Software Engineering

Sub Code : 22413

MSBTE I SCHEME PATTERN
S. Y. DIPLOMA SEM IV
COMPUTER ENGINEERING GROUP
& INFORMATION TECHNOLOGY
(CO/CM/IF/CW)

EDITION : 2020

SOLVED MSBTE PAPERS

- SUMMER 2014
- WINTER 2014
- SUMMER 2015
- WINTER 2015
- SUMMER 2016
- WINTER 2016
- SUMMER 2017
- WINTER 2017
- SUMMER 2018
- Sample Paper (I Scheme)
- WINTER 2019 (I Scheme)

TECHNICAL®

SUBJECT CODE : 22413

As per Revised Syllabus of

MSBTE - I SCHEME

S.Y. Diploma Semester - IV

Computer Engineering Group & Information Technology

(CO / CM / IF / CW)

SOFTWARE ENGINEERING

Anuradha A. Puntambekar

M.E. (Computer)

Formerly Assistant Professor in

P.E.S. Modern College of Engineering,

Pune

Yogesh S. Gunjal

M. Tech (Computer Science and Engineering)

I/C Principal,

JCEI's Jaihind Polytechnic Kuran, Pune

Narendra S. Joshi

M.E. (CSE), Ph.D. (Pursuing),

HOD (Computer Engineering Department),

Assistant Professor,

Sandip Foundation's, Sandip Institute of Polytechnic (SIP),

Nashik

Yogesh B. Patil

M.Tech (I.T.), B.E. (I.T.)

HOD Computer Department,

G.M.Chaudhari Polytechnic, Shahada.

TECHNICAL® PUBLICATIONS

SINCE 1993 An Up-Thrust for Knowledge

SOFTWARE ENGINEERING

Subject Code : 22413

S.Y. Diploma Semester - IV
Computer Engineering Group & Information Technology (CO / CM / IF / CW)

First Edition : January 2019
Second Revised Edition : January 2020

Published by :

TECHNICAL PUBLICATIONS
An Up-Thrust for Knowledge

Amit Residency, Office No.1, 412, Shaniwar Peth, Pune - 411030, M.S. INDIA
Ph.: +91-020-24495496/97, Telefax : +91-020-24495497
Email : sales@technicalpublications.org Website : www.technicalpublications.org

Printer :
Yogiraj Printers & Binders
Sr.No. 10/1A,
Ghule Industrial Estate, Nanded Village Road,
Tal. - Haveli, Dist. - Pune - 411041.

Price : ₹ 100/-

ISBN 978-93-332-0046-2

MSBTE I

PREFACE

The importance of **Software Engineering** is well known in various engineering fields. Overwhelming response to our books on various subjects inspired us to write this book. The book is structured to cover the key aspects of the subject **Software Engineering**.

The book uses plain, lucid language to explain fundamentals of this subject. The book provides logical method of explaining various complicated concepts and stepwise methods to explain the important topics. Each chapter is well supported with necessary illustrations, practical examples and solved problems. All chapters in this book are arranged in a proper sequence that permits each topic to build upon earlier studies. All care has been taken to make students comfortable in understanding the basic concepts of this subject.

Representative questions have been added at the end of each section to help the students in picking important points from that section.

The book not only covers the entire scope of the subject but explains the philosophy of the subject. This makes the understanding of this subject more clear and makes it more interesting. The book will be very useful not only to the students but also to the subject teachers. The students have to omit nothing and possibly have to cover nothing more.

We wish to express our profound thanks to all those who helped in making this book a reality. Much needed moral support and encouragement is provided on numerous occasions by our whole family. We wish to thank the **Publisher** and the entire team of **Technical Publications** who have taken immense pain to get this book in time with quality printing.

Any suggestion for the improvement of the book will be acknowledged and well appreciated.

Authors
A. A. Puntambekar
Yogesh S. Gunjal
Narendra S. Joshi
Yogesh B. Patil

Dedicated to God

SYLLABUS

Software Engineering (22413)

Teaching Scheme			Credit (L+T+P)	Examination Scheme											
				Theory						Practical					
				Paper Hrs.	ESE		PA		Total		ESE		PA		Total	
L	T	P			Max	Min	Max	Min	Max	Min	Max	Min	Max	Min	Max	Min
3	-	2	5	3	70	28	30*	00	100	40	25@	10	25	10	50	20

Unit	Unit Outcomes (UOs) (in cognitive domain)	Topics and Sub - topics
Unit - I **Software Development Process**	1a. Suggest the attributes that match with standards for the given software application. 1b. Recommend the relevant software solution for the given problem with justification. 1c. Select the relevant software process model for the given problem statement with justification. 1d. Suggest the relevant activities in Agile Development Process in the given situation with justification.	1.1 Software, Software Engineering as layered approach and its characteristics, Types of software. l.2 Software development framework. 1.3 Software Process Framework, Process models : Perspective Process Models, Specialized Process Models. 1.4 Agile Software development : Agile Process and its importance, Extreme Programming, Adaptive Software Development, Scrum, Dynamic Systems Development Method (DSDM), Crystal. 1.5 Selection criteria for software process model.
Unit - II **Software Requirement Engineering**	2a. Apply the principles of software engineering for the given problem. 2b. Choose the relevant 'requirement engineering' steps in the given problem. 2c. Represent the 'requirement engineering' model in the given problem. 2d. Prepare SRS for the given problem.	2.1 Software Engineering Practices and its importance, Core principles. 2.2 Communication Practices, Planning Practices, Modelling practices, construction practices, software deployment (Statement and meaning of each principle for each practice). 2.3 Requirement Engineering : Requirement Gathering and Analysis, Types of requirements (Functional, Product, organizational, External Requirements), Eliciting Requirements, Developing Use - cases, Building requirement models, Requirement Negotiation, Validation. 2.4 Software Requirement Specification : Need of SRS, Format, and its Characteristics.

Unit - III **Software Modelling and Design**	3a. Identify the elements of analysis model for the given software requirements. 3b. Apply the specified design feature for software requirements modelling, 3c. Represent the specified problem in the given design notation. 3d. Explain the given characteristics of software testing. 3e. Prepare test cases for the given module.	3.1 Translating Requirement model into design model : Data Modelling. 3.2 Analysis Modelling : Elements of Analysis model. 3.3 Design modelling : Fundamental Design Concepts (Abstraction, Information hiding, Structure, Modularity, Concurrency, Verification, Aesthetics). 3.4 Design notations : Data Flow Diagram (DFD), Structured Flowcharts, Decision Tables. 3.5 Testing - Meaning and purpose, testing methods - Black Box and White - box, Level of testing - Unit testing. 3.6 Test Documentation - Test Case Template, test plan, Introduction to defect report, test summary report.
Unit - IV **Software Project Estimation**	4a. Estimate the size of the software product using the given method. 4b. Estimate the cost of the software product using the given empirical method. 4c. Evaluate the size of the given software using CoCoMo model. 4d. Apply the RMMM strategy in Identified risks for the given software development problem.	4.1 The management spectrum - 4P's. 4.2 Metrics for Size Estimation : Line of Code (LoC), Function Points (FP). 4.3 Project Cost Estimation Approaches : Overview of Heuristic, Analytical, and Empirical Estimation. 4.4 COCOMO (Constructive Cost Model), COCOMO II. 4.5 Risk Management : Risk Identification, Risk Assessment, Risk Containment, RMMM strategy.
Unit - V **Software Quality Assurance and Security**	5a. Use the given scheduling technique for the identified project. 5b. Draw the activity network for the given task. 5c. Prepare the timeline chart / Gantt chart to track progress of the given project. 5d. Describe the given Software Quality Assurance (SQA) activity. 5e. Describe features of the given software evaluation standard.	5.1 Project Scheduling : Basic Principles, Work breakdown structure, Activity network and critical path Method, Scheduling tachniques (CPM, PERT). 5.2 Project Tracking : Timeline charts, Earned Value Analysis, Gantt Charts. 5.3 Software Quality Management vs. Software Quality Assurance Phases of Software Quality Assurance : Planning, Activities, audit, and review. 5.4 Quality Evaluation standards : Six Sigma, ISO for software, CMMI : Levels, Process areas. 5.5 Software Security, Introduction to DevOps, Secure software engineering.

TABLE OF CONTENTS

Unit - I

Unit - II

UNIT- I

1 Software Development Process

1.1 Definition of Software and Software Engineering

MSBTE : Winter-15, Marks 4

Software : Software is nothing but a collection of computer programs and related documents that are intended to provide desired features, functionalities and better performance.

Software Engineering : "Software engineering is a discipline in which theories, methods and tools are applied to develop professional software product."

Attributes of Good Software : There are some essential attributes of good software and those are

(1) **Maintainability :** Sometimes there is a need to make some modifications in the existing software. A good software is a software which can be easily modified in order to meet the changing needs of the customer.

(2) **Usability :** It is the ability of the software being useful. For making the software useful it is necessary that it should have proper GUI and documentation.

(3) **Dependability :** The dependability is a property that includes reliability, security and safety of software. In other words the developed software product should be reliable and safe to use; it should not cause any damage or destruction.

(4) **Efficiency :** The software should be efficient in its performance and it should not waste the memory.

Board Question

1. *State any four attributes of a good software.*

MSBTE : Winter-15, Marks 4

1.2 Software as Layered Approach

MSBTE : Winter-15, 16, Summer-15, 16, 17, Marks 6

- Software engineering is a layered technology. Any software can be developed using these layered approaches. Various layers on which the technology is based are **quality focus layer**, **process layer**, **methods layer**, **tools layer**.
- A disciplined **quality management** is a backbone of software engineering technology.
- **Process layer** is a foundation of software engineering. Basically, process defines the framework for timely delivery of software.
- In **method layer** the actual method of implementation is carried out with the help of requirement analysis, designing, coding using desired programming constructs and testing.
- Software **tools** are used to bring automation in software development process.
- Thus software engineering is a **combination** of process, methods, and tools for development of quality software.

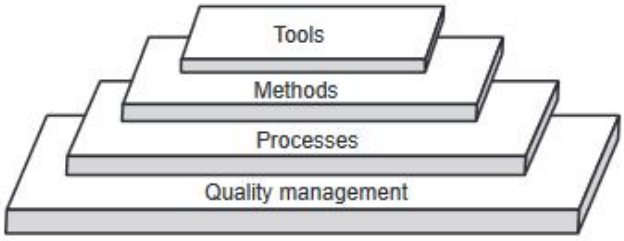


Fig. 1.2.1

Board Questions

1. *Explain software engineering as a layered technology approach.*

MSBTE : Winter-15, Summer-15,17, Marks 4, Summer-16, Marks 6

2. *Explain software engineering as a layered technology approach with neat diagram.*

MSBTE : Winter-16, Marks 4

1.3 Characteristics of Software Engineering

MSBTE : Winter-15, 17, Summer-16, 18, Marks 8

• Software is engineered, not manufactured

1) Software development and hardware development are two different activities.
2) A good design is a backbone for both the activities.
3) Quality problems that occur in hardware manufacturing phase cannot be removed easily. On the other hand, during software development process such problems can be rectified.
4) In both the activities, developers are responsible for producing qualitative product.

