related risks are continuously identified and strategies established for mitigation of identified risks.

Employee or human factors are some of the most critical sources of risk and extremely difficult to plan and prepare for. The human factors in regards to risk is very different from the risk introduced by machines or automated processes, as the human factors are highly dynamic and difficult to regulate in relationship to controls for machines and automated processes. After all the reason you are employing an individual is to allow her to work in an environment, which allows her to feel free to perform at her best, with a strong support and trust of management.

11.2 Human Resource and Risk Management :

Human resources have two roles in risk management. **First**, people are a source of risk, e.g., shortage of employees, people doing sloppy work, an employee refusing to take on additional responsibility, or a key employee leaving two months after completion of a one–year training program.

Second, people are important in handling risk, e.g., people using their ingenuity to solve unexpected problems, employees going the extra mile for the good of the organization, a key employee redesigning her own job to avoid unnecessary delays in getting work done, or an employee persuading a talented friend to apply for a position in the business.

Human resource management is a process that can be broken down into specific activities :

- **Job analysis, Writing job descriptions,**
- **Hiring,**
- **Orientation, Training,**
- **Employer/Employee interactions,**
- **Performance appraisal, Compensation, and Discipline.**

Understanding these activities helps explain the relationship between human resources and risk. Failure to successfully carry out these activities increases risk and penalizes the business by not taking advantage of what its people could be contributing.

The first activity is **job analysis** and **writing job descriptions**. Job analysis is 8 The emphasis is on what the farm needs rather than on who wants to be promoted or who could be easily hired. The tasks that must be carried out to accomplish the firm's goals determine duty and skill requirements. Job descriptions summarize for both employees and employers just what a job entails : job title, duties, compensation, and skills, knowledge, and abilities to do the job. In family farm businesses, job descriptions for family members often include both management and labor responsibilities.

Hiring is the next human resource management activity. The objective of hiring is to staff each job with a person who can succeed in the position. In today's exceptionally tight labor market, hiring is one of the most difficult human resource activities. The position must be described carefully and creatively to potential applicants. From among the pool of applicants, people must be carefully chosen if they and the employer are to have a successful relationship.

The next activity after hiring is **orientation** and **training**. Orientation socializes new people to the business. It introduces them to the business' mission, its history, and its culture. It gives them the information essential for getting off to a good start. Training and experience give the employees the knowledge, skills, and abilities necessary to succeed in the position.

Day–to–day **employer/employee interaction** includes leadership, motivation, and communication that build on hiring, orientation, and training. Employer/employee interaction cannot make up for an ill–defined job, having hired the "wrong" person, or inadequate orientation and training.

The last three activities are closely related : **performance appraisal, compensation,** and **discipline**. Performance appraisal is the continuous

assessment, in cooperation with the employee, of how she or he is doing relative to the standards and expectations laid out in the job description and follow–up training. Performance appraisal also includes identifying with the employee whatever corrective action may be necessary and steps by which the employee can advance his or her career.

❑ **Check Your Progress – 1 :**

1. _____ is determining the duties and skill requirements of a job and the kind of person to fill it.

   1. Hiring 2. Job Analysis
   3. Training 4. Writing Job Description

2. _____ introduces new people to the business' mission, its history, and its culture.

   a. Orientation b. Training c. Hiring d. Job Analysis

3. _____ give the employees the knowledge, skills, and abilities necessary to succeed in the position.

   a. Training b. Hiring c. Job Analysis d. Orientation

11.3 The HR Executive and Risk Control :

The HR executive may identify a risk and a specific need that is not being addressed. For example, there is a deviation between what should be occurring operationally and what is occurring. This deviation is causing productivity to drop slowly but steadily. Upper management is aware of this drop in productivity and is motivated to rectify the situation.

The HR executive having identified a risk, the drop in productivity for example, goes on to define the risk in terms of the human element. This is the underlying cause of the drop in productivity. Perhaps the line manager is not communicating effectively with staff, perhaps he/she is not delegating, or perhaps the workers feel they are not being listened to.

In this example, the HR executive meets with the manufacturing line managers, and their direct reports. The executive should ask questions to assess the risk, determine the extent of the problem and prioritize the risk. These should include :

- ✓ Questions about the business and performance needs of both line managers and their reports.
- ✓ Questions regarding work environment and how it must support the performance and or business needs.
- ✓ Questions to determine who is ultimately responsible for delivering these needs from the perspective of the managers and the direct reports.

Having met with and interviewed the line managers and their staff, the HR executive should be in a position to recommend a strategy to deal with the risk if he/she feels it is warranted. In the example of the manufacturing drop organization, the HR professional may recommend management development training, team building, and or coaching to fill the needs of management and their staff. The appropriate HR initiative effectively identified and applied will result in a return to productivity and elimination of this particular business risk.

The practical partnership between today's HR executive and strategic and operational business managers is discussed in Performance Consulting. "Performance consultants take the initiative to meet, work with, and gain the trust of managers and others within their organization. It has frequently been said that building trust takes time; it does not happen overnight. Performance consultants actively forge these relationships. Partnerships can, and should, be formed with various people in an organization : senior managers, other managers and supervisors, and thought leaders and subject matter experts, whatever their title. Partnerships are also forged with customers and suppliers to the organizations."

The HR Executive and Risk Control :

The HR executive has a vital role in controlling risk. A major component of Risk Management planning is risk avoidance. Many risks can be avoided by controlling and planning the human side of the corporate equation. Succession planning, adequate severance and outplacement, executive coaching and development will ensure that an organization has the means to deal with current and future challenges.

Risk resolution and control are important responsibilities for HR executives. Identifying crucial attributes for key executives within an organization, coaching and developing these attributes and monitoring the executives on an ongoing basis will help to minimize and resolve potential areas of risk such as employee turnover, low morale, potential litigation from misunderstandings arising between staff and management and negative publicity resulting from these or other human issues.

For example, a global organization based in the Bay Area plans to open a distribution center in Malaysia. The corporate goal is to increase sales and productivity in this growing region. The company has plans to grow their market share by establishing a local presence. The risk manager is concentrating on property exposure in this new region as well as political and environmental risk. The staffing plans include the transfer of two senior executives from the company headquarters in Silicon Valley.

The issues from the standpoint of Risk Management and HR are as follows :

Risk : **Property exposure**

HR Initiatives :

- ✓ Hiring professional engineers and surveyors to minimize the risk of fire, ensure building meets highest standards possible.

Risk : **Political risk**

HR Initiatives :

- ✓ Training and development of executives charged with running the new facility. Ensure they are sensitive to cultural differences.
- ✓ Hiring qualified public relations and local negotiations team.

Risk : **Business Interruption/ drop in productivity**

HR Initiatives :

- ✓ Succession planning.
- ✓ Diversify talent pool to ensure each key executive has a successor.
- ✓ Management development, team building and executive coaching.

Risk : **Directors and officers**

HR Initiatives :

- ✓ Succession planning.
- ✓ Diversify talent pool to ensure each key executive has a successor.
- ✓ Management development, team building and executive coaching.
- ✓ Diversity training.

Risk : **Employment practices liability**

HR Initiatives :

- ✓ Human resources handbooks and manuals.
- ✓ Documented employee hiring and training and termination procedures.
- ✓ Internal employee surveys.
- ✓ Outplacement services.
- ✓ Management, team building and executive coaching.

Not all Risk Management issues can be addressed by HR. These issues would include the risk of natural disaster as it relates to property. However, hiring appropriately trained professionals can minimize an organization's risk of engineering and design defects. In addition, even the effects of a devastating natural disaster can be minimized by training staff to effectively and efficiently respond to disaster.

A major Risk Management issue for high–tech companies in the Silicon Valley arises from the fact that up to 80% of engineers come from outside the United States, especially from countries such as the former North Korea, China and India. Companies could be at risk for the loss of an important part of their work force if foreign employees chose to return to their native lands due to political and economic changes that would allow them to earn western style salaries, be near families, and raise children in their traditional cultures.

The HR initiative addressing this risk might include helping employees to gain U.S. citizenship, purchase homes and truly settle down here, giving them more roots to our culture. Another HR/Risk Management initiative to address engineer turnover would provide adequate management training to technical managers so they are better able to assign projects among members of the technical staff thereby reducing the risk of competitor run–off. Software companies are particularly at risk of losing bored engineers who feel unappreciated and underutilized to companies who promise them more exciting responsibilities.

HR executives can monitor the success of their risk control initiatives through employee surveys, team building exercises and ongoing 360–degree multi–raters. The results can be tracked and as each goal is recognized, benchmarked. Clearly the disciplines of HR and Risk Management must be integrated to maximize productivity and positively impact the bottom line of any organization.

❑ **Check Your Progress – 2 :**

1. _____ identify a risk and a specific need that is not being addressed.

   a. HR Executive b. Risk Control

   c. Risk Executive d. None of Above

2. Hiring professional engineers and surveyors to minimize the risk of fire, ensure building meets highest standards possible is done by _____.

   a. Political Risk b. Directors and Officers

   c. Property Exposure d. None of Above

3. Explain issue from the standpoint of Risk Management and HR.

_____

_____

_____

_____

_____

11.4 Team Risk Management :

Team Risk Management describes the structure, operational activities for handling risks throughout all phases of the development of organization, such that all persons within the organizations, groups, departments, and agencies who are directly involved in the development and active as a team member.

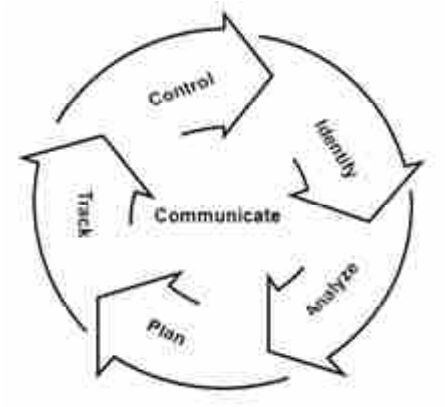
Team risk management practices bring together persons within an organization and between organizations to create working teams.