• Software does not ware out

1) In early stage of hardware development process the failure rate is very high because of manufacturing defects. But after correcting such defects the failure rate gets reduced.
2) The failure rate remains constant for some period of time and again it starts increasing because of environmental maladies (extreme temperature, dusts, and vibrations).
3) On the other hand software does not get affected from such environmental maladies. Hence ideally it should have an "idealized curve". But due to some undiscovered errors the failure rate is high and drops down as soon as the errors get corrected. Hence in failure rating of software the "actual curve" is as shown below :

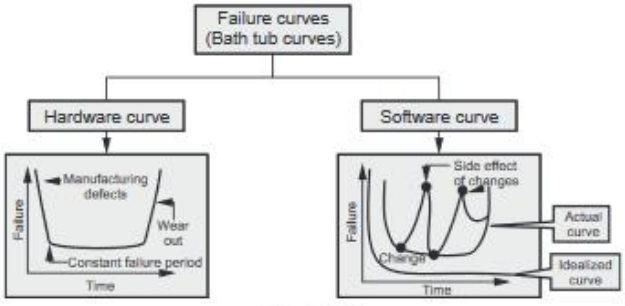


Fig. 1.3.1

4) During the life of software if any change is made, some defects may get introduced. This causes failure rate to be high.
5) Before the curve can return to original steady state another change is requested and again the failure rate becomes high.
6) Thus the failure curve looks like a spike. Thus frequent changes in software cause it to deteriorate.
7) Another issue with software is that there are no spare parts for software. If hardware component wears out it can be replaced by another component but it is not possible in case of software.
8) Therefore software maintenance is more difficult than the hardware maintenance.

• Most software is custom built rather than being assembled from components

1) While developing any hardware product firstly the circuit design with desired functioning properties is created. Then required hardware components such as ICs, capacitors and registers are assembled according to the design, but this is not done while developing software product.
2) Most of the software is custom built.
3) However, now the software development approach is getting changed and we look for reusability of software components.
4) It is practiced to reuse algorithms and data structures.
5) Today software industry is trying to make library of reusable components. For example : In today's software, GUI is built using the reusable components such as message windows, pull down menus and many more such components.
6) The approach is getting developed to use in-built components in the software. This stream of software is popularly known as component engineering.

Board Questions

1. *Describe the characteristics of software.* MSBTE : Winter-15, Marks 4
2. *Define software. State three characteristics of software.* MSBTE : Summer-16, Marks 4
3. *Explain changing nature of software.* MSBTE : Winter-17, Marks 4
4. *What is software ? What are its characteristics ?* MSBTE : Winter-17, Marks 8
5. *Elaborate the software characteristic "Software does not wear out".* MSBTE : Summer-18, Marks 4

1.4 Types of Software

MSBTE : Winter-16, Summer-16, 17, 18, Marks 6

Based on changing nature of software, various types of software are defined as follows -

1. **System software -**
   - It is collection of programs written to service other programs.
   - Typical programs in this category are compiler, editors and assemblers.
   - The purpose of the system software is to establish a communication with the hardware.
2. **Application software -**
   - It consists of standalone programs that are developed for specific business need.
   - This software may be supported by database systems.
3. **Engineering / Scientific software -**
   - This software category has a wide range of programs from astronomy to volcanology, from automatic stress analysis to space shuttle orbital dynamics and from molecular biology to automated manufacturing.
   - This software is based on complex numeric computations.
4. **Embedded software -**
   - This category consists of program that can reside within a product or system.
   - Such software can be used to implement and control features and functions for the end-user and for the system itself.
5. **Web applications -**
   - Web application software consists of various web pages that can be retrieved by a browser.
   - The web pages can be developed using programming languages like JAVA, PERL, CGI, HTML, DHTML.
6. **Artificial Intelligence software -**
   - This kind of software is based on knowledge based expert systems.
   - Typically, this software is useful in robotics, expert systems, image and voice recognition, artificial neural networks, theorem proving and game playing.

Board Questions

1. *State and explain any four types of software.*
   MSBTE : Summer-16, Marks 4
2. *Describe any four categories of software.*
   MSBTE : Winter-16, Marks 4
3. *What is software ? What is embedded software ?*
   MSBTE : Summer-17, Marks 4
4. *Elaborate any six types of software considering the changing nature.*
   MSBTE : Summer-18, Marks 6

1.5 Software Development Framework

- The Software Development Life Cycle (SDLC) is the logical process of developing any system.
- Using SDLC one can develop a system which satisfies customer needs, can be developed within the predefined schedule and cost.
- Normally the system analyst makes use of software development life cycle for developing the information systems.
- The SDLC is a linear or sequential model in which output of previous phase is given as input to next subsequent stage.
- Various phases of software development life cycle are - (Refer Fig. 1.5.1)

1. **Feasibility study** : It is initial phase of software development framework. In this phase, it is decided whether to built the system or not.
2. **Requirement gathering and analysis** : The basic requirements of the software project are identified and analysed in this phase.

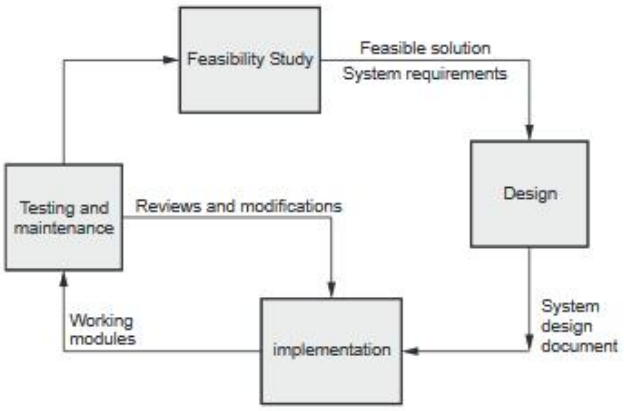


Fig. 1.5.1 Phases in SDLC

3. **Design :** The model of the software system is prepared in this phase.
4. **Coding or implementation :** Using the software design the coding is done in this phase. Thus the implementation model is prepared.
5. **Testing and maintenance :** The code is tested and modified if required.

1.6 Software Process Framework

MSBTE : Winter-15, 16, Summer-15, 17, Marks 4

- The process framework is required for representing the common process activities.
- As shown in Fig. 1.6.1, the software process is characterized by process framework activities, task sets and umbrella activities.

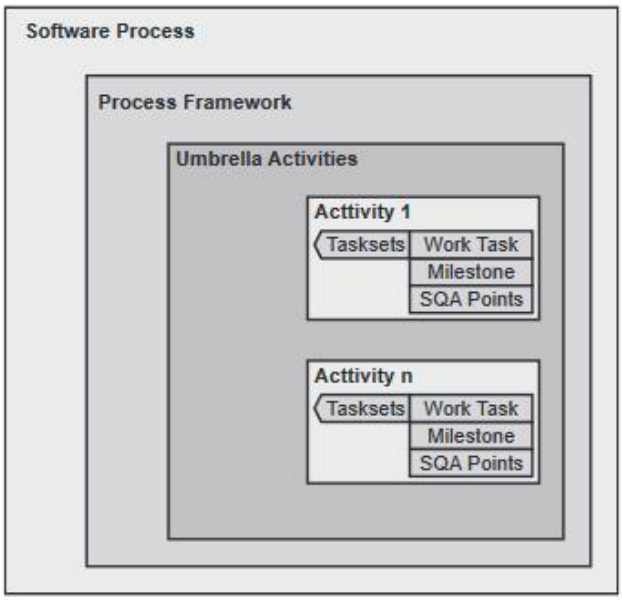


Fig. 1.6.1 Software process framework

Process framework activities

- Communication
  - By communicating customer requirement gathering is done.
- Planning - Establishes engineering work plan, describes technical risks, lists resource requirements, work products produced and defines work schedule.
- Modeling - The software model is prepared by :
  - Analysis of requirements
  - Design
- Construction - The software design is mapped into a code by :
  - Code generation
  - Testing
- Deployment - The software delivered for customer evaluation and feedback is obtained.

Task sets - The task set defines the actual work done in order to achieve the software objective. The task set is used to adopt the framework activities and project team requirements using :

- Collection of software engineering work tasks
- Project milestones
- Software quality assurance points

Umbrella activities - The umbrella activities occur **throughout the process**. They focus on project management, tracking and control. The umbrella activities are

1. **Software project tracking and control** - This is an activity in which software team can **assess progress** and take corrective action to **maintain schedule**.
2. **Risk management** - The **risks** that may affect project outcomes or quality can be **analyzed**.
3. **Software quality assurance** - These are activities required to **maintain** software **quality**.
4. **Formal technical reviews** - It is required to **assess engineering work products to** uncover and **remove errors** before they propagate to next activity.
5. **Software configuration management** - Managing of configuration process when any **change** in the software occurs.
6. **Work product preparation and production** - The activities to create **models**, **documents**, **logs**, **forms** and **lists** are carried out.
7. **Reusability management** - It defines criteria for **work product reuse**.
8. **Measurement** - In this activity, the process can be defined and collected. Also **project and product measures** are used to assist the software team in delivering the required software.

Board Questions

1. *What do you mean by process framework ? Explain with suitable diagram.*

MSBTE : Summer-15,17, Winter-16, Marks 4

2. *Explain the basic process framework activities.*

MSBTE : Winter-15, Marks 4

1.7 Process Model

MSBTE : Winter-15, 16, 17, Summer-14, 15, 17, 18, Marks 8

- **Definition of process model :** The process model can be defined as the abstract representation of process. The appropriate process model can be chosen based on abstract representation of process.
- The software process model is also known as Software Development Life Cycle (SDLC) model or software paradigm.
- Various types of process models are -

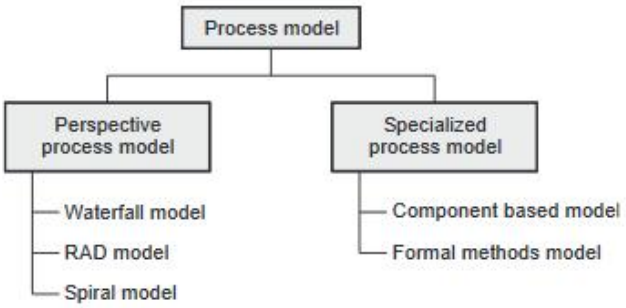


Fig. 1.7.1 Types of process models

1.7.1 Perspective Process Model

1.7.1.1 Waterfall Model

- The waterfall model is also called as 'linear-sequential model' or 'classic life cycle model'.
- The software development starts with requirements gathering phase. Then progresses through analysis, design, coding, testing and maintenance.
- Fig. 1.7.2 illustrates waterfall model.
- In **requirement gathering and analysis** phase the basic requirements of the system must be understood by software engineer.
- The **design** is an intermediate step between requirements analysis and coding. Design focuses on program attributes such as –i) Data structure ii) Software architecture iii) Interface representation iv) Algorithmic details.
- **Coding** is a step in which design is translated into machine-readable form. Programs are created in this phase.
- **Testing** begins when coding is done. The purpose of testing is to uncover errors, fix the bugs and meet the customer requirements.
- **Maintenance** is the longest life cycle phase. When the system is installed and put in practical use then error may get introduced, correcting such errors and putting it in use is the major purpose of maintenance activity. Similarly, enhancing system's services as new requirements are discovered is again maintenance of the system.