The team risk management method is built upon the nine principles which are as follows :

No.	Principle	Effective Risk Management Requires
1	Shared vision of product	A shared vision for success based upon unity of purpose, shared ownership, and shared assurance.
2	Forward-thinking search for doubts	Thinking in the direction of tomorrow, anticipating possible outcomes, identifying doubts, and handling program resources and activities while recognizing these doubts.
3	Open communications	A free flow of information at and between all levels through formal, informal, and unplanned communication and consensus-based processes.
4	Value of Individual awareness	The individual voice which can bring unique knowledge and awareness to the identification and management of risk.
5	Systems viewpoint	That software development be viewed within the larger systems-level definition, design, and development.
6	Combination into program management	That risk management be an important part of program management.
7	Practical policies	Practical policies that include planning and executing program activities based upon anticipating future events.

8	Systematic and adaptable methodology	A systematic approach that is adaptable to the program's infrastructure and culture.
9	Routine and nonstop processes	A nonstop observance considered by routine risk identification and management activities throughout all phases of the life cycle of the program.

These principles combine the SEI risk management model as shown in below figure and the ideas of helpful teams to create the foundation for a complete set of processes, methods, and tools for managing risks in software–dependent development programs.



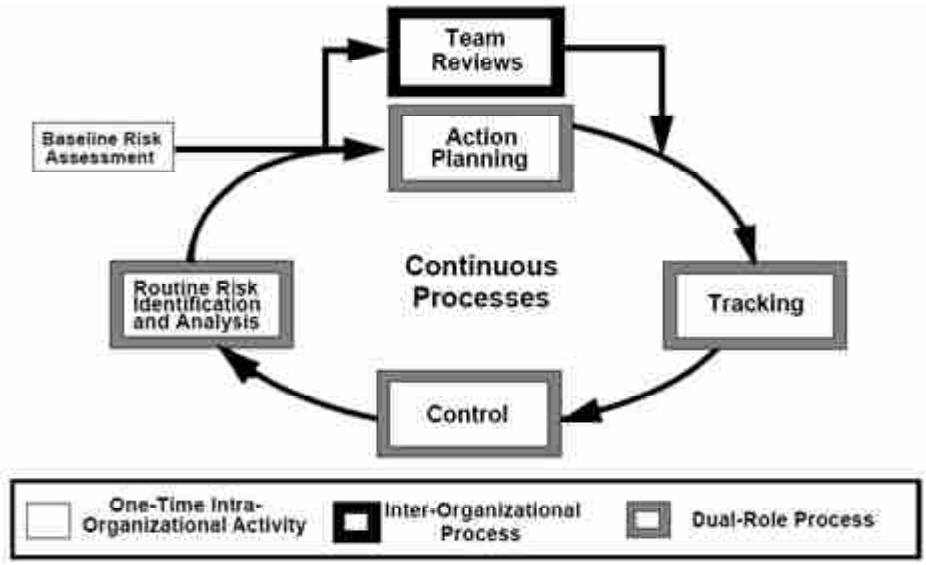
Risk Management Model

1. **Identify :** Search for and locate risks before they become problems adversely affecting the project.
2. **Analyze :** Process risk data into decision making information.
3. **Plan :** Translate risk information into decisions and actions and implement those actions.
4. **Track :** Monitor the risk indicators and actions taken against risks.
5. **Control :** Correct for originality from planned risk actions.
6. **Communicate :** Provide visibility and feedback data internal and external to your program on current and emerging risk activities.

- **Team Risk Management Processes :**

The team risk management procedures shown graphically in below figure include nonstop processes and a one–time process activity, the starting point of risk calculation. The nonstop processes of team risk management include all five steps of risk management model into four processes by joining the identification and analysis steps into the routine risk identification and analysis process.

Similarly, the starting point risk calculation joins the identification and analysis steps of the model into a single process activity that establishes the initial set of risks and starts the continuous process activities. The team review process defines cooperative risk management activities between the partner organizations.



Team Risk Management Process

The basic classifications of team risk management processes are summarized as follows :

Classification	Description
Continues Processes	The dual–role continuous risk management processes involve identification, analysis, planning, tracking, and control as well as inter–organizational activities, including regularly planned team reviews as well as unscheduled informal meetings and communications.
Starting point Activities	The initial, one–time identification and analysis of activities that establish the starting point set of risks shown by each partner organization, government and contractor.

1. Overview of Nonstop Team Risk Management Process :

The basic activities in nonstop team risk management have five parts as follows :

- Routine risk identification and analysis
- Team assessments
- Action planning
- Tracking
- Control

All of the activities are tied together through informal and formal communication processes. These activities improve the helpful interactions and trust among partners and team members, and build and support the shared program vision required for effective team risk management.

A. Routine Risk Identification and Analysis :

Routine risk identification and analysis includes the input of employees throughout the organization and combines the processes of identification and analysis into a set of separate team risk management activities. The approaches which can be working in routine risk identification and analysis are summarized as follows :

Method	Description	Characteristics
Routine Risk Form Processing	Routine distribution and processing of risk forms, submitted by program employees as risks are identified.	Separate input Nonstop Unidentified
Irregular Risk Reporting	Irregular reporting of risks by program employees.	Separate input Irregular scheduled event
Irregular Individual Interview Sessions	Irregular interviews of individuals throughout the program.	Separate input Irregular scheduled event Confidential Non-attribution
Periodic Risk Calculations	Abbreviated versions of the starting point risk calculation which are held periodically based upon time milestone events.	Peer group Irregular scheduled event Confidential Non-attribution

B. Team Assessments :

The team assessment is a combined meeting of the government and contractor program managers and their immediate staff to discuss and order risks. It carries together each program manager's list of current top risks, maintains continuity between these risks and those that were most important at the previous meeting, promises that there is a common understanding of the most important risks to the program, and assigns new action items. Its purpose is to build and maintain energy in government/contractor team risk management.

Each program manager will have the list of prioritized risks, the Joint List of Risks, from the previous team review meeting; this list will be the starting point for the team review. However, new risks will also have been identified in the organizations through the routine risk identification and analysis process. From these new risks, each program manager will select candidates for inclusion in the Joint List of Risks on the basis of responses to three questions :

- ✓ Which of these new risks informs the other party of a serious risk that they should be aware of ?
- ✓ Which may need to be transferred or delegated to the other party ?
- ✓ Which will require joint action to resolve ?

C. Action Planning :

Action planning for risks is the determination and implementation of actions necessary to manage a program's risks. This is where the integration of risk management processes with existing program management becomes most evident. Planning, in general, is an integral part of program management, whether planning how to meet specific milestones or determining the best design strategy for meeting specified requirements. Risk planning requires a systems perspective to maximize the effective use of rare resources within a program.

D. Tracking and Control :

The risk tracking and control procedures include creating and maintaining risk status information, defining action plans, and taking action based upon the status information. The important elements of risk tracking and control are very similar to the equivalent processes in traditional program or project management and can be readily integrated into a program's established tracking and control processes and methods.

2. Starting point Risk Assessment :

In a starting point risk calculation, a variety of methods and tools are used to initially recognize and analyze a set of risks and produce the initial Master Lists of Risks, one for the government and one for the contractor.

Paradigm	Method/Tools	Communication Characteristics
Recognize	Group interview Text-based questionnaire	Non-judgmental Non-attribution Confidential Peer grouping
Analyze	Criteria filtering Individual Top 5 Minimal group technique Comparison risk ranking	Individual voice Mutual understanding Agreement

❑ **Check Your Progress – 3 :**

1. Explain Continuous Team Risk Management Process.

11.5 Let Us Sum Up :

In this unit we have learnt that risk is expected in business organization at the time of taking projects. It is seen that project manager requires to make sure that risks are kept to lower level. It is noted that risk management carries human resources that has regular full–time employees with management and Labour personnel along with family and non–family members those working for full–time or part–time or seasonal.

It is studied that in order to keep track of progress of HR professional, it is clear that it should involves various phases of development.

11.6 Answers for Check Your Progress :

❑ **Check Your Progress 1 :**

1. (b) 2. (a) 3. (a)

❑ **Check Your Progress 2 :**

1. (a) 2. (c) 3. (refer 11.3)

❑ **Check Your Progress 3 :**

1. (refer 11.4)

11.7 Glossary :

1. **Job Analysis –** Job analysis is determining the duties and skill requirements of a job and the kind of person to fill it.
2. **Team Risk Management –** Team Risk Management defines the organizational structure and operational activities for managing risks throughout all phases of the life–cycle of a software–dependent development program such that all individuals within the organizations, groups, departments, and agencies directly involved in the program are participating team members.

11.8 Assignment :

1. Explain Human Resource and Risk Management.

11.9 Activities :

1. Write a note HR Executive and Risk Control.

11.10 Case Study :

Explain Team risk management and give difference between Continues Team Risk Management Process and Baseline Risk Assessment.

11.11 Further Reading :

1. Bruegge, B. and A. H. Dutoit (2000). Object–Oriented Software Engineering : Conquering Complex and Changing Systems. Upper Saddle River, NJ, Prentice Hall.
2. Cockburn, A. (2001). Agile Software Development. Reading, Massachusetts, Addison Wesley Longman.
3. Gluch, D. P., "A Construct for Describing Software Development Risks," Software Engineering Institute, Pittsburgh, PA CMU/SEI–94–TR–14.

BLOCK SUMMARY :

In this block, you have learnt and understand about the basic of risk management techniques with role of human resources. The block gives an idea on the study and concept of software risk analysis with its characteristic associated with risk management. You will be detailed with knowledge of risk identification issues with relative risks measures in software project.

The block detailed about the basic of insertion of risk management principles and practices appearing in software life cycle. The concept related to role and responsibilities of HR in maintaining risk for company growth and prosperity are also well detailed. You will be demonstrated practically about contingency plan technique.

BLOCK ASSIGNMENT :

❖ **Short Questions :**

1. Write a note on Risk.
2. Write a note on Risk Identification.
3. Explain Quantitative Risk Analysis.
4. What is Planning Risk Responses ?
5. Write a note on Monitoring and Controlling Risks.
6. Write a note on Human Resource and Risk Management

❖ **Long Questions :**

1. Explain Risk Analysis in Project Management.
2. Explain Qualitative Risk Analysis with its types and its process.
3. Explain Software Risk Management Implementation.
4. Explain HR Executive and Risk Control.
5. Explain Team Risk Management in Detail.

❖ **Enrolment No. :** ☐

1. How many hours did you need for studying the units ?

Unit No.	1	2	3
No. of Hrs.			

2. Please give your reactions to the following items based on your reading of the block :

Items	Excellent	Very Good	Good	Poor	Give specific example if any
Presentation Quality	☐	☐	☐	☐	_____
Language and Style	☐	☐	☐	☐	_____
Illustration used (Diagram, tables etc)	☐	☐	☐	☐	_____
Conceptual Clarity	☐	☐	☐	☐	_____
Check your progress Quest	☐	☐	☐	☐	_____
Feed back to CYP Question	☐	☐	☐	☐	_____

3. Any other Comments

.....

.....

.....

.....

.....

.....

.....

.....

Dr. Babasaheb Ambedkar
Open University Ahmedabad

BCAR-402

Software Engineering

BLOCK 4 : CASE STUDIES

UNIT 12 CASE STUDY – I : WASTE MANAGEMENT INSPECTION TRACKING SYSTEM

UNIT 13 CASE STUDY – II : LIBRARY MANAGEMENT SYSTEM

UNIT 14 CASE STUDY – II : SOFTWARE PROJECT MANAGEMENT

CASE STUDIES

Block Introduction :

In this block we will go through the various case studies. In case study are going to identify the different criteria like customer's requirement, existing system analysis, cost & time to develop the system and design of the software product or system.

As a part of case study there is a Waste Management Inspection Tracking System (WMITS) is going to manage by identifying requirements, risk, design, scheduling, team and control mechanism.

Furthermore, as a part of second case study there is Library Management System which help Students, Library administrator and Teachers to access the library in a computerized way. We found that if our Library Management system is automated or computerized then it will be very easy to search any book. It saves our time and our total Library Management system become very easy.

Block Objectives :

After learning this block, you will be able to understand:

- Basic Project Plan
- Project Estimation
- Risk Management
- Project Scheduling
- Project Team Organization
- Tracking and Control Mechanism

Block Structure :

Unit 12 : Case Study – I : Waste Management Inspection Tracking System

Unit 13 : Case Study – II : Library Management System

Unit 14 : Case Study – II : Software Project Management

Unit
12

CASE STUDY – I
WASTE MANAGEMENT INSPECTION TRACKING SYSTEM

UNIT STRUCTURE

12.0 Learning Objectives

12.1 Introduction

12.2 Waste Management System

12.2.1 Basic Project Plan

12.2.2 Project Estimates

12.2.3 Risk Management

12.2.4 Project Schedule

12.2.5 Project Team Organization

12.2.6 Tracking and Control Mechanism

12.3 Let Us Sum Up

12.4 Glossary

12.5 Assignment

12.6 Activities

12.7 Case Study

12.8 Further Readings

12.0 Learning Objectives :

After learning this unit, you will be able to understand :

- Idea about Basic Project Plan
- Project Estimation
- Detail about Risk Management
- Idea about Project Scheduling
- Project Team Organization
- Tracking and Control Mechanism

12.1 Introduction :

In this we will learn about how the **Waste Management Inspection Tracking System (WMITS)** is going to manage by identifying requirements, risk, design, scheduling, team and control mechanism.

The main purpose of WMITS is to help automate the entire process that the Department of Environmental Quality (DEQ) Waste Management Division (WMD) staff members perform throughout an inspection. Which will minimize the time span of any inspection, paper work, provide searchable database, and automated channel for the public to request information.

12.2 Waste Management System :

12.2.1 Basic Project Plan :

12.2.1.1 Goals and Objectives :

The main purpose of WMITS is to help automate the entire process that the Department of Environmental Quality (DEQ) Waste Management Division (WMD) staff members perform throughout an inspection. The goals of WMITS are :

- ✓ To minimize the time span of any inspection
- ✓ To minimize the amount of paper work required
- ✓ To provide a searchable database of all past inspections
- ✓ To provide an automated channel for the public to request information

12.2.1.2 System Statement of Scope :

1. General Requirements :

The following general requirements were laid out for our project named WMITS :

- ✓ A way in which DEQ could add new facilities to the database.
- ✓ A way in which DEQ could generate electronic checklists.
- ✓ A search on all electronic checklists.
- ✓ A way in which they could generate letters to be sent out to facilities based on inspection results.
- ✓ A way in which all letters and checklists could be stored electronically.
- ✓ A way to search for existing facilities.
- ✓ A way to print blank checklists and staff reports.
- ✓ A way in which they could view data which was entered into the database prior to our software.
- ✓ DEQ wanted a product that would allow them to easily add new checklists and letters or change existing checklists and letters.

- **Interface Enhancements :** Staff members of WMD have requested a lot of interface enhancements that will increase the usability of the product for the staff.
- **Database Administrative Interface :** There is currently no documented interface for WMD staff members to maintain the checklist and letter templates. Should no such interface existed, Cyber Rovers will have to implement one from scratch.
- **Online Help :** To develop an extensive help menu for the users and also to setup the online help for the need of the help in the future.
- **Training :** The staff members have also requested throughout training for the entire staff for use with the software.

2. Extended Enhancement :

- **Palm Pilot Integration :** Out of the two extended enhancement requests (palm pilot integration & online record access), the team and client both agree on doing the palm pilot integration. From the design point of view, online record access has a major security risk, which the team has little

or no experience on it. Palm pilot integration on other hand, needs only long programming, which can be (and will be) achieved by the team. We also suggest to the DEQ that they can make the online record request to be the next semester's project.

Before we decide on what kind of Palm Pilot we use, the team and the client have explored several options.

- **Database Restructuring :** The current database structure is not optimized at all. We will try to improve it as we go along.

3. System Context :

Eventually, multiple users will be using the product simultaneously. Therefore, concurrent connection will be an issue for implementation. In addition, this is a pilot product that hopefully, if successful, can be used in other locations as well. This leads to issues about future support for a larger user base.

4. Major Constraints :

- **Time :** We only have about two months to finish all documentation, software creation and enhancements. We have a lot of ideas but cannot implement them due to time constraint. One of the major ones is to move the application to be completely browser–based.
- **Funding :** To develop and implement the Palm Pilot integration, we will need funding to buy at least one Palm Pilot.

12.2.2 Project Estimates :

This portion of the document provides cost, effort and time estimates for the project using various estimation techniques, which will be elaborated in the appropriate section.

12.2.2.1 Historical Data Used for Estimates :

Although this project is to enhance the existing software, we were unable to obtain cost information from the previous project team.

12.2.2.2 Estimation Techniques Applied and Results :

Two estimation techniques have been used to generate two independent results for higher accuracy.

- Process–based
- Lines of Code (LOC) $\rightarrow$ COCOMO Model

1. Process–Based Estimation :

For process–based estimation, the process is decomposed into a relatively small set of activities or tasks. Then, the effort required to accomplish each task is estimated. Based on the project scope, the following software functions are defined :

- User Interface Re–engineering UIR
- Database Re–engineering DR
- Database Administrative Interface DAI
- Existing Bug Fixing EBF
- PalmPilot Integration PI

Activity →	Cust. Comm.	Planning	Risk Analysis	Engineering		Construction Release		Cust. Eval.	Totals
Task →				Analysis	Design	Code	Test		
Function									
UIR	0.40	0.02	0.02	0.02	0.50	0.30	1.00	0.05	2.31
DR	-	0.01	0.10	0.10	0.30	0.10	0.10	-	0.71
DAI	0.20	0.01	0.05	0.05	0.40	0.20	0.06	0.05	1.02
EBF	0.20	0.01	0.02	0.01	0.02	0.50	0.08	0.05	0.89
PI	0.25	0.02	0.04	0.20	0.50	0.30	1.00	0.06	2.37
Total	1.05	0.07	0.23	0.38	1.72	1.40	2.24	0.21	7.30
% Effort	14.38	0.96	3.15	5.21	23.56	19.18	30.68	2.88	100

Table 1 – Process–based Estimation Table

2. LOC–Based Estimation :

The following estimates are based on "best–effort" estimation from previous programming experiences and existing software size.

Functions		Estimated LOC
User Interface Re–engineering	UIR	2,300
Database Re–engineering	DR	200
Database Administrative Interface	DAI	1,000
Existing Bug Fixing	EBF	800
PalmPilot Integration	PI	1,000
Total Estimated Lines of Codes		**5,300**

12.2.2.3 Project Resources :

1. People :

This project will require three programmers in order to be finished in time. Each of the members will have to have specific skills (either obtained previously or on the fly). Team members will have to work in an interrelated network environment (ego–less) where everyone needs to communicate with everyone else on the regular basis.

2. Minimal Hardware Requirements :

Development :

Three IBM PC or compatibles with the following configurations

- Intel Pentium II 333MHz processor
- 64MB SDRAM
- 500MB Hard disk space
- Internet Connection

User Client–side :

IBM PC or compatibles with the following configurations

- Intel Pentium 166MHz processor
- 32MB SDRAM
- 20MB Hard disk space

User Server–side :

IBM PC or compatibles with the following configurations

- Intel Pentium II 333MHz processor
- 64MB SDRAM
- 500MB Hard disk space

3. Minimal Software Requirements :

Development :

- Windows 98
- Windows NT Workstation
- Windows NT Server
- Visual Basic 6.0 (three user licenses)
- Microsoft Access 97 and 2000
- Microsoft Word 97 and 2000

User Client–side :

- Windows 98/NT Workstation
- Microsoft Access 97/2000 (optional)
- Microsoft Word 97/2000 (optional)

User Server–side :

- Windows NT Server
- Microsoft Access 97/2000 (optional)
- Microsoft Word 97/2000 (optional)

12.2.3 Risk Management :

12.2.3.1 Scope and intent of RMMM activities :

We want the software to be free of any defects or errors, but it is hard or at times almost impossible to develop a system that is free of any defects. To be safe we would like to have a risk management plan to counter any difficulties that may impact the development or the creation of the software. Our goal is to assist the project team in developing a strategy to deal with any risk. For this we will take a look at the possible risks, how to monitor them and how to manage the risk.