Advantages of waterfall model

1. The waterfall model is **simple to implement.**
2. For implementation of **small systems** waterfall model is useful.

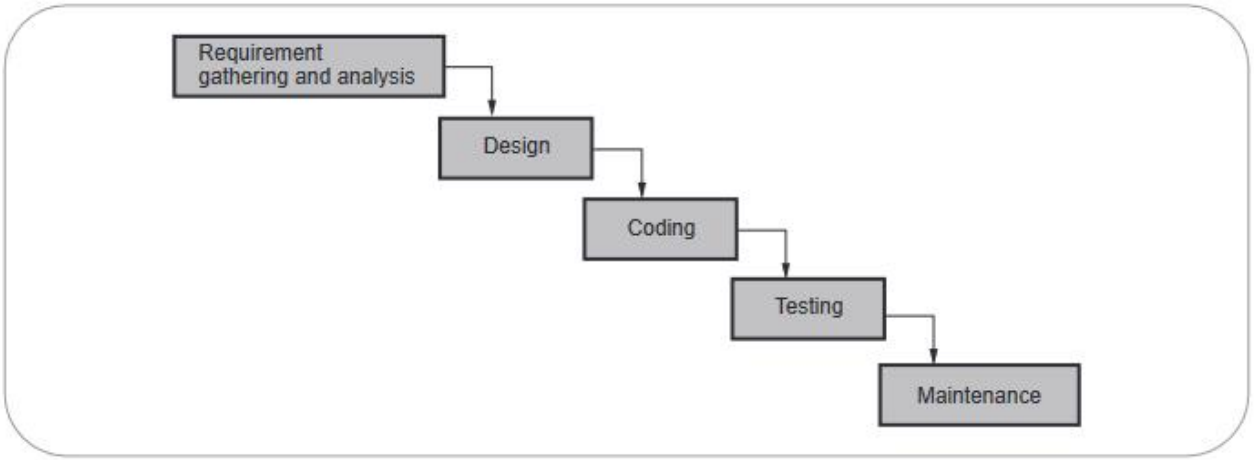


Fig. 1.7.2 Waterfall model

Disadvantages of waterfall model

1. It is difficult to follow the sequential flow in software development process. If some changes are made at some phases then it may cause some confusion.
2. The requirement analysis is done initially and sometimes it is not possible to state all the requirements explicitly in the beginning. This causes difficulty in the project.
3. The customer can see the working model of the project only at the end. After reviewing of the working model; if the customer gets dissatisfied then it causes serious problems.

1.7.1.2 RAD Model (Incremental Model)

- The RAD Model is a type of **incremental process** model in which there is extremely short development cycle.
- When the requirements are fully understood and the component based construction approach is adopted then the RAD model is used.
- Using the RAD model the fully functional system can be developed within 60 to 90 days.
- Various phases in RAD are Requirements Gathering, Analysis and Planning, Design, Build or Construction and finally Deployment.
- Multiple teams work on developing the software system using RAD model parallely.
  - In the **requirements gathering phase** the developers communicate with the users of the system and understand the business process and requirements of the software system.
  - During **analysis and planning phase**, the analysis on the gathered requirements is made and a planning for various software development activities is done.
  - During the **design phase** various models are created. Those models are business model, data model and process model.
  - The **build** is an activity in which working code is generated. This code is well tested by its team. The functionalities developed by all the teams are integrated to form a whole.
  - Finally the **deployment** of all the software components (created by various teams working on the project) is carried out. (Refer Fig. 1.7.3)

Advantages of RAD Model

(1) Faster development cycle.

(2) Visualization of related routines periodically.

(3) Encourages user involvement.

(4) Low maintenance cost.

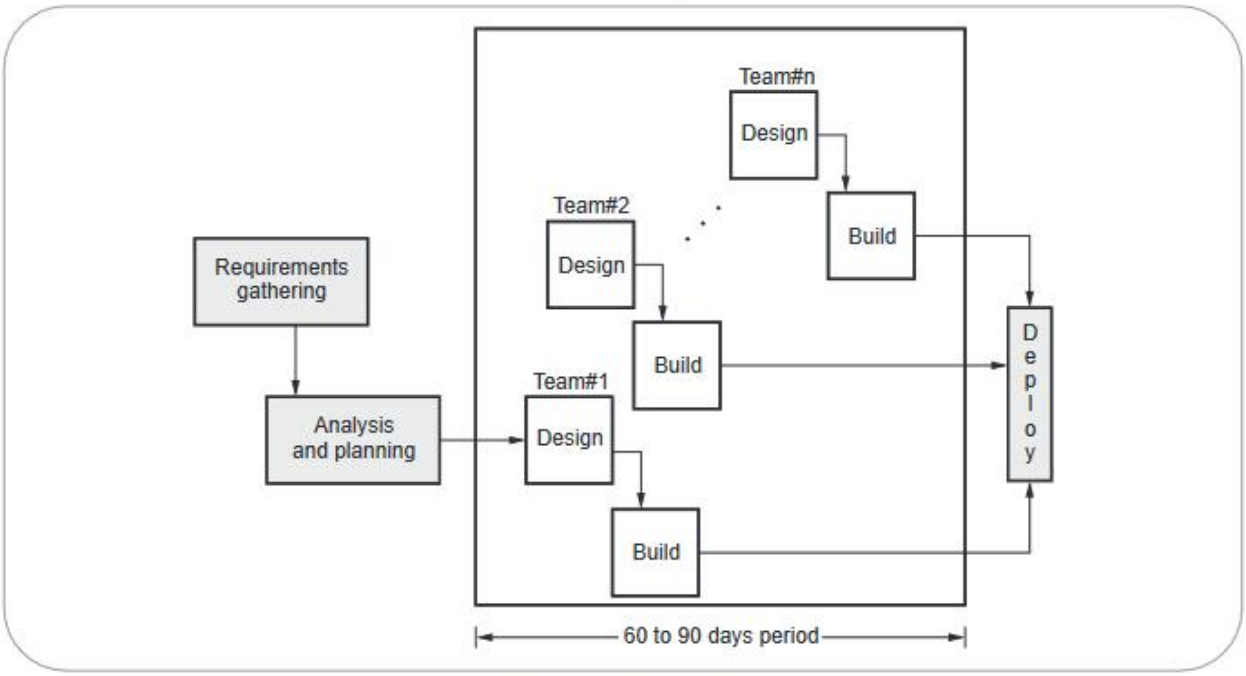


Fig. 1.7.3 RAD model

Disadvantages of RAD Model

(1) It has reduced scalability.

(2) Not appropriate when technical risks are high.

(3) This model requires heavily committed developer and customers. If commitment is lacking then RAD projects will fail.

Difference between Waterfall Model and Incremental Model

Sr. No.	Waterfall model	Incremental model
1	This model is used when requirements are clearly defined.	This model is used when there is possibility of change in requirements.
2	There is no customer interaction until the last phase of the waterfall model.	After each increment, the customer can take a review of the product generated so far.
3	Depending upon the requirements of the project, the human resource is required.	Less human resource is required.
4	Risk of failure of project is high.	Risk of failure of project is low.

1.7.1.3 Spiral Model

- This model possess the iterative nature of prototyping model and controlled and **systematic approaches** of the linear sequential model.
- This model gives efficient development of incremental versions of software. In this model, the software is developed in series of increments.
- The sprial model is divided into a number of **framework activities**. These framework activities are denoted by **task regions**.
- Usually there are six **tasks regions**. The spiral model is as shown in Fig. 1.7.4.
- Spiral model is realistic approach to development of large-scale systems and software. Because customer and developer better understand the problem statement at each evolutionary level. Also risks can be identified or rectified at each such level.
- In the initial pass, product specification is built and in subsequent passes around the spiral the prototype gets developed and then more improved versions of software gets developed.

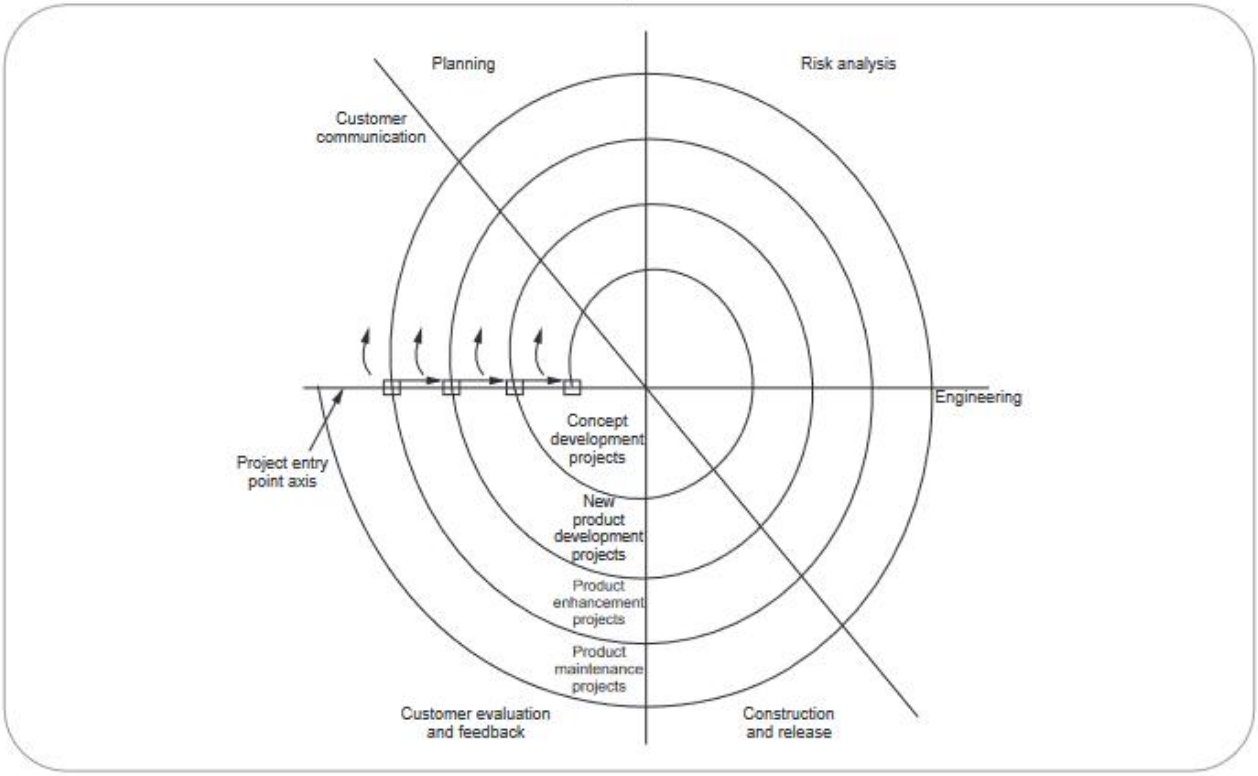


Fig. 1.7.4 Spiral model

- During planning phase, the cost and schedule of software can be planned and adjusted based on feedback obtained from customer evaluation.
- In spiral model, **project entry point axis** is defined. This axis represents starting point for **different types** of **projects**.
- For instance, concept development project will start at core of spiral and will continue along the spiral path. If the concept has to be developed into actual project then at entry point 2 the product development process starts. Hence entry point 2 is called product development project entry point. The development of the project can be carried out in iterations.
- The task regions can be described as :

i) **Customer communication -** In this region, it is suggested to establish customer communication.

ii) **Planning -** All planning activities are carried out in order to define resources time line and other project related activities.

iii) **Risk analysis -** The tasks required to calculate technical and management risks are carried out.

iv) **Engineering -** In this task region, tasks required to build one or more representations of applications are carried out.

v) **Construct and release -** All the necessary tasks required to construct, test, install the application are conducted. Some tasks that are required to provide user support are also carried out in this task region.

vi) **Customer evaluation -** Customer's feedback is obtained and based on customer evaluation required tasks are performed and implemented at installation stage.

- In each region, number of work tasks are carried out depending upon the characteristics of project. For a small project relatively small number of work tasks are adopted but for a complex project large number of work tasks can be carried out.
- In spiral model, the software engineering team moves around the spiral in a clockwise direction beginning at the core.

Advantages of spiral model

1. This model has iterative nature. Hence requirements can be identified at new iteration.
2. Requirement changes can be made at every stage of new version.
3. Risks can be identified and reduced before they get problematic.
4. The working model is available to the customer at certain stage of iteration.

Disadvantages of spiral model

1. This model is based on customer communication. If the communication is not proper then the product being developed is not up to the mark.
2. It demands considerable risk assessment. If the risk assessment is done properly then only successful product can be obtained.