For software development to avoid any risk both the developer and client have to work together. Client has to spend time with the developer in the beginning phase of the software development. If client decides to change the software, meaning if client wants to add some more functions into the software or to change the requirement, this will have major effect on the development of the software.

12.2.3.2 Risk management organizational role :

Everyone associated with the software has responsibility of managing the risk. That is if everyone participated and paid close attention to all the details during the early phase of the software development many risks can be avoided.

✓ Software development can avoid having risk by double–checking their schedule, product size, estimates regarding costs of the development etc.

- ✓ Customer can help avoid risk by providing all necessary software information during the early phase of the development.
- ✓ Software development team can avoid risk by getting all the details of the equipment that are provided or are accessible to them.
- ✓ Client can avoid risk by making all necessary business changes before initiating request for the software.

12.2.3.3 Risk Description :

This section describes the risks that are likely to be encountered during this project.

Risk is made up of two parts : the probability of something going wrong, and the negative consequences if it does.

1. Description of Risks

Business Impact Risk : This is the risk where concern is that of the not being able to come up or produce the product that has impact on the client's business. If the software produced does not achieve its goals or if it fails to help the business of clients improve in special ways, the software development basically fails.

Customer Risks : This is the risk where concern is client's motivation or willingness in helping the software development team. If the client fails to attend meeting regularly and fails to describe the real need of the business the produces software will not be one that helps the business.

Development Risks : If client fails to provide all the necessary equipment for the development and execution of the software this will cause the software to become a failure. So, in other words customer has to be able to provide time and resources for the software development team. If all the requested resources are not provided to the software development team odds for the software development to fail rises greatly.

Employee Risk : This risk is totally dependent on the ability, experience and willingness of the software development team members to create the working product. If the team members are not experience enough to use the application necessary to develop the software it will keep pushing the development dates until it's too late to save the project. If one or more members of the software development team are not putting in all the effort required to finish the project it will cause the project to fail. Employee risk is one of the major risks to consider while designing the software.

Process Risks : Process risk involves risks regarding product quality. If the product developed does not meet the standards set by the customer or the development team it is a failure. This can happen because of the customer's failure to describe the true business need or the failure of the software development team to understand the project and then to use proper equipment and employees to finish the project.

Product Size : This risk involves misjudgment on behalf of the customer and also the software development team. If the customer fails to provide the proper size of the product that is to be developed it will cause major problems for the completion of the project. If software development team misjudges the size and scope of the project, team may be too small or large for the project thus spending too much money on project or not finishing project at all because of shortage of finances.

Technology Risk : Technology risk involves using technology that already is or is soon to be obsolete in development of the software. Such software will only be functional for short period of time thus taking away resources from the customer. Since the technology changes rapidly these days it is important to pay importance to this risk. If customer request use of software that soon to be obsolete software development team must argue the call and have to pursue customer to keep–up with current technology.

12.2.3.4 Risk Table :

The following table describes the risks associated with the project. The appropriate categories of the risks are also given, as well as probability of each risk and its impact on the development process.

Probability and Impact for Risk m

The following is the sorted version of the above table by probability and impact :

Category	Risks	Probability	Impact
Employee Risks	Lack of training and experience	40%	1
Process Risk	Low product quality	35%	1
Product Size	Where size estimates could be wrong	30%	2
Development Risks	Insufficient resources	30%	2
Customer Risk	Customer may fail to participate	20%	3
Technology Risk	Obsolete technology	10%	2
Business Impact	Product may harm the business	10%	3

Table 1 – Risks Table (sorted)

Impact Values	Description
1	Catastrophic
2	Critical
3	Marginal
4	Negligible

Above is the table that categorizes the risks involved in software development. It gives brief description of the risk in Risk's column and also provides the probability of risk occurring in percentages in Probability column and also the impact of the risk in the Impact column.

The impacts values assigned to each risk are described in the section below the risk table. It is very convenient way to look at the risk and derive the information of the risk.

12.2.4 Project Schedule :

Following is the master schedule and deliverables planned for each stage of the project development lifecycle, and their respective planned completion dates.

12.2.4.1 Deliverables and Milestones :

Stage of Development	Stage Completion Date	Deliverable	Deliverable Completion Date
Planning	01/21/00	Quality Assurance Plan Project Plan Milestone	01/15/00 01/20/00 01/21/00
Requirements Definition	02/25/00	Draft Requirements Specification Draft Design Specification Project Test Plan Requirements Specification (final) Milestone	02/09/00 02/09/00 02/15/00 02/22/00 02/22/00
Design (Functional & System)	03/01/99	Draft Training Plan Program and Database Specifications Design Specification (final) Milestone	02/23/00 02/26/00 02/29/00 03/01/00
Programming	04/02/00	Software (frontend and backend) System Test Plan User's Guide Operating Documentation Milestone	03/31/00 03/10/00 03/20/00 03/28/00 04/02/00
Integration & Testing	04/15/00	Test Reports Training Plan (final) Acceptance Checklist User's Guide (final) Milestone	04/03/00 04/10/00 04/14/00 04/12/00 04/15/00
Installation & Acceptance	04/20/00	Maintenance Plan Acceptance Test Report Milestone	04/16/00 04/20/00 04/20/00

12.2.4.2 Work Breakdown Structure :

Please refer to the above table for the work breakdown structure (deliverables are set according to logical work breakdown).

12.2.5 Project Team Organization :

The structure of the team and the roles of team members are defined in this section.

12.2.5.1 Team Structure :

Due to the small size of the project team, the team will be organized in an egoless structure, where the entire group will make most of the decisions together.

Conceptual and Advanced Interface Development :

- Overall process specification
- Database re–engineering
- Advanced Interface Development
- Internal Modules Development
- Draft Documentation

User Interface Design and Development / Trainer :

- Intermediate User Interface Development
- Training Sessions
- Operational Manual
- Draft Documentation

Editor / Master Tester / Maintenance :

- Proof Reading
- Overall Testing and Reports
- All Final Documentation
- User Guide

12.2.5.2 Additional Member Responsibilities :

The following are additional notes on team members' responsibilities :

- ✓ Each person will work on multiple tasks throughout the development process in addition to their main role.
- ✓ Due to the small size of the team, project design and specification will be discussed with the whole team.
- ✓ All development personnel are required to prepare draft documentation of their work, as well as individual testing on their part. Tester will do the overall final testing at the end of the testing stage.
- ✓ One programmer will take on the role as coordinator to ensure the project is on schedule, as well as scheduling In–Stage Assessments.

12.2.6 Tracking and Control Mechanism :

12.2.6.1 Quality Assurance Mechanisms :

- ✓ Tight Change Management
- ✓ Extensive before implementation Design using Rapid Prototyping
- ✓ Close Contact with Clients, meeting every two weeks and regular email contacts
- ✓ Plenty of Research on PalmPilot platform before development

12.2.6.2 Change Management and Control :

- ✓ For changes affect the user experiences we will have to notify all clients
- ✓ For changes that do not affect the user experiences we will notify a client representative
- ✓ Due the size of the team, internal control panel will be used. One member of the team suggests a change, it will need to be approved by the other two members
- ✓ Formal version numbering will be used. All version changes must be documented in a common document accessible to all team members before a new version number can be released. Version number will be structured as follows :

<Major Release>.<Minor Release><Bug fix>

12.3 Let Us Sum Up :

In this unit we have learnt that goals and objective, system scope of system, project estimation, project resource in which people, minimal hardware requirement, minimal software requirement are going to define.

We also seen about Risk management in which we learnt scope and intent of RMMM activities and risk description. As a part of risk description, we learn about Business impact risk, Customer risk, Development risk, Employee risk, Process risk, Product size, Technology risk.

We also seen about the master schedule, the structure of team and the roles of team members, and tracking and control mechanism.

12.4 Glossary :

1. Risk – Risk is made up of two parts : the probability of something going wrong, and the negative consequences if it does.

12.5 Assignment :

1. Explain Team Structure in detail.

12.6 Activities :

1. Define Project Scheduling in detail.

12.7 Case Study :

Explain the future of Quality Assurance.

12.8 Further Reading :

1. Bruegge, B. and A. H. Dutoit (2000). Object–Oriented Software Engineering : Conquering Complex and Changing Systems. Upper Saddle River, NJ, Prentice Hall.
2. Cockburn, A. (2001). Agile Software Development. Reading, Massachusetts, Addison Wesley Longman.
3. Gluch, D. P., "A Construct for Describing Software Development Risks," Software Engineering Institute, Pittsburgh, PA CMU/SEI–94–TR–14.

Unit 13

CASE STUDY – II LIBRARY MANAGEMENT SYSTEM

UNIT STRUCTURE

13.0 Learning Objectives

13.1 Introduction

13.2 Library Management System

- **13.2.1 Objective**
- **13.2.2 Project Life Cycle**
- **13.2.3 Existing System**
- **13.2.4 Proposed System**
- **13.2.5 Requirement Determining**
- **13.2.6 Development Phase**
- **13.2.7 Design of System Model**
- **13.2.8 Conceptual Model of our Proposed Library Management System**

13.3 Let Us Sum Up

13.4 Assignment

13.5 Activities

13.6 Case Study

13.7 Further Readings

13.0 Learning Objectives :

After learning this unit, you will be able to understand :

- Idea about of project life cycle
- Project Development
- Strategy for determining requirement information
- Software Requirement
- Hardware Requirement

13.1 Introduction :

We are trying to develop an automation system which will provide lots of facilities to our college. The total automation system divided into many modules, here our parts is "Library Management System". This is a small part of total automation System but The Library Management System will provide an environment which facilitate teachers & students easy to access the library information.

The Aim of this project is to help our student, Library administrator and Teacher to access our library in a computerized way. We found that if our Library Management system is automated or computerized then it will be very easy to search any book. It saves our time and our total Library Management system become very easy.

13.2 Library Management System :

13.2.1 Objective :

- ✓ It will help student or library administrator to access library easily
- ✓ To reduce people messy.
- ✓ Searching process of a book becomes very easy.
- ✓ Maintenance of these books becomes very easy.
- ✓ To assure the information of the library such as book types, copy number of books, authors name, availability of particular book etc.
- ✓ To make secured data storage of library information.
- ✓ Manage the library as a systematic way.
- ✓ Huge information can be stored.