Sr. No.	Waterfall model	Spiral model
1	It requires well understanding of requirements and familiar technology.	It is developed in iterations. Hence the requirements can be identified at new iterations.
2	Difficult to accommodate changes after the process has started.	The required changes can be made at every stage of new version.
3	Can accommodate iteration but indirectly.	It is iterative model.
4	Risks can be identified at the end which may cause failure to the product.	Risks can be identified and reduced before they get problematic.
5	The customer can see the working model of the project only at the end. After reviewing of the working model; if the customer gets dissatisfied then it causes serious problems.	The customer can see the working product at certain stages of iterations.
6	Customers prefer this model.	Developers prefer this model.
7	This model is good for small systems.	This model is good for large systems.
8	It has sequential nature.	It has evolutionary nature.

1.7.2 Specialized Process Model

1.7.2.1 Component Based Development

- The commercial **off-the-shelves** components that are developed by the vendors are used during the software built.
- These components have specialized **targeted functionalities** and **well defined interfaces**. Hence it is easy to **integrate** these components into the existing software.
- The component based development model makes use of various characteristics of **spiral model**. This model is **evolutionary** in nature. That means the necessary changes can be made in the software during the each iteration of software development cycle.
- Before beginning the modeling and construction activity of software development the **candidate component** must be searched and analyzed. The components can be simple functions or can be object oriented classes or methods.
- Following steps are applied for component based development -
- Identify the component based products and analyze them for fitting in the existing application domain.
- Analyze the component integration issues.
- Design the software architecture to accommodate the components
- Integrate the components into the software architecture.
- Conduct comprehensive testing for the developed software.
- **Software reusability** is the major advantage of component based development.
- **The reusability** reduces the development cycle time and overall cost.

1.7.2.2 Formal Methods Model

- This model consists of the set of activities in which the formal mathematical specification is used.
- The software engineers specify, develop and test the computer based systems using the mathematical notations. The notations are specified within the formal methods.
- **Cleanroom software engineering** makes use of the formal method approach.
- The **advantage** of using formal methods model is that it overcomes many problems that we encounter in traditional software process models. **Ambiguity, incompleteness and inconsistency** are those problems that can be overcome if we use formal methods model.

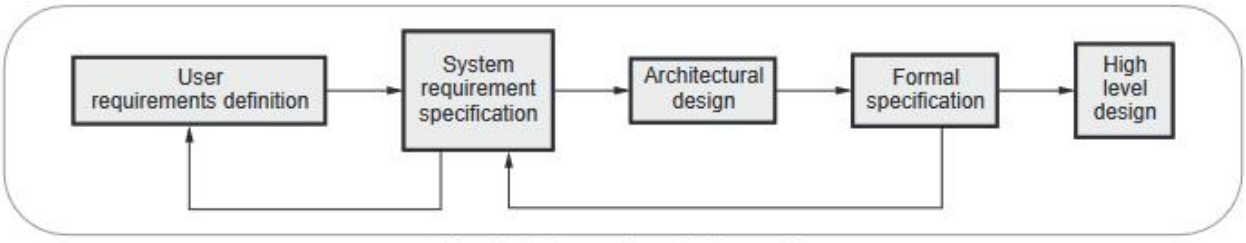


Fig. 1.7.5 Formal methods model

- The formal methods model offers **defect-free** software. However there are some **drawbacks** of this model which resists it from getting used widely. These drawbacks are
- The formal methods model is time consuming and expensive.

- For using this model, the developers need the strong mathematical background or some extensive training.
- If this model is chosen for development then the communication with customer becomes very difficult.

Board Questions

1. *Differentiate between waterfall model and incremental model.*
   MSBTE : Summer-14, 18, Winter-16, Marks 4
2. *Write four drawback of RAD model.*
   MSBTE : Summer-15, Marks 4
3. *In which situation RAD model is applicable ? Give its advantages and disadvantages.*
   MSBTE : Winter-15, Marks 4
4. *Explain spiral model with neat diagram.*
   MSBTE : Summer-15, Marks 4
5. *With neat diagram, explain RAD model with its advantages and disadvantages. (2 each)*
   MSBTE : Winter-16, Marks 6, Summer-17, Marks 8
6. *Explain the waterfall model.*
   MSBTE : Winter-17, Marks 4
7. *Draw the neat labeled diagram of spiral model and list two disadvantages of spiral model.*
   MSBTE : Summer-18, Marks 4

1.8 Agile Software Development

MSBTE : Winter-15, 16, 17, Summer-15, 16, 17, Marks 4

- The agile processes are the **light-weight methods** are **people-based** rather than plan-based methods.
- The agile process forces the development team to **focus on software** itself rather than design and documentation.
- The agile process make use of **iterative method.**
- The aim of agile process is to deliver the working model of **software quickly to the customer.**

Agile Principles

- There are famous 12 principles used as agility principles -

1. Satisfy the customer by early and continuous delivery of valuable software.
2. The changes in the requirements must be accommodated. Even though the changes occur late in the software development process, the agile process should help to accommodate them.
3. Deliver working software quite often. Within the shorter time span deliver the working unit.
4. Business people and developers must work together throughput the project.
5. Motivate the people who are building the projects. Provide the environment and support to the development team and trust them for the job to be done.
6. The working software is the primary measure of the progress of the software development.
7. The agile software development approach promote the constant project development. The constant speed for the development of the product must be maintained.
8. To enhance the agility there should be continuous technical excellence.
9. Proper attention to be given to technical excellence and good design.
10. Simplicity must be maintained while developing the project using this approach.
11. The teams must be the self-organizing team for getting best architecture, requirements and design.
12. At regular intervals the team thinks over the issue of becoming effective. After the careful review the team members adjusts their behavior accordingly.

1.8.1 Agile Process and its Importance

Agile process is based on following assumptions about software projects

1. It is difficult to predict the software requirements in advance. Similarly the customer priority often get changed.
2. It is difficult to predict how much design is necessary before the implementation.
3. All the software development activities such as analysis, design, construction and testing are just difficult to predict.

Characteristics of agile process are -

1. Agile processes must be **adaptable to technical and environmental changes.** That means if any technological changes occur, then the agile process must accommodate it.
2. The development of agile processes must be **incremental**. That means, in each development the increment should contain some functionality that can be tested and verified by customer.
3. The **customer feedback** must be used to create the next increment of the process.
4. The software increment must be **delivered in short span of time**.
5. It must be **iterative**, so that each increment can be evaluated regularly.

Features of agile process

The features of agile process models

The key features of an agile process model can be summarised as follows :

- The software itself is the important measure of the team's progress, rather than documentation.
- The development **team has autonomy** to determine how to structure and handle the development work.
- Changes can be easily adapted.
- Customers can more easily examine the software and provide feedback.

Merits :

1) Customer satisfaction can be attained by rapid and continuous delivery of useful software.
2) Customer, developer and tester interact with each other during software development process.
3) Continuous attention can be given for excellent technical design and software quality.
4) Even late changes in requirements can be accommodated.

Demerits :

1) There is lack of emphasis on necessary designing and documentation during software development process.
2) The project can easily get off the track if customer is not clear about his requirements.

Difference between Prescriptive Process Model and Agile Process Model

Sr. No.	Prescriptive process model	Agile process model
1	It is a traditional approach of software development.	It is a modern approach of software development.
2	It is product oriented process model	It is people oriented process model.
3	Sometimes it is difficult to adapt changes during the software development.	It is easy to adapt the changes during software development.
4	Some models allow less involvement of customers.	Customer involvement is important feature of agile process model.
5	Lengthy cycles of software development may cause delayed delivery of the product.	Quick delivery of the product.
6	Emphasis is on documentation.	Less emphasis of documentation during software development.
7	Examples – Waterfall model, spiral model.	Examples – Extreme programming, Scrum.

Process Models

There are various agile process models -

1. Extreme Programming
2. Adaptive Software Development
3. Dynamic System Development Method(DSDM)
4. Scrum
5. Crystal

1.8.2 Extreme Programming

Extreme Programming (XP) is one of the best known agile methods. The extreme programming approach was suggested by Kent Beck in 2000. The extreme programming process is explained as follows -

- Customer specifies and priorities the system requirements. Customer becomes one of the important members of development team. The developer and customer together prepare a **story-card** in which customer needs are mentioned.
- The developer team then aims to implement the scenarios in the story-card.
- After developing the story-card the development team breaks down the total work in **small tasks**. The efforts and the estimated resources required for these tasks are estimated.
- The customer priorities the stories for implementation. If the requirement changes then sometimes unimplemented stories have to be discarded. Then release the complete software in **small** and frequent **releases**.
- For accommodating new changes, **new story-card** must be developed.
- **Evaluate** the system along with the customer.

This process is demonstrated by the following Fig. 1.8.1.

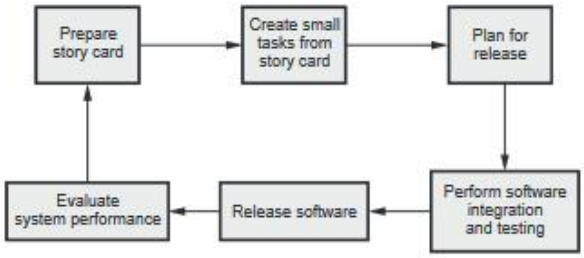


Fig. 1.8.1 Extreme programming release cycle

1.8.3 Adaptive Software Development

- The adaptive software development approach was proposed by Jim Highsmith. This approach is useful in building the complex software systems using iterative apporach
- The focus of this method is on **working in collaboration** and **team self organization.**

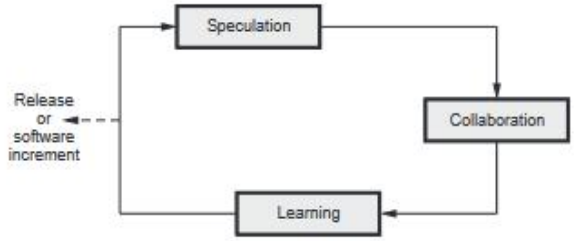


Fig. 1.8.2 Adaptive software development life cycle

- The life cycle of ASD consists of **three phases of software development** and those are -

  1. Speculation 2. Collaboration 3. Learning.

1. **Speculation :** This is an initial phase of the adaptive software development process. In this phase the **adaptive cycle planning** is conducted. In this cycle planning mainly three types of information is used such as - Customer's mission statement, project constraints (delivery date, user description, budgets and so on) and basic requirements of the project.
2. **Collaboration :** The motivated people work in collaboration to develop the desired software product. In this phase **collaboration among the members** of development team is a key factor. For successful collaboration and coordination it is necessary to have following qualities in every individual -
   - Assist each other without resentment
   - Work hard.
   - Posses the required skill set.
   - Communicate problems and help each other to accomplish the given task.
   - Criticize without any hate.
3. **Learning :** As the team members go on developing the components, the emphasize is on learning new skills and techniques. There are three ways by which the team members learn -
   - **Focus groups :** The feedback from the end-users is obtained about the software component being developed. Thus direct feedback about the developed component can be obtained.
   - **Formal technical review :** This review for software components is conducted for better quality.
   - **Postmortems :** The team analyses its own performance and makes appropriate improvements.

1.8.4 SCRUM

- SCRUM is an **agile process model** which is used for developing the complex software systems.

- It is a **lightweight process framework** that can be used to manage and control the software development using **iterative and incremental** approach. Here the term **lightweight** means the overhead of the process is kept as small as possible in order to maximize productive time.
- This model is developed by Jeff Sutherland and Ken Schwaber in 1995.

Principles

- Various **principles** using which the SCRUM works are as given below -

1. There are **small working teams** on the software development projects. Due to this there is maximum communication and **minimum overhead.**
2. The tasks of people must be partitioned into small and clean **packets or partitions.**
3. The process must **accommodate** the technical or business **changes** if they occur.
4. The process should produce **software increments**. These increments must be inspected, tested, documented and built on.
5. During the product building the constant **testing and documentation** must be conducted.
6. The SCRUM process must **produce the working model** of the product whenever demanded or required.

- Various development activities (requirements analysis, design, evolution and delivery) are guided by SCRUM principles.

Development Activities

In SCRUM emphasize is on **software process pattern.** The software process pattern defines a set of development activities. Refer Fig. 1.8.3.

Various development activities in SCRUM are -

1. **Backlog :** It is basically **a list** of project requirements or features that must be provided to the customer. The items can be included in the backlog list at any time. The product manager analyses this list and updates the priorities as per the requirements.
2. **Sprint :** These are the **work units** that are needed to achieve the requirements mentioned in the backlogs. Typically the sprints have **fixed duration or time-box** (typically of 2 to 4 weeks). Thus sprints allow the team members to work in stable and short-term environment.
3. **Meetings :** These are 15 minutes daily meetings to report the completed activities, obstacles and plan for next activities. Following are three questions that are mainly discussed during the meetings
   i) What are the tasks done since last meeting ?
   ii) What are the issues (obstacles) that team is facing ?
   iii) What are the next activities that are planned ?
4. **Demo :** During this phase, the software increment is delivered to the customer. The implemented functionality which is demonstrated to the customer. Note that demo focuses on only implemented functionalities and not all the planned functionalities (and yet to get implemented) of the software product.

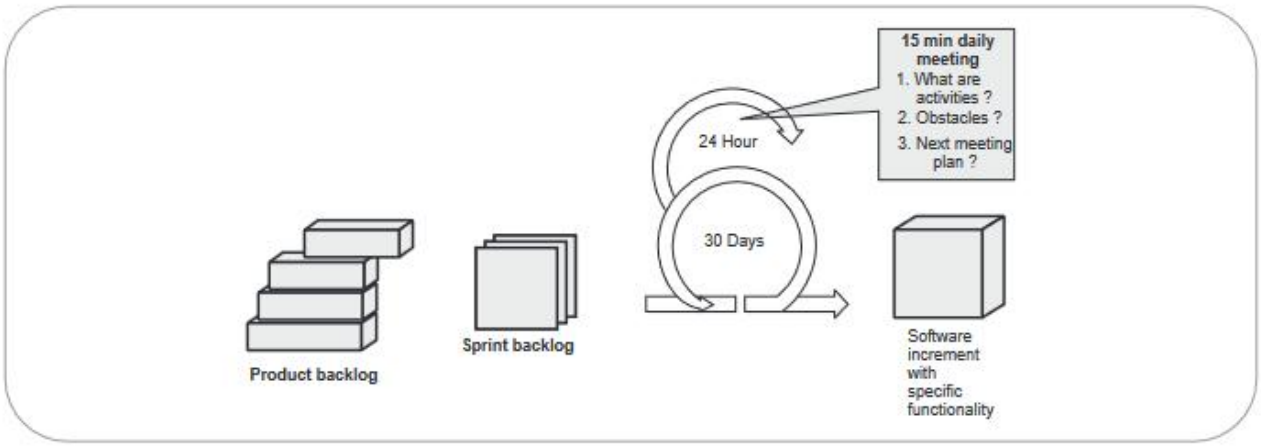


Fig. 1.8.3 SCRUM workflow activities

Roles :

1. **Scrum Master -** The Scrum master leads the meeting and analyses the response of each team member. The potential problems are discussed and solved in the meeting with the help of master.
2. **Team Members -** These are the persons working in a team to develop the software solutions.

Advantages and Disadvantages :

Advantages :

1. SCRUM model brings **transparency** in project development status.
2. It provides **flexibility** towards the changes.
3. There is **improved communication, minimum overhead** in development process.
4. The **productivity** can be **improved**.

Disadvantages :

1. Some decisions are hard to track in fixed time span.
2. There are problems to deal with non-functional requirements of the system.

1.8.5 Dynamic System Development Method (DSDM)

In this agile method, the project deadline is met using the incremental prototyping approach. This is an iterative development process.

The Dynamic System Development Method (DSDM) consortium has defined an agile process model called **DSDM life cycle**.

Various phases in this life cycle model are as follows -

1. **Feasibility study :** By analyzing the business requirements and constraints the viability of thee application is determined in this phase.
2. **Business study :** The functional and informational requirements are identified and then the business value of the application is determined. The basic application architecture is decided in this phase.
3. **Functional model iteration :** The incremental approach is adopted for development. The basic functionalities are demonstrated to the customer by building the suitable increments. The intention of iterative cycle is to gather additional requirements by eliciting the requirements from the customer as each prototype is being developed.
4. **Design and build iteration :** Each prototype is revisited during the functional model iteration to ensure that the business requirements are satisfied by each software component. Sometimes if possible, the design and build activities can be carried out in parallel.
5. **Implementation :** In this phase, the software increment is placed in the working environment. If changes are suggested or if the end-user feels it incomplete then the increment is placed in iteration for further improvement.

The DSDM can be combined with XP method or ASD concepts to create a combination model.

1.8.6 Crystal

- Cockburn and Highsmith suggested the **crystal family** of agile methods.
- The primary goal of this method is to deliver useful and working software.
- In this model, a set of methodologies are defined which contains the core elements that are common to all. These methodologies also contain roles, process patterns, work products and practice that are unique to each.
- Thus the crystal family is actually a set of agile processes that are useful for **different types of projects**. The agile team has to select the memebers of the crustal family that is most approapriate for their ongoing project and environment.

Board Questions

1. *Describe Agile process models in detail.* **MSBTE : Summer-15, Marks 4**
2. *Explain the features of Agile software development approach.* **MSBTE : Winter-15, 16, Marks 4**

3. *Differentiate between Prescriptive Process Model and Agile Process Model (any four points).* **MSBTE : Summer-16, Winter-17, Marks 4**
4. *Explain the term scrum.* **MSBTE : Summer-17, Marks 4**
5. *What is agile process ?* **MSBTE : Summer-17, Marks 4**

1.9 Selection Criteria for Software Process Model

MSBTE : Summer-16, Marks 4

Following table shows the situations in which particular process model can be used

Type of the project	Suggested model
• If a small project is to be implemented. • If the requirements of the project are well understood. • Existing manual system has to be automated. • If there is no need for customer involvement in the project development cycle.	Waterfall model
• When project requirements are not clear. • The system will be operated by novice users. • The GUI of the project is very important. • Delivery of the project is expected within a short period of time.	RAD model
• The risk of long project is not affordable. • Requirements are not known and will be known only with time. • Project is of large size.	Spiral model
• When the requirements are not properly known. • For the systems in which customer involvement is must. • The GUI of project is important. • Quick delivery of the product is expected.	Agile model

Board Question

1. *What is Waterfall Model ? State the practical situations in which it can be used.* **MSBTE : Summer-16, Marks 4**

❑❑❑

Notes

UNIT - II

2 Software Requirement Engineering

Part I : Software Engineering Practices

2.1 Software Engineering Practices and its Importance

Definition : Software engineering practices includes - concepts, principles, methods and tools that must be considered for planning and development of software system.

Importance :

1) By following software engineering practices, every concerned entity gets involved in software development process.
2) The software engineering practices - provide detailed insight for software development process.
3) It acknowledges the software engineer about the principles that must be used during the software development.

Essence of Software Engineering Practice :

- The problem solving activity is normally based on following four steps -
  1. Understanding of the problem (Communication and requirement analysis).
  2. Planning for possible solution (Modelling and design).
  3. Execute the plan (Code generation).
  4. Check the accuracy of the solution (Testing and quality assurance).

2.2 Core Principles

MSBTE : Winter-15, 16, 17, Summer-15, 17, Marks 8

- David Hooker proposed seven core principles for software engineering practice. These are as follows -

Principle 1. Reason it all Exists : The software system that you are going to develop must give value to its users. If your development is not going to add any value to its users then don't develop such system.

Principle 2. Keep it Simple, Stupid (KISS) : The software design must be simple, easy for understanding and easy to maintain.

Principle 3. Maintain the Vision : There should be clear vision for the software system to be built. The clear vision about the system helps to develop it without any ambiguity.

Principle 4. What you will Produce, others will Consume : The software system that you develop will be used by users, software designers, programmers, testers and so on. So whenever you develop the system, develop in such a way that your job will be easier for them to handle.

Principle 5. Be Open to the Future : Develop a system in such a way that it will have longer life time. Develop the system in such a manner that it will adapt any changes comfortably. Never design the system for specific problems only, rather create it for general purposes.

Principle 6. Plan ahead for Reuse : Reusability has got vital importance in software industry. The reusability can be achieved using the approaches like object oriented programming. The software must be developed and documented in such a way that reusability can be easier one.

Principle 7. Think : Think about the system that you are going to develop and acquire the required

knowledge. Thoughtfully, apply above six principles during the software development process.

Board Questions

1. *State and describe various core principles of software engineering.*
   MSBTE : Winter-15, 16, Marks 8
2. *List core principle of Software Engineering.*
   MSBTE : Summer-15, 17, Winter-17, Marks 4

2.3 Communication Practices

MSBTE : Winter-15,16,17, Marks 4

- Communication is carried out in order to understand the customer requirements. Following are the principles that are used to make the communication effective -

Principle 1 - Listen : It is important to listen the customer carefully. Ask for the clarifications appropriately. Never interrupt him/her annoyingly during the narration.

Principle 2 - Prepare for communication : Do some research to understand the business domain. If you are conducting the meeting, then prepare the agenda before the meeting.

Principle 3 - Have facilitator for communication : The facilitator or a leader is required for the communication for three reasons. Firstly to move the conversation in productive direction, secondly to resolve any conflicts and thirdly to ensure that the designated principles and standards are being followed.

Principle 4 - Have face-to-face communication : The face to face communication is important to have effective communication.

Principle 5 - Take notes and document the decisions : It is important to note down the important points and decisions made during the communication.

Principle 6 - Strive for collaborations : The collective knowledge of the members of the team is combined to understand the system. This makes a small collaboration which help the team members in deciding the goals of the software system being developed.

Principle 7 - Stay focused : During the communication and discussion, it is possible to switch frequently from one topic to another. The leader or facilitator of the communication must keep the communication focused and must make every topic modular. The modular discussion means leave one topic only after resolving the issues within in it and then only switch to another topic.

Principle 8 - Make pictorial representation : If something is unclear, then make a drawing or sketch of it for understanding.

Principle 9 - Move on : During the communication just move on by resolving the issues and by understanding the things.

Principle 10 - Negotiate : There are times when the developers and customer need to negotiate on some of the issues such as some functionalities, delivery date, features priorities and so on.

Board Questions

1. *Briefly describe the principles of communication.*
   MSBTE : Winter-15, Marks 4
2. *State and describe six principles of communication practices.*
   MSBTE : Winter-16, Marks 4
3. *What are communication principles ? Explain their meaning.*
   MSBTE : Winter-17, Marks 4

2.4 Planning Practices

MSBTE : Summer-15, 17, 18, Marks 8

- Planning is an activity that includes the set of management and technical practices that software has to follow in definite direction. Various principles of planning are -

Principle 1 - Understand the scope of the project : The scope of the project help the software team what is the goal of development.

Principle 2 - Make the planning by involving the customer : Customers specify what they exactly want from the software system. The developer can negotiate order of delivery, some unrealistic functionalities, cost and so on. Hence customers involvement in the project is must.

Principle 3 - Planning activity should be iterative : Iterations help the developer to accommodate the changes in the system plan.

Principle 4 - Make estimation based on the knowledge : Estimation of the project represents the efforts, cost and duration based on current understanding of the work. But it is always vague.