13.2.2 Project Life Cycle :

The project life cycle includes various development phases that occur in the life of project starting right from the inception of the project to its final development at the client's end. The three development phases in a project life cycle are :

- **Project initiation**
- **Project execution**
- **Project development**

- **Project initiation :** The project initiation phase is first phase of life cycle. This phase involves creating a complete plan for the project, specifying various activities that will be performed and assigning responsibilities to team members on the basis of their skill set.
- **Project execution :** After the project plan is made and the responsibilities assigned, the actual development of the project starts. The phase in which the actual development of the project takes place is known as the project execution phase. This is the most crucial phase of any project and is subdivided into the following phases :

A. System Analysis :

- ✓ Initial study
- ✓ Information gathering
- ✓ Feasibility study

B. System Design :

- ✓ Design standard
- ✓ High level design & design tools
- ✓ Database design
- ✓ Logical design
- ✓ Construction

C. System Implementation :

- ✓ Integration & testing
- ✓ Post implementation

- **Project development :** After the project execution phase, the final phase of a project life cycle is the project development phase. In this phase, the deployed at the client side. This phase also involves providing customer support to the client for some specified period of time.

When project is built it may possibly remain error les of more, because several type of modification can take place several times. So, for the very first time when we run the database web site, we found few problems in tools potions. We fixed this problem including some minor problems immediately, and afterwards the application runs properly.

13.2.3 Existing System

The system we have currently is a poor manual library system. There is a lot of books in library but no serial number of them. Different writers have different books but no chart of them. Our library supervisor maintains only a register chart. Where there is no information about the book lender. So, it is difficult to find out the book lender in next time. And it is risky too to give a book. Students are not able to lend a book from the library because library supervisor has no sufficient information about them that she/he can search out the lender.

Our existing library management system is a manual system. The whole system is manually defined and it has some problems. The problems of existing systems are as follows :

- ✓ It is very slow and takes many times.
- ✓ It is very difficult to maintain.
- ✓ It is not error free.

13.2.4 Proposed System :

A Library Management System is a system where a user can access a library automatically. Here automatically stands for computerized way. In a manual system when we go a library, we see a lot of books are in self by self. There is no member, no serial of these books. It is difficult to find out a certain book for a certain writer. To reduce these haphazard, we decide to make this LMS system automated. In this system a user easily get which books are in the library. How many copies have of them, the name of the writer of the book etc.

But now we want to do it automatically. Which will be so easier for Whole University and it has some advantages as follows :

- ✓ Dynamic System
- ✓ Error free
- ✓ User Friendly

13.2.5 Requirement Determining

- ✓ For the requirement analysis we use the key strategies for determination of requirements of the user.
- ✓ Getting information from the existing system.
- ✓ Interview
- ✓ Questionnaires
- ✓ Hardware & software requirements Getting Information from the Existing

In this stage we simply ask the personnel of meeting management information section– what information are currently received and what other information are required. From this stage we find that the meeting members of the university are recording manually their personal information in the member register.

13.2.5.1 Interview :

Why do we conduct interviews during system analysis, the reasons are these ?

- ✓ We need to gather information about the behavior of a current system or the requirements of a new.
- ✓ We need to verify our own understanding as system analyst of the behavior of a current system or the requirements of a new system. This understanding was probably acquired through previous interviews together with independently gathered information.
- ✓ We need to gather information about the current system and/or system in order to carry out cost–benefit meeting between a system analyst and an end user.

We took the interview of the teacher, student, and Library Supervisor. As interview is the most common and most satisfactory way of obtaining information, particularly to obtain information about objectives constraints, allocation of details and problems and failures in the existing system.

After the interview all notes are read through and expanded to make them intangible. As the data are in random order, they were revised into a more useful order before the next work is commenced.

Hardware & software requirements.

13.2.5.2 Software Requirements :

- ✓ Any Operating System.
- ✓ Internet Explorer

13.2.5.3 Hardware Requirements :

- ✓ Component Minimum Maximum
- ✓ Process speed 233 MHz Higher (P4)
- ✓ Ram 64 MB Higher
- ✓ Graphic Card AGP 32 MB Higher
- ✓ Monitor Any Color Monitor Higher
- ✓ CD Rom Any 16X Higher Project phases

13.2.6 Development Phase :

Phase1 : Analysis the requirements of the project.

In this phase we basically analysis the requirements and develop our knowledge on demand. We will sort out all the necessary tools that will be needed. We will grow up the technological background to make workable the software in all environments.

The method of collecting requirements :

- ✓ Reading books & related reference book.
- ✓ Internet Browsing.

- ✓ Talking with the students, our friends who are interested to help us by giving information about Library management System.
- ✓ Talking with our supervisor & other teacher who are experienced to make Library and working with the automation.
- ✓ Talking with Programmer or experienced people who are working this type of related sector.

Phase2 : Module Analysis

In this phase we will analyses our module and fragment the overall module in some small modules. Which help us to complete total system easily.

Phase3 : Develop Modules

We will make the task flow and code flow of each module in this phase. We will write the row code to build up the modules.

Phase 4 : Integrate Modules

In this phase we will integrate all modules. The backbone of the software will stand up in this phase and the software will be useable.

Phase 5 : Test, bug finding and bug fixing We will test the overall features of the software.

By testing the features, we will find out the bugs. After that all the bugs will be solved.

Phase 6 : Use of the software

This software will be used for Our University Automation System for Library management.

13.2.7 Design of System Model :

In our Routine Management System there are three types of User models are shown

These are :

- **Normal User**
- **Administrator**
- **Registered User**

- **Normal User :** A regular user is any kind of user like students, teachers or anybody who uses the system and can see the online library and get information.
- **Administrator :** An admin user is a selected user who has the permissions to create a new admin or edit update delete operation. The admin users also perform the book function like book borrow, book lending book return etc.
- **Registered user :** It means that, only our students, teacher, & employee are permitted to registration. These types of people have to has perform book borrow, return function.

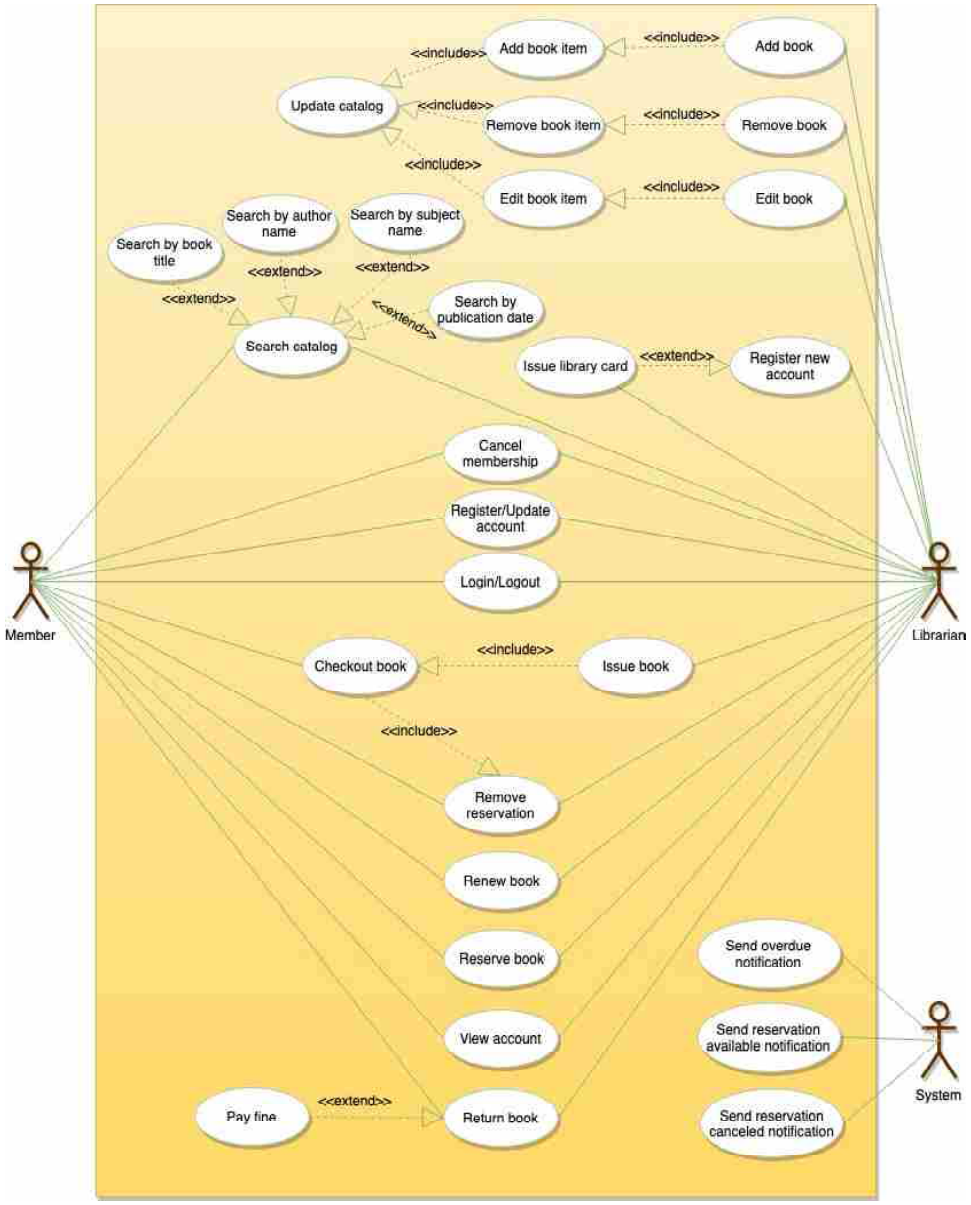
We will focus on the following set of requirements while designing the Library Management System :

1. Any library member should be able to search books by their title, author, subject category as well by the publication date.
2. Each book will have a unique identification number and other details including a rack number which will help to physically locate the book.
3. There could be more than one copy of a book, and library members should be able to check–out and reserve any copy. We will call each copy of a book, a book item.
4. The system should be able to retrieve information like who took a particular book or what are the books checked–out by a specific library member.
5. There should be a maximum limit (5) on how many books a member can check–out.
6. There should be a maximum limit (10) on how many days a member can keep a book.
7. The system should be able to collect fines for books returned after the due date.
8. Members should be able to reserve books that are not currently available.
9. The system should be able to send notifications whenever the reserved books become available, as well as when the book is not returned within the due date.
10. Each book and member card will have a unique barcode. The system will be able to read barcodes from books and members' library cards.