Principle 5 - Consider risk while defining the plan : The project plan must be flexible enough to accommodate one or more risks.

Principle 6 - Be realistic : Software development can not be 100 % perfect, there can be more or less ambiguities and omissions, budgets and schedule may vary, developers may make mistakes and so on. Such things might occur during the planning of the system.

Principle 7 - Adjust granularity according to plan : Granularity means how much detailed is your project plan. The fine granularity plan provides more work plan details planed relatively short time increment. On the other hand, the coarse granular plan provide broader work task planed over long period. The plan must be flexible enough for making the adjustments about the granularity of the project.

Principle 8 - Ensure the quality : The plan must help the software team to induce quality in their development.

Principle 9 - Describe the accommodated changes : The plan must help the software team to induce required changes in their development.

Principle 10 - Track the plan frequently and make the changes as per the requirements : The project plan must be adjusted frequently.

Board Questions

1. *Describe eight principles of good planning.*
   MSBTE : Summer-15, 17, Marks 8
2. *Explain principles of planning practices in software engineering (any four)*
   MSBTE : Summer-18, Marks 4

2.5 Modeling Practices

MSBTE : Winter-17, Marks 4

- Models are created for understanding the system. During software development, there are two kinds of models that are developed - analysis model and design model.

(1) Analysis Modeling Principles

Following are analysis modeling principles -

1. **The information domain of the problem must be represented and understood :** The information domain represents the data that flows in and out of the system. For designing the analysis model it is necessary to understand this data flow.
2. **The functionality of the software must be performed :** There are various types off the functionality that must be present in the system. Some functions can be directly beneficial to the user. Some functions are control functions, some functions are for transforming the data and so on.
3. **The behavior of the software must be represented by the model :** By creating appropriate model the behavior of the computer system for the input submitted by the user, interaction of the system with the external system are represented by the software.
4. **The model should be created in such a way that it represents the details of the software in layered or hierarchical manner :** Software systems are usually created to solve the complex problems. The large and complex problems are solved using divide and conquer strategy. These sub-problems are relatively easy to understand. This process of dividing the problems into smaller sub-problems is called partitioning.
5. **The analysis task must move from basic information to the implementation of the concept :** Before creating the analysis model the information gathered is from users' point of

view. This information should be transformed in such a way that the implementation of the concept using the computer based system.

(2) Design Modeling Principles

Following are design modeling principles -

1. **Analysis model must be used to create design model** : The analysis model describes the information domain, system behavior, user visible functionalities and so on. The design model translates this information into the system architecture. Hence the elements of design model should be traceable to analysis model.
2. **Consider the architecture of the system to be built** : The skeleton of the system should be prepared before creating the actual design of the system. For creating such architecture the information present in the analysis model is used.
3. **Design of data is an important activity** : The data design is an important element of the architectural design. It represents the flow of information.
4. **Interfaces must be designed carefully** : The interfaces are important for communication of two components and communication with the external environment. Hence these need to be designed carefully.
5. **User interface designs must be as per the needs** : User interfaces basically assist the users to interact with the system via user interfaces. Hence the design of user interface must be by considering the users needs.
6. **Component level design must be functionally independent** : Functional independence is a quality that indicates the single mindedness of software component. Ideally every component must focus on one functionality at a time. This is called cohesiveness of the components. During component level design high cohesiveness (functionally independence) is required.
7. **Components must be loosely coupled** : If the components are tightly coupled then error propagation might get increased. The maintaining such systems become difficult. This denotes low coupling property of design.
8. **Design representations must be easy to understand** : The software design is used to generate the code. Hence it must be easy to understand so that implementation of the system can be effectively done.
9. **Design development must be iterative** : The iterative development of the design model will refine the work and errors can be corrected in each iteration. But each after each iteration it becomes difficult to keep the software design simple.

Board Question

1. *Explain modeling practice in software engineering with principles.* **MSBTE : Winter-17, Marks 4**

2.6 Construction Practices

MSBTE : Summer-16, Marks 4

- The construction practices are based on - Preparation principles, coding principles and validation principles.

(1) Preparation principles

These are some important principles that software developers must follow before writing the code.

1. Firstly, understand the problem for which the system is designed.
2. Know the basic design principles and concepts.
3. Choose appropriate programming language for coding.
4. Select such a programming environment which is convenient to work.
5. Prepare the set of unit tests for each component of the code.

(2) Coding principles

These are some important principles that software developers must follow while writing the code.

1. Adopt structured or modular programming approach for the algorithm.
2. Choose the appropriate data structures.

3. By understanding the system architecture, create appropriate interfaces.
4. Keep conditional logic very simple.
5. Create the nested loop in such a way that they can be tested easily.
6. Give meaningful variable names.
7. Make the code documented by adding appropriate comment statements within it.
8. Make use of indentation and blank lines in between the code so that appropriate visual layout of the code can be created.

(3) Validation principles

These are some important principles that software developers must follow after completion of writing the code.

1. Review the code thoroughly.
2. Perform unit tests and correct the errors.
3. Refactor the code.

Board Question

1. *What is software coding ? State three principles of code validation.* **MSBTE : Summer-16, Marks 4**

2.7 Software Deployment

MSBTE : Summer-15, 16, 17, Winter-15, Marks 8

- During deployment of the software component three activities are carried out - i) Delivery ii) Support and iii) Feedback.
- Following are the set of principles that must be followed while delivering the software increment/release to the customer.
  1. **Manage or fulfill the customer expectations :** Quiet often customers expect too much from the delivered software product. To avoid this have a clear communication with the customer about the functionalities of the product.
  2. **The delivery package must be assembled and well tested :** Provide CD-ROM or other media to the customer containing all the executables of the software components.
  3. **The support service must be ready before delivery of the software system :** Assist the user to handle the software system appropriately. Moreover, if any problem or query occurs then help the end-user to solve it.
  4. **Appropriate instructional material must be given to the end user :** Provide the user with instruction manuals, help material, training aids, demo and so on for handling the system properly.
  5. **Do not deliver erroneous software :** Never provide the low quality or erroneous software to the customers.

Board Questions

1. *Explain deployment principle.* **MSBTE : Summer-15, Winter-15, Marks 4**
2. *What is Software deployment ? State the principles to be followed while preparing to deliver the software increment.* **MSBTE : Summer-16, Marks 8**
3. *What is meant by software deployment ?* **MSBTE : Summer-17, Marks 4**

Part II : Requirement Engineering

2.8 Requirement Gathering and Analysis

MSBTE : Summer-15, 17, 18, Winter-15, 17, Marks 8

- Requirement engineering is the process of establishing the services that the customer requires from a system. And the constraints under which it operates and is developed.

Requirement Engineering Tasks

- Requirement engineering is the process **characterized** for achieving following **goals -**
  - Understanding customer requirements and their needs
  - Analyzing the feasibility of the requirement
  - Negotiating the reasonable solutions
  - Specification of an unambiguous solution.
  - Managing all the requirements of the project
  - Finally transforming the requirements into the operational systems
- Requirement engineering process performs following **seven distinct functions -**

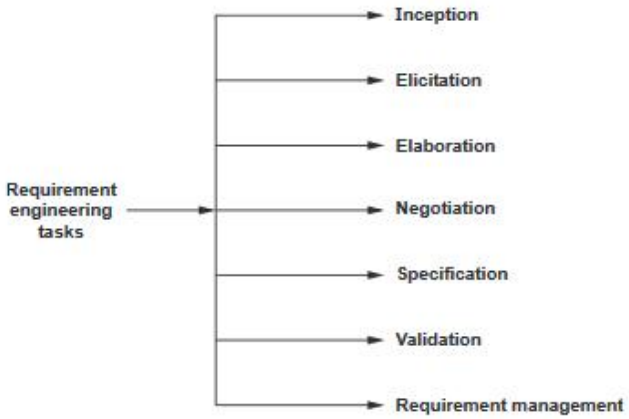


Fig. 2.8.1 Requirement engineering tasks

Let us now discuss these tasks in detail -

2.8.1 Inception

- The inception means specifying the **beginning** of the software project. Most of the software projects get started due to business requirements. There may be potential demand from the market for a particular product and then the specific software project needs to be developed.
- There exist several **stakeholders** who define the **business ideas.** Stakeholders mean an **entity** that takes active participation in project development. In software project development, the stakeholders that are responsible for defining the ideas are business managers, marketing people, product managers and so on. Their role is to do rough feasibility study and to identify the scope of the project.
- During the **inception** a set of **context free questions** is discussed. The purpose of inception is to -
  1. Establish the basic understanding of the project.
  2. Find out all possible solutions and to identify the nature of the solution.
  3. Establish an effective communication between developer and the customer.

2.8.2 Elicitation

- Before the requirements can be analyzed and modelled they must undergo through the process of elicitation process. Requirements elicitation means requirements discovery. Requirements elicitation is very difficult task.

2.8.3 Elaboration

- Elaboration is an activity in which the information about the requirements is **expanded** and **refined.** This information is gained during inception and elicitation.
- The **goal** of elaboration activity is to prepare a technical model of software functions, features and constraints.

2.8.4 Negotiation

- Sometimes customer may **demand for more** than that is achieved or there are certain situations in which customer demands for something which cannot be achieved in **limited business resources.** To handle such situations requirement engineers must convince the customers or end users by solving various **conflicts.** For that purpose, requirement engineers must ask the customers and stakeholders to **rank** their **requirements** and then priority of these requirements is decided. Using **iterative approach** some requirements are eliminated, combined or modified. This process continues until the users' satisfaction is achieved.

2.8.5 Specification

- A specification can be a written document, mathematical or graphical model, collection of use case scenarios or may be the prototypes.
- There is a need to develop a **standard specification** in which requirements are presented in consistent and understandable manner.
- For a large system it is always better to develop the specification using natural language and in a written document form. The use of graphical models is more useful for specifying the requirements.
- Specification is the final work product of requirement engineering process. It describes the functions, constraints and performance of computer based systems.

2.8.6 Validation

- Requirement Validation is an activity in which requirement specification is analyzed in order to ensure that the requirements are specified unambiguously. If any inconsistencies, omissions

and errors are identified then those are corrected or modified during the validation.

- The most commonly used requirement validation mechanism is **Formal Technical Review (FTR).** In FTR, the review team validates the software requirements. The review team consists of requirement engineers, customers, end users, marketing person and so on. This review team basically identifies conflicting requirements, inconstancies or unrealistic requirements.

2.8.7 Requirement Management

Definition : Requirements management is the process of managing changing requirements during the requirements engineering process and system development.

Why requirements get change ?

- Requirements are always incomplete and inconsistent. New requirements occur during the process as business needs change and a better understanding of the system is developed.
- System customers may specify the requirements from business perspective that can conflict with end user requirements.
- During the development of the system, its business and the technical environment may get changed.

Board Questions

1. *List seven tasks of requirement engineering.*
   MSBTE : Summer-15, Marks 4
2. *With reference to requirement engineering, explain i) Inception and ii) Elicitation*
   MSBTE : Winter-15, Marks 4
3. *Explain following requirements engineering tasks : i) Negotiation ii) Specification*
   MSBTE : Summer-17, Marks 4
4. *What are major tasks of requirement engineering ?*
   MSBTE : Winter-17, Marks 8
5. *Explain following requirements of engineering tasks : i) Negotiation ii) Validation*
   MSBTE : Summer-18, Marks 4

2.9 Types of Requirements

Functional and Non-Functional Requirements

- Software system requirements can be classified as functional and non functional requirements.