- **Use case diagram :**

We have three main actors in our system :

1. **Librarian :** Mainly responsible for adding and modifying books, book items, and users. The Librarian can also issue, reserve, and return book items.
2. **Member :** All members can search the catalog, as well as check–out, reserve, renew, and return a book.
3. **System :** Mainly responsible for sending notifications for overdue books, canceled reservations, etc.



Use case diagram

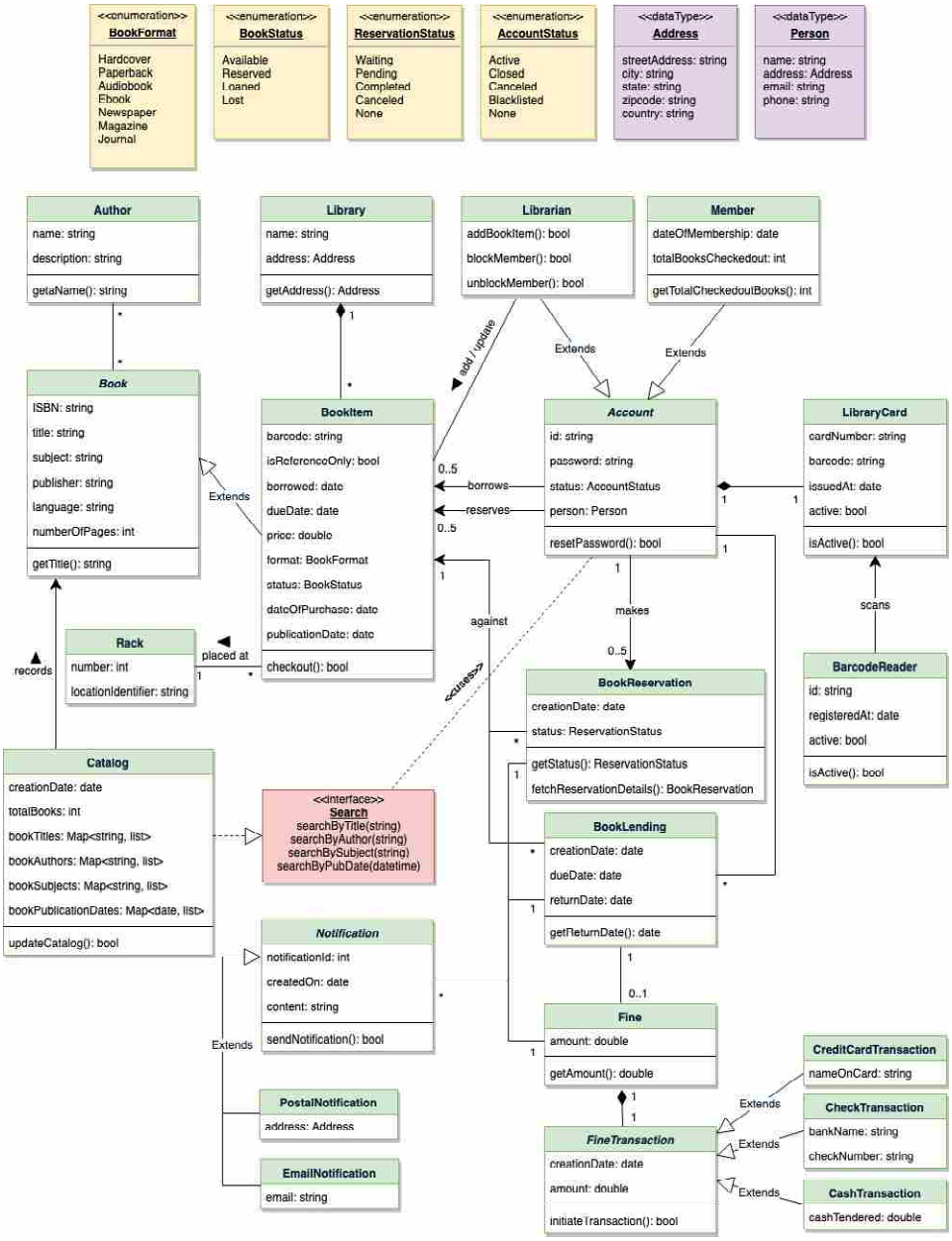
Here are the top use cases of the Library Management System :

1. **Add/Remove/Edit book :** To add, remove or modify a book or book item.
2. **Search catalog :** To search books by title, author, subject or publication date.
3. **Register new account/cancel membership :** To add a new member or cancel the membership of an existing member.
4. **Check–out book :** To borrow a book from the library.
5. **Reserve book :** To reserve a book which is not currently available.
6. **Renew a book :** To reborrow an already checked–out book.
7. **Return a book :** To return a book to the library which was issued to a member.

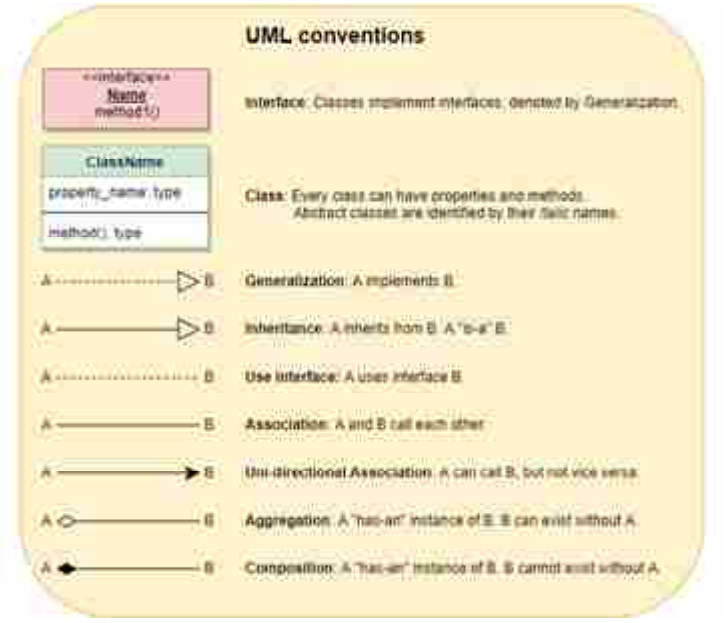
- **Class diagram :**

Here are the main classes of our Library Management System :

1. **Library :** The central part of the organization for which this software has been designed. It has attributes like 'Name' to distinguish it from any other libraries and 'Address' to describe its location.
2. **Book :** The basic building block of the system. Every book will have ISBN, Title, Subject, Publishers, etc.
3. **BookItem :** Any book can have multiple copies; each copy will be considered a book item in our system. Each book item will have a unique barcode.
4. **Account :** We will have two types of accounts in the system, one will be a general member, and the other will be a librarian.
5. **LibraryCard :** Each library user will be issued a library card, which will be used to identify users while issuing or returning books.
6. **BookReservation :** Responsible for managing reservations against book items.
7. **BookLending :** Manage the checking–out of book items.
8. **Catalog :** Catalogs contain list of books sorted on certain criteria. Our system will support searching through four catalogs : Title, Author, Subject, and Publish–date.
9. **Fine :** This class will be responsible for calculating and collecting fines from library members.
10. **Author :** This class will encapsulate a book author.
11. **Rack :** Books will be placed on racks. Each rack will be identified by a rack number and will have a location identifier to describe the physical location of the rack in the library.
12. **Notification :** This class will take care of sending notifications to library members.



Class diagram for Library Management System



UML Conventions

13.2.8 Conceptual Model of our Proposed Library Management System :

13.2.8.1 Analyzing & Specification :

We have found we have three libraries in three different buildings in an average each library has 5000 books. Different department has different library in a separate building Library administrator maintains an account (khata) to keep information about the book no student could lend any book or even couldn't see what the books in the library are.

To reach our project goals our LMS system must has to provide following features :

- ✓ Student could see book list
- ✓ Student could lend book from library
- ✓ Administrator has to have the option add edit delete remove the booklist.
- ✓ Administrator & student could see the borrowed the book list
- ✓ The total system should be internet based.

13.3 Let Us Sum Up :

In this unit we have learnt project initiation, execution and development in project life cycle. We also seen how requirements are going to gather and for gathering requirement interview method is going to use.

We also seen about different development phase like analysis the requirements of the project, Module Analysis, Develop Modules, Integrate Modules, and Test, bug finding and bug fixing.

We also seen about the Normal user, Administrator and Registered user as a part of design of system model.

13.4 Assignment :

1. Discuss Project Life Cycle.

13.5 Activities :

1. Explain Requirement Determining.

13.6 Case Study :

Explain Development phase and Design of System model.

13.7 Further Reading :

1. Bruegge, B. and A. H. Dutoit (2000). Object–Oriented Software Engineering : Conquering Complex and Changing Systems. Upper Saddle River, NJ, Prentice Hall.
2. Cockburn, A. (2001). Agile Software Development. Reading, Massachusetts, Addison Wesley Longman.
3. Gluch, D. P., "A Construct for Describing Software Development Risks," Software Engineering Institute, Pittsburgh, PA CMU/SEI–94–TR–14.