2.9.1 Functional Requirements

- **Purpose :** Functional requirements should describe all the required **functionality** or **system services.**
- The customer should provide statement of service. It should be clear how the system should react to particular inputs and how a particular system should behave in particular situation.
- Functional requirements are heavily dependent upon the type of software, expected users and the type of system where the software is used.
- Functional user requirements may be high-level statements of what the system should do but functional system requirements should describe the system services in detail.
- **For example :** Consider a library system in which there is a single interface provided to multiple databases. These databases are collection of articles from different libraries. A user can search for, download and print these articles for a personal study.
- From this example we can obtain functional requirements as,
  1. The user shall be able to search either all of the initial set of databases or select a subset from it.
  2. The system shall provide appropriate viewers for the user to read documents in the document store.
  3. A unique identifier (ORDER_ID) should be allocated to every order. This identifier can be copied by the user to the account's permanent storage area.

2.9.1.1 Problems Associated with Requirements

- **Requirements imprecision**
  1. Problems arise when requirements are not precisely stated.
  2. Ambiguous requirements may be interpreted in different ways by developers and users.
  3. Consider meaning of term 'appropriate viewers'

- **User intention** - Special purpose viewer for each different document type;
- **Developer interpretation** - Provide a text viewer that shows the contents of the document.
- **Requirements completeness and consistency** -

  The requirements should be both complete and consistent. **Complete** means they should include descriptions of all facilities required. **Consistent** means there should be no conflicts or contradictions in the descriptions of the system facilities.

  Actually in practice, it is impossible to produce a complete and consistent requirements document.

2.9.2 Non Functional Requirements

- **Purpose** : The non functional requirements define **system properties** and **constraints.**

  Various properties of a system can be : Reliability, response time, storage requirements. And constraints of the system can be : Input and output device capability, system representations etc.
- Process requirements may also specify programming language or development method.
- Non functional requirements are more critical than functional requirements. If the non functional requirements do not meet then the complete system is of no use.

2.9.2.1 Types of Non Functional Requirements

- The classification of non functional requirements is as given in Fig. 2.9.1.

Product requirements

- These requirements specify how a delivered product should behave in a particular way. For instance: execution speed, reliability.

Organizational requirements

- The requirements which are consequences of organizational policies and procedures come under this category. For instance : Process standards used implementation requirements.

External requirements

- These requirements arise due to the factors that are external to the system and its development process. For instance : Interoperability requirements, legislative requirements.

Performance requirements

- These requirements specify the performance or durability of its functioning. For instance : Response to various events at particular instance.
- In short, non functional requirements arise through
  i) User needs
  ii) Because of budget constraints
  iii) Organizational policies
  iv) The need for interoperability with other software or hardware systems

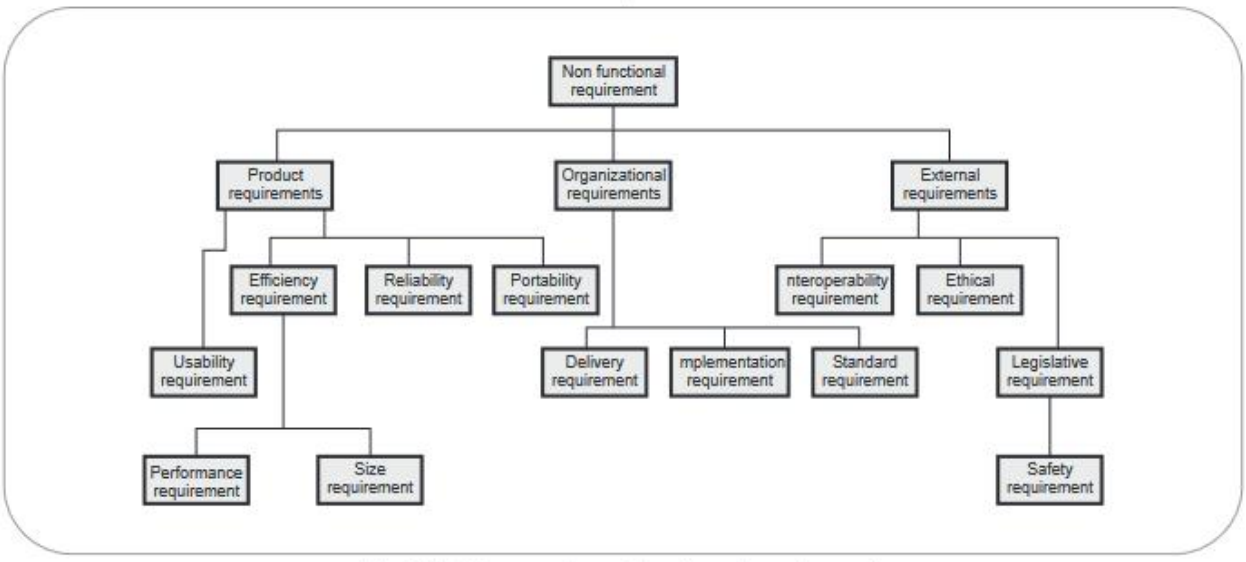


Fig. 2.9.1 Types of non functional requirement

v) Because of external factors such as safety regulations.

- Metrics used for specifying the non functional requirements

Property	Metric
Speed	Events per response time processed transactions per second.
Size	Kilobytes.
Reliability	Mean time to failure. Rate of failure. Occurrence availability.
Robustness	Time to restart after failure. Probability of events causing failure.
Portability	Number of target statements.

Ex. 2.9.1 : *Enlist various functional and non functional requirements for the Bank ATM system.*

Sol. : Functional requirements

1. There should be the facility for the customer to insert a card.
2. The system should first validate card and PIN.
3. The system should allow the customer to deposit amount in the bank.
4. The system should dispense the cash on withdrawal.
5. The system should provide the printout for the transaction.
6. The system should make the record of the transactions made by particular customer.
7. On invalid PIN entry for three times the card should be retained by the system.
8. The cash withdrawal is allowed in multiple of 100.
9. The cash deposition is allowed in multiple of 100.
10. The customer is allowed to transfer amount between the two accounts.
11. The customer is allowed to know the balance enquiry.
12. The customer is allowed to get the printout for desired transaction.
13. The system should be efficient.

Non functional requirements

1. Each of the transaction should be made within 60 seconds. If the time limit is exceeded, then cancel the transaction automatically.
2. If there is no response from the bank computer after request is made within the minutes then the card is rejected with error message.
3. The bank dispenses money only after the processing of withdrawal from the bank. That means if sufficient fund is available in user's account then only the withdrawal request is processed.
4. Each bank should process the transactions from several ATM centers at the same time.
5. The machine should be loaded with sufficient fund in it.

2.9.2.2 Domain Requirements

- Domain requirements are derived from the application domain of the system instead of specific user needs.
- These requirements make use of domain terminologies specific to the existing domain concept.
- The domain requirements may be in the form of new functional requirements, constraints on existing functional requirement or guidance on how to carry out certain computation.
- These are the specialised requirements and hence software engineers find it difficult to co-relate the domain requirements with the system requirements.
- It is important to specify the domain requirements otherwise the system will not work properly.
- **Example :** Domain requirements for the library system.
- There should be user interface for handling the databases. These interfaces should be according to some international standard.
- If there is copyright restriction on some document then it should get printed locally on the server. The copies of such document should not get created.

2.9.3 Difference between Functional and Non Functional Requirements

Functional requirements	Non functional requirements
The functional requirements specify the features of the software system.	The non functional requirements specify the properties of the software system.
Functional requirements describe what the product must do.	Non functional requirements describe how the product should perform.
The functional requirements specify the actions with which the work is concerned.	The non functional requirements specify the experience of the user while using the system.
Example : For a library management system, allowing user to read the article online is a functional requirement.	Example : For a library management system, for a user who wishes to read the article online must be authenticated first.

2.10 Eliciting Requirements

MSBTE : Summer-16, Marks 4

- Questioning is useful only at inception of the project but for detailed requirement elicitation it is not sufficient. During requirement elicitation certain activities such as problem solving, elaboration, negotiation and specification must be carried out. Various ways by which the requirement elicitation can be done are -
  1. Collaborative requirement gathering
  2. Quality function deployment
  3. Use scenarios
  4. Elicitation work product

  Let us discuss these aspects of requirement elicitation in detail -

2.10.1 Collaborative Requirements Gathering

- Collaborative requirement gathering is done using collaborative, team-oriented approach.
- **F**acility **A**pplication **S**pecification **T**echnique (FAST) is an approach in which **joint team of customers and developers** work together to identify the problem, propose elements of solution, negotiate different approaches and prepare a specification for preliminary set of solution requirements.

Guideline for FAST approach -

1. A meeting should be conducted and attended by both software engineers and customers. The place of meeting should be a neutral site.
2. Rules for preparation and participation must be prepared.
3. An agenda should be prepared in such a way that it covers all the important point as well as it allows all the new innovative ideas.
4. A facilitator controls the meeting. He could be customer, developer or outsider.
5. A definition mechanism is used. The mechanism can be work sheets, flip charts, wall stickers, electronic bulletin board, chart room, virtual forum.
6. The goal is to identify the problem, decide the elements of solution, negotiate different approaches and specify the preliminary set of solution requirements.

- In FAST meeting each FAST attendee is asked to prepare - a list of objects, list of services and a list of constraints.
- The list of objects consists of all the objects used in the system, the objects that are produced by the system and the objects that surround the system.
- The list of services contain all the required functionalities that manipulate or interact with the objects.
- The list of constraints consists of all the constraints of the system such as cost, rules, memory requirement, speed accuracy etc.
- As the FAST meeting begins, the very first issue of discussion is the need and justification for the new product. Once everyone agrees upon the fact that the product is justified, each participant has to present his lists.
- These lists are then discussed, manipulated and these modified or refined lists are combined by a group.
- The combined list eliminates redundant entries adds new ideas that come up during the discussion. The combined list is refined in such a way that it helps in building the system.

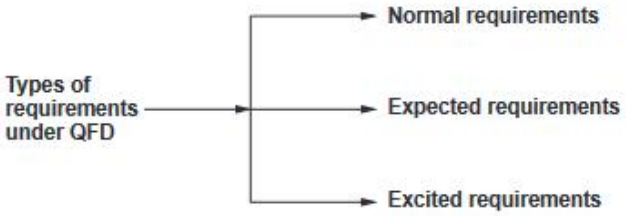
- The combined list should be prepared in such a way that a "consensus lists" can be prepared, for object, services and constraints.
- A team is divided into subteams. Each subteam develops a minispecification from each consensus list.
- Finally a complete draft specification is developed.

For example -

- A FAST team is working on a commerical product. A following product description is given as below -
  - "Nowadays the market for video game is growing rapidly. We would like to enter this market with more features, like attractive GUI, multiple sound setting, realistic (3D) animations. This product is tentatively called 'Gamefun'. At the end of game, scores of each player should be displayed".
  - The FAST attendee prepare following lists -
    1) List of objects - Display, menu, a sound, an event (moving from one level to another) and so on.
    2) List of services - Setting sounds, setting colors in GUI, HELP, instructions for players, score card etc.
    3) List of constraints - Must be user friendly, must have high speed, must accommodate less size, should have less cost.
- The minispecification for Menu (object) can be as given below -
  - Contains 'Start game' and 'exit' options.
  - List of all functional keys with corresponding functionality.
  - Software provides interaction guidance, quick tour, sound controls.
  - All players will play or interact through keys.
  - Software provides facility for change in the look of GUI.
  - Software displays scores of each player.

2.10.2 Quality Function Deployment

- Quality function deployment is a quality management technique which translates the customer needs and wants into technical requirements. This technique was introduced in Japan.
- Under quality function deployment three types of requirements can be defined -



Normal requirements

- The requirements as per goals and objectives of the system are called normal requirements. These requirements can be easily identified during the meeting with the customer.
- **For example :** Handling mouse and keyboard events for any GUI based system.