Unit
14

CASE STUDY – III SOFTWARE PROJECT MANAGEMENT

UNIT STRUCTURE

14.0 Learning Objectives

14.1 Introduction

14.2 Measuring a Software Project

14.3 Rapid Application Development (RAD) Method

14.4 Prototype Method

14.5 Agile Scrum Method

14.6 Hospital Management System

14.7 Let Us Sum Up

14.8 Glossary

14.9 Assignment

14.10 Activities

14.11 Case Study

14.12 Further Readings

14.0 Learning Objectives :

After learning this unit, you will be able to understand :

- Basic of RPC arrangement
- Understanding of Shared virtual memory
- Concept of Unix Processing
- Idea about Unix Shell

14.1 Introduction :

Many software projects carry different scenarios in terms o0f completion and finalization. They fail in developing required functionality which is available in their schedule with planned budget. It results in lack of quality. Hence from the past years many companies started taking interest to enhance software development. These initiatives mostly focus on improving the software processes and the technology used during software development. One area often underestimated but crucial for every software development project is project management. Project management is one of the key factors influencing the project success or failure.

14.2 Measuring a Software Project :

At first, we conducted a study whose purpose was to assess and better understand the current practices and problems of software project management as well as how they impact software development projects. The investigation was performed by means of structured, on–site interviews with software project managers. The questionnaire used for these interviews covers more than 70 factors potentially influencing the process and outcomes of software projects.

The questions mainly address human factors and organizational aspects. In case the interviewed project managers had managed more than one software project, they were asked to answer the questions for each of these software projects. Thus, the project managers could supply all the information they had acquired. But it became apparent that the same project manager answered most of the questions identically for every project he had managed. The data in the following sections describe the kinds of information we collected.

For this aspect it had to be analyzed in which way the project environment had affected the course of the project and in particular project management activities. In our investigation we considered the following environmental factors : senior management, users, customers, and subcontractors.

Based on the results of the first assessment we performed a second investigation by measuring an already completed software development project in considerable detail. Its goal was to verify whether the qualitative and subjective information collected during the first study could be validated and melded with the quantitative and qualitative data of a measured software project. Measuring a software project requires multiple sources of information. Hence, the empirical investigation was implemented as follows : At the beginning, as in our first study, we performed structured interviews with the technical and the project manager of the examined software project. In addition, we executed written questionings with the project staff. Some information concerning defect and change management data was provided by a tracking system used in the software project. Data on the project key deliverables such as specification, source code, and user documentation were collected by applying several code, size, and complexity metrics.

For each method and tool used in the software project we recorded information on the schedule of application, the impact on productivity and on quality, the costs caused for purchasing the tool and for training as well as the usefulness of the method or tool. Furthermore, we were interested in the general tool availability. In case of defect removal methods, some more information was required. To explore the effectiveness of quality assurance activities applied in the project we collected information on the number of defects found and the effort spent for the quality assurance activity. For every defect we recorded the origin, severity, the amount of effort required to fix it, and the quality assurance activity that had detected the defect.

Result :

The first investigation shows that the effort project managers invested in leading a project was in the range of 5 to 100% of their overall working time. Although even those project managers who spent less time for project management activities (5–35%) regarded this time as adequate and sufficient, analysis of the data indicates that less time for project managing and controlling often led to enormous overruns in cost and schedule. As a consequence, 75% of the investigated projects were not completed in the planned schedule – the deviations from schedule were up to 150%. Another interesting aspect in this context is that most project managers regarded bad education of team members, moving requirements, or unstable developing environments as the main reasons for those overruns in schedule. If more time was spent in managing the project and controlling its progress the deviations from schedule were minor (from 20–70%). Here project managers regarded too optimistic planning as the main reason for the overruns in schedule.

The second investigation verifies these results. It shows that the project manager could only spend a maximum of 50% of his overall working time for leading the project, because at the same time he participated in up to four different software projects. The project, too, was completed with enormous cost and schedule overruns.

14.3 Rapid Application Development (RAD) Method :

- **Introduction :**

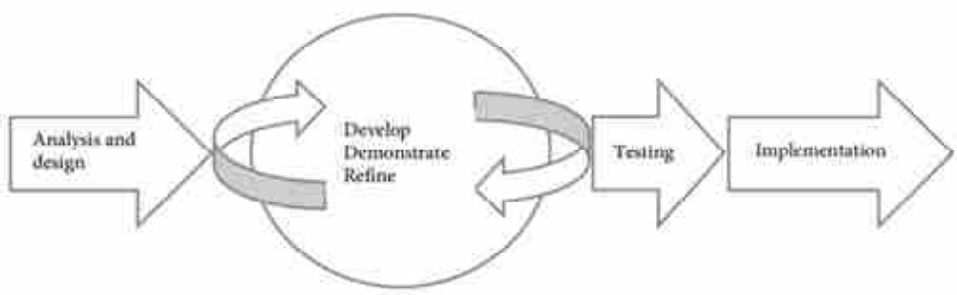
Rapid Application Development is an advanced development model which provides extra significance to rapid prototyping and fast response over an extensive development and testing cycle. This model agrees developers to create multiple repetitions and apprises to a software rapidly, without the requirement to resume a development plan from scratch each time.

It is considered to take benefit of great development software like CASE tools, prototyping tools and code initiators. The main objectives of RAD are :

- ✓ High Speed
- ✓ High Quality
- ✓ Low Cost.

RAD extremely increases the quality of completed systems while dropping the time it proceeds to construct them.

Rapid Application Development is a software development method that uses marginal planning all for rapid prototyping. The functional components are developed all together as prototypes and are joined to make the whole product for quicker product delivery.



Rapid Application Development Model

Study :

Business needs to reduce the consumption of vehicle in the plant area. Convention of the vehicle is for movement of logistics and to give a serious maintenance in un–planned power cut. Essential online reservation of vehicles as on mobile application.

The suggested output will be a mobile application produced and data put in storage on cloud. Technology used is J2ME (Java 2 Platform, Micro Edition) and Oracle and Google application used to identify the locality and shortest path in source and destination. OLA type of application organized for users together with phone booking skill.

SDLC Model improved RAD Model and other related models like prototype, agile, iterative enhancement models and spiral.

We have arranged a small application with restricted possibility only with booking ability for a small number of user division and for single branch. Once

fruitfully completed then joined rest of the elements like Booking Cancellation, Advance Booking, Urgent Booking, Generate Bill and then installed it for all places and all sectors.

14.4 Prototype Method :

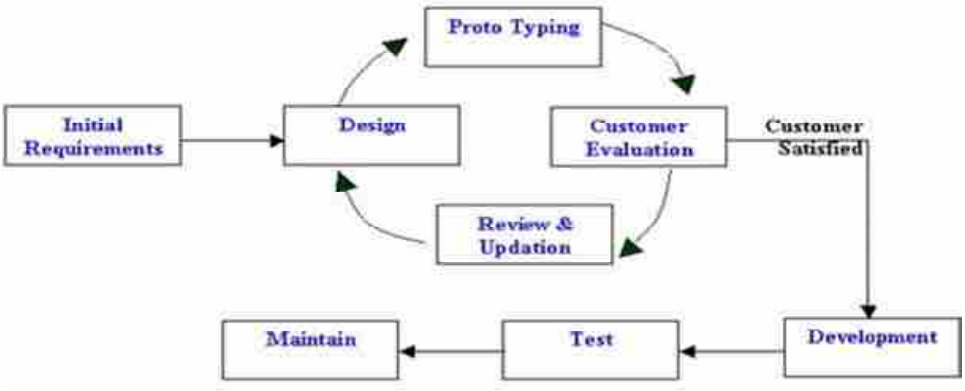
- **Introduction :**

Prototyping Model is a software development model in which prototype is built, tested, and reworked till an adequate prototype is completed. It also generates base to produce the complete software. It works best in situations where the project's needs are not known in detail.

The lifecycle of the prototype. Some are built quickly, tested, thrown away, and then replaced with an improved version–this is called rapid prototyping. Most prototypes will go over five different phases :

- ✓ Defining
- ✓ Focusing on features
- ✓ Production
- ✓ Testing
- ✓ Presenting

Prototype Model is an initial model, ideal, or publication of a product constructed to test a model or method. It is a time used in a selection of frameworks, comprising semantics, design, electronics, and software encoding. A prototype is aimed to check and attempt a new design to improve accuracy by system experts and managers.



Proto Type Model

Study :

Business needs to build a GPS tracker that can be designed on the railways holders to get the most out of the consumption and display the similar.

The suggested output will be cloud based web application shaped and data put in storage on cloud. Technology used is Java Hibernate and Oracle application server for checking. Furthermore, a GPS generated distinctly by research and development sector.

SDLC Model Improved Prototyping Model and other related model is Rapid Application Development Model.

We have arranged a small website application and let a single GPS tracker as a path and observe its movement to see the holder point among a specific range. When it got successfully completed then executed whole project.

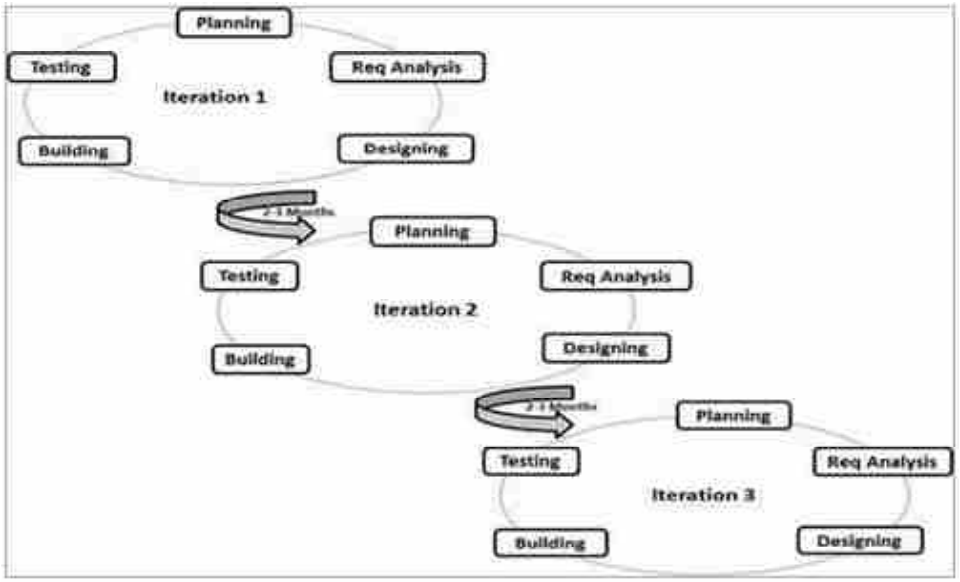
14.5 Agile Scrum Model :

- **Introduction :**

Scrum is an agile development method used in the development of Software based on an iterative and incremental methods. Scrum is flexible, fast and effective active framework that is designed to provide value to the customer during the development of the project.

Agile scrum approach has several profits. It inspires products to be built faster, as each set of aims must be accomplished within each sprint's time frame. It also involves repeated scheduling and goal setting, which helps the scrum group effort on the current run's objectives and increase production.

This model considers that each task desires to be held otherwise and the current approaches essential to be custom–made to top suit the project necessities. In Agile, the jobs are separated to phase cases to provide particular structures for a statement.



Agile Scrum Model

Iterative method is occupied and functioning software construct is provided after separately reiteration. Respectively construct is incremental in expressions of sorts; the final form embraces all the structures essential by the consumer.

Study :

Business needs to make panel to show the Overall Equipment Effectiveness for all manufacturing plants on monthly basis and developments for whole year and make available an ability to show the drawbacks for concentrated Overall equipment effectiveness. We have organized reports for a plant in Tableau desktop tool (data visualization software) which was free of cost. After than we have applied it in authorized form and installed it in all departments as per suggested output. SDLC Model improved Agile Scrum Model and the other appropriate model is RAD. We have developed a Web Application and installed it in one sector for a month and then installed it for all sectors during completion of development process.

14.6 Hospital Management System :

HC Hospital Management System :

HC Infotech Ltd. has developed a core package – Hospital Management System that addresses all major functional areas of Hospital. The development environment ensures that HC HMS has the portability and connectivity to run on virtually all standard hardware platforms, with stringent data security and easy recovery in case of a system failure. HC HMS provides the benefits of streamlined operations, enhanced administration and control, improved response to patient care, cost control, and increased profitability.

Some of the Subsystem Modules in HC HMS :

Reception : The reception module handles various enquiries about the patient's admission and discharge details, bed census, and the patient's movements within the hospital. The system can also handle fixed–cost package deals for patients as well as Doctor Consultation and Scheduling, Doctor Consultancy Fees and Time Allocation.

OPD, IPD Registration and Admission : This module helps in registering information about patients and handling both IPD and OPD patient's query. A unique ID is generated for each patient after registration. This helps in implementing customer relationship management and also maintains medical history of the patient.

Administration : This module handles all the master entry details for the hospital requirement such as consultation detail such as doctor specialization, consultancy fee, and service charges.

Security : This module handles multi–level security of HC HMS so that every admission and transaction can be traced with the help of user ID.

Pharmacy Store : This module deals with all medical items. This module helps in maintaining Item Master Maintenance, Receipt of Drugs/consumables, issue handling of material return, generating retail bills, stock maintenance. It also helps in fulfilling the requirements of both IPD and OPD Pharmacy.

Purchase : This module helps in raising purchase orders, maintaining purchase details and other purchase related details.

Phlebotomy : This specific module caters in maintaining test requisitions, sample collection status and various procedures for collection of samples for the tests prescribed.

Laboratory : This module enables the maintenance of investigation requests by the patient and generation of test results for the various available services, such as clinical pathology, X–ray and ultrasound tests. Requests can be made from various points, including wards, billing, sample collection and the laboratory receiving point. The laboratory module is integrated with the in–patient/ outpatient registration, wards and billing modules.

Emergency : The development of this module keeps in mind the criticality of this department. Every care has been taken to ensure that minimum of time is taken to register the patient, so as to reduce the tension of the already stressed–out relatives. Neither any detailed contact information of the patient is required nor any information about the payment type is solicited.

OT Management : This module deals with operation theatre activities such as equipment used detail, resource ordering, drug order, gynecology detail

recording, laboratory order and reports transfer requisition, patient monitoring, blood request, new born baby detail and details of delivery.

Minor Surgery : This module is same in features as in OT management though the function is different. This module deals with the surgeries minor in nature, which does not require complete anesthesia.

Blood Bank : The blood bank module provides information on the collection and storage of blood, results of blood tests, cross–matching identifications, and transfusion reactions.

Ward Management : The ward management module takes care of medical equipment, doctor visit, vitals recording, patient case sheet, diet ordering, blood requisition, transfer intimation and discharge intimation etc. It also deals with the maintenance of the wards, inter– and intra–ward transfers.

OPD and IPD Billing : The billing module facilitates cashier and billing operations for different categories of patients and automatic posting of charges for different services such as lab tests, medicines supplied, consulting fees, food and beverage charges, etc. It enables credit party billing through integration with the financial accounting module.

Intensive Care Unit (ICU) : This module caters to scheduling, maintaining ICU Record, drug orders, consultant details, specific blood requests etc.

Food and Beverages : This module facilitates collection of information regarding various diet routines of patients and identifies the resources required to satisfy diet orders. Depending on the diet orders and other requests from canteen, the kitchen order plan can be prepared to decide the menu for the day. Analysis of the consumption patterns helps in better and efficient management of the kitchen.

Discharge Summary : The module helps in generating patient's discharge summary, which includes patient's health at the time of discharge, medical history, various diagnosis and drug prescriptions, history of present illness and course in hospital.

Financial Accounting : This module deals with cash/bank, receipts/ payments, journal vouchers, etc. Various books of accounts, such as cashbook, bankbook and ledgers, can be generated and maintained using this module. It can also generate trial balance, balance sheet, and profit and loss statements.

Marketing Module : This module ensures that the hospital gets maximum exposure to the general public and vice versa. This module keeps track of the enquiries made at the reception and follows the lead.

Doctor's Module : This module helps the doctors to keep a track of the entire medical history of a particular patient. Details such as the medicines prescribed, general medical records, previous consultations are all available to the doctor.

HR Management : Various MIS Reports are generated on the above modules for the smooth functioning of the hospital management so that checks can be made on any irregularity done in the hospital.

14.7 Let Us Sum Up :

In this unit we have learnt that the effort of the project manager in investigating in leading project appears in the range from 5 to 100% with overall working time.

Further it is seen that the even those project managers who spent less time for project management activities (5–35%) regarded this time as adequate and sufficient, analysis of the data indicates that less time for project managing and controlling often led to enormous overruns in cost and schedule.

In this unit we have learnt that Rapid Application Development Model reduces the consumption of vehicles in–plant area and provided fast rapid prototyping, extensive responses and fast testing.

In this we have learnt about Agile Scrum Methodology, scrum can help groups complete project deliverables quickly and efficiently. Scrum confirms actual use of time and money. Developments are coded and tested during the run review. Works well for fast–moving development projects. The team gets clear reflectivity through scrum meetings. Scrum approves feedback from customer. The individual work of each group member is observable during daily scrum meetings.

In this we have learnt about the flexibility of prototyping model in design. Easiness of model in error detection. In this model new requirements can be easily accommodated. It is ultimate for online systems.

14.8 Glossary :

1. **Risk –** It is related to potential future harm which appears from some action.
2. **Marginal Planning –** Marginal refers to the focus on the cost or benefit of the next unit or individual. Companies use marginal planning as a decision–making tool to support them maximize their possible profits.
3. **GPS Tracker –** A GPS tracking unit or simply tracker is a direction–finding device generally on a vehicle, asset, person or animal that uses the Global Positioning System (GPS) to decide its association and decide its geographic position to determine its location.
4. **Scrum –** Scrum is a specific Agile methodology that is used to facilitate a project.
5. **Agile –** Agile is a project management philosophy that utilizes a core set of values or principles.

14.9 Assignment :

1. Explain the features show how to manage project by manager in an effective manner.

14.10 Activities :

1. Explain the view of project managers in terms of education provided to team members.

14.11 Case Study :

Compile and explain the necessary steps used in effectively managing a project.

14.12 Further Reading :

1. https://www2.swc.rwth–aachen.de/docs/98_FESMA.pdf?

BLOCK SUMMARY :

In this block, you have learnt and understand about the goals and objective, system scope of system, project estimation, project resource in which people, minimal hardware requirement, minimal software requirements are going to define. We also seen about Risk management in which we learnt scope and intent of RMMM activities and risk description. As a part of risk description, we learn about Business impact risk, Customer risk, Development risk, Employee risk, Process risk, Product size, Technology risk. We also seen about the master schedule, the structure of team and the roles of team members, and tracking and control mechanism.

The block detailed about the basic of project initiation, execution and development in project life cycle. We also seen how requirements are going to gather and for gathering requirement interview method is going to use. We also seen about different development phase like analysis the requirements of the project, Module Analysis, Develop Modules, Integrate Modules, and Test, bug finding and bug fixing.

BLOCK ASSIGNMENT :

❖ **Short Questions :**

1. Explain scope and intent of RMMM activities.
2. Explain organizational role of risk management.
3. Write a note on Project Team Organization.
4. Write a note on Tracking and Control System.
5. Explain determining requirement

❖ **Long Questions :**

1. Explain basic Project Plan of Waste Management System.
2. Write a detailed note on Project Estimates.
3. Discuss : Risk Description
4. Write a detailed note on Project Life Cycle.
5. Write a note on Development Phase.

❖ **Enrolment No. :** ☐

1. How many hours did you need for studying the units ?

Unit No.	1	2	3
No. of Hrs.			

2. Please give your reactions to the following items based on your reading of the block :

Items	Excellent	Very Good	Good	Poor	Give specific example if any
Presentation Quality	☐	☐	☐	☐	_____
Language and Style	☐	☐	☐	☐	_____
Illustration used (Diagram, tables etc)	☐	☐	☐	☐	_____
Conceptual Clarity	☐	☐	☐	☐	_____
Check your progress Quest	☐	☐	☐	☐	_____
Feed back to CYP Question	☐	☐	☐	☐	_____

3. Any other Comments

.....

.....

.....

.....

.....

.....

.....

.....

DR.BABASAHEB AMBEDKAR
OPEN UNIVERSITY

'Jyotirmay' Parisar,
Sarkhej-Gandhinagar Highway, Chharodi, Ahmedabad-382 481.
Website : www.baou.edu.in