Expected requirements

- These types of requirements are such requirements which system must be having even if customer did not mention about them. These are such requirements that if they are not present then the system will be meaningless.
- **For example :** A software package for presentation (like Microsoft Power Point) must have option of 'new slide insert', so that user will be able to insert a new slide at any position during his presentation.

Exciting requirements :

- When certain requirements are satisfied by the software beyond customer's expectations then such requirements are called exciting requirements.
- **For example :** Spell check facility in Microsoft Word is an exciting requirement. Various types of deployments that can be conducted during software development process are -
- **Function deployment :** For determining the value of each function this deployment can be done.
- **Information deployment :** After identifying various functionalities events and data objects must be identified.
- **Task deployment :** The task associated with each function must be identified.

- **Value analysis** : Identify the priorities of requirements. The technique of QFD requires proper interaction with customer.

2.10.3 Usage Scenarios

- During requirement gathering overall vision for systems functions and features get developed.
- In order to understand how these functions and features are used by different classes of end users, developers and users create a set of scenarios. This set identifies the usefulness of the system to be constructed. This set is normally called as **use-cases.**
- The use-cases provide a description of how the system will be used.

2.10.4 Elicitation Work Product

Following are some work products that get produced during requirement elicitation -

- A statement of **feasibility study** performed in order to find the need of the project.
- Statement for the scope of the system.
- A list of various stakeholders such as customer, end-users, technical persons, and many others who participate during requirement elicitation.
- A technical description of system environment
- A list of requirements and constraints.
- A set of usage scenarios along with operating conditions.
- The prototype that may get developed for defining the requirements in better manner.

These work products are then reviewed by all the people who participate in requirement elicitation.

Board Question

1. *What is requirements elicitation ? What are the problems faced in eliciting requirements ?*

MSBTE : Summer-16, Marks 4

2.11 Developing Use Cases

MSBTE : Summer-16, 18, Marks 4

Ex. 2.11.1 : *Draw a use case diagram for a Bank Management System.* **MSBTE : Summer-16, Marks 4**

Sol. :

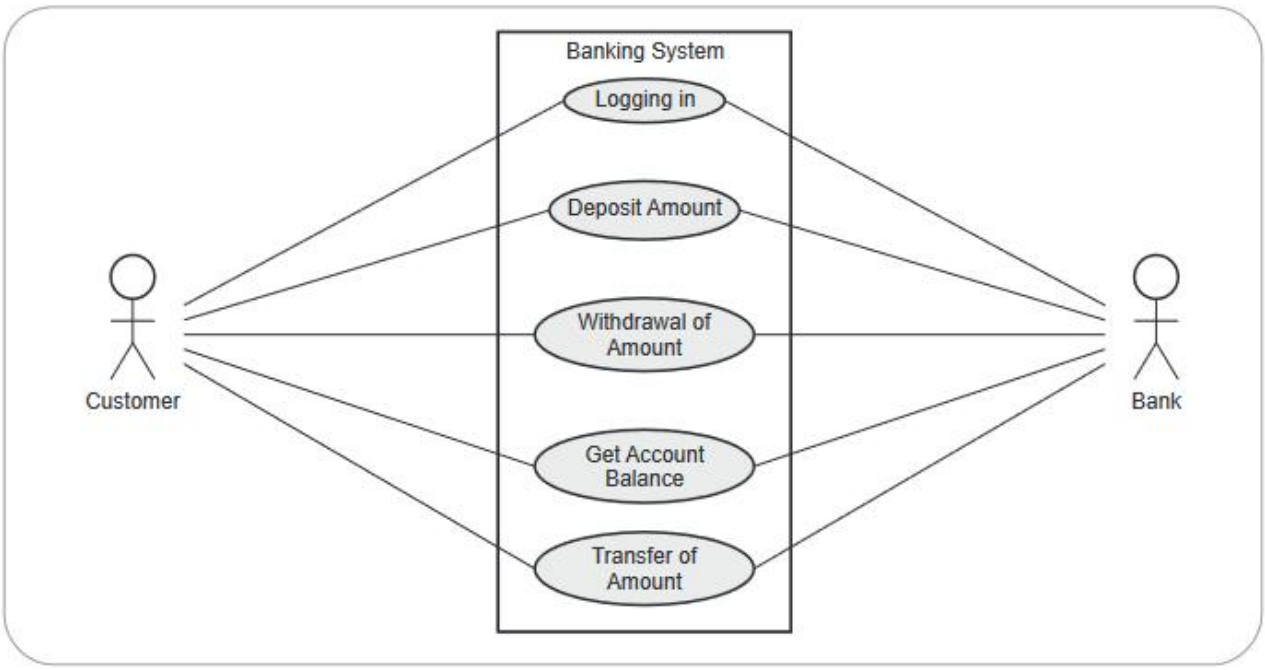


Fig. 2.11.1

Ex. 2.11.2 *Draw a use case diagram for music system.*

MSBTE : Summer-16, Marks 4

Sol. :

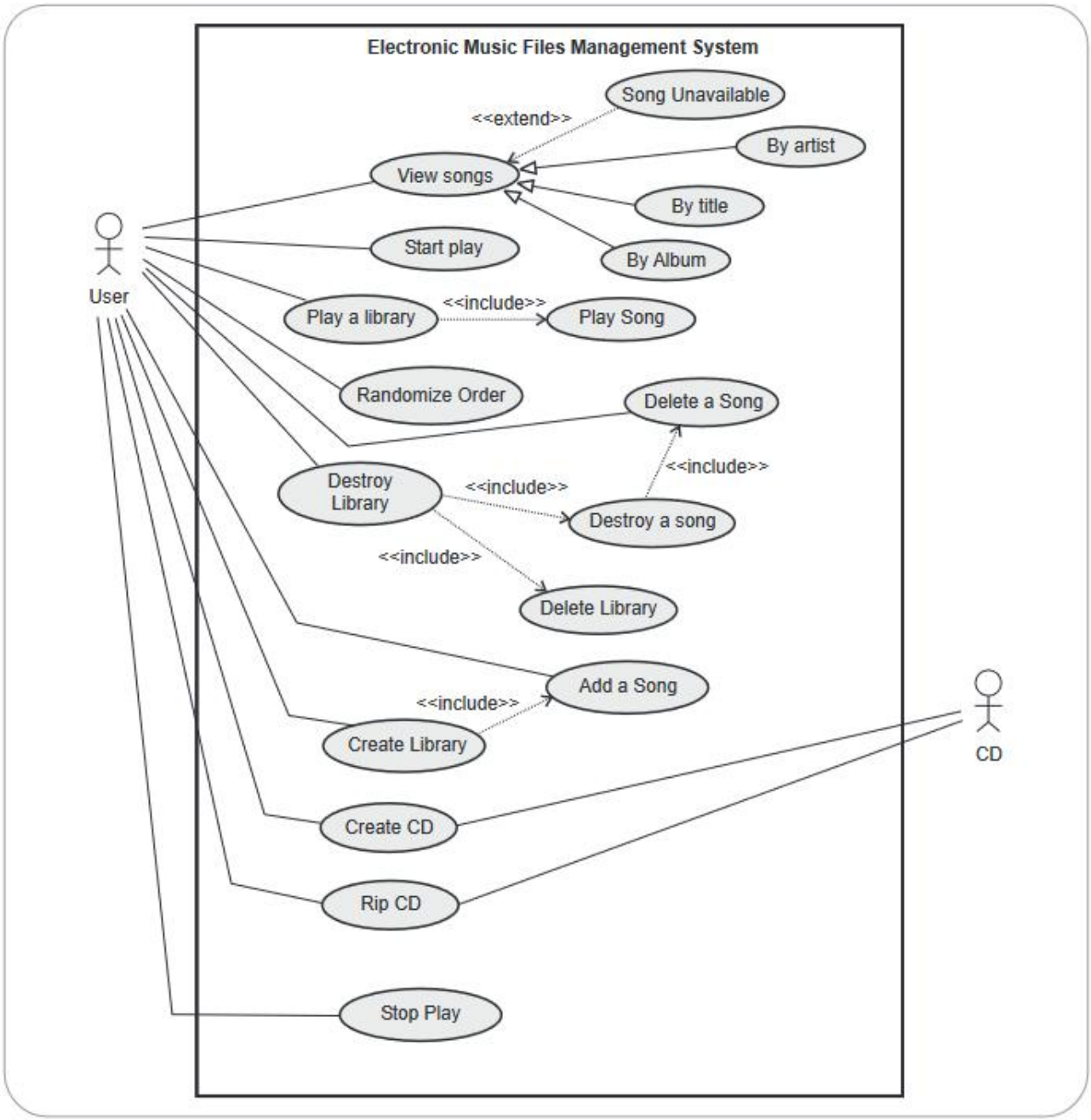


Fig. 2.11.2

Ex. 2.11.3 : *Draw the use case diagram for taking "photocopy of ans books from msbte" website.*

MSBTE : Summer-18, Marks 4

Sol. :

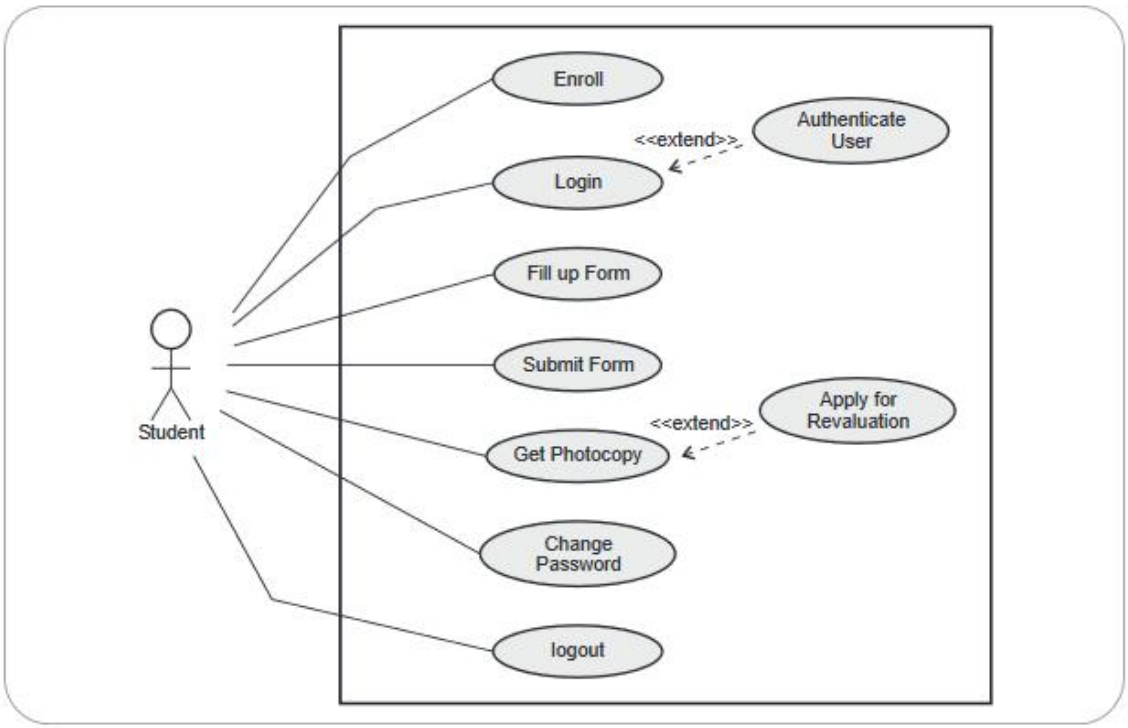


Fig. 2.11.3

Ex. 2.11.4 : *Draw use case diagram for railway ticket reservation system.*

Sol. :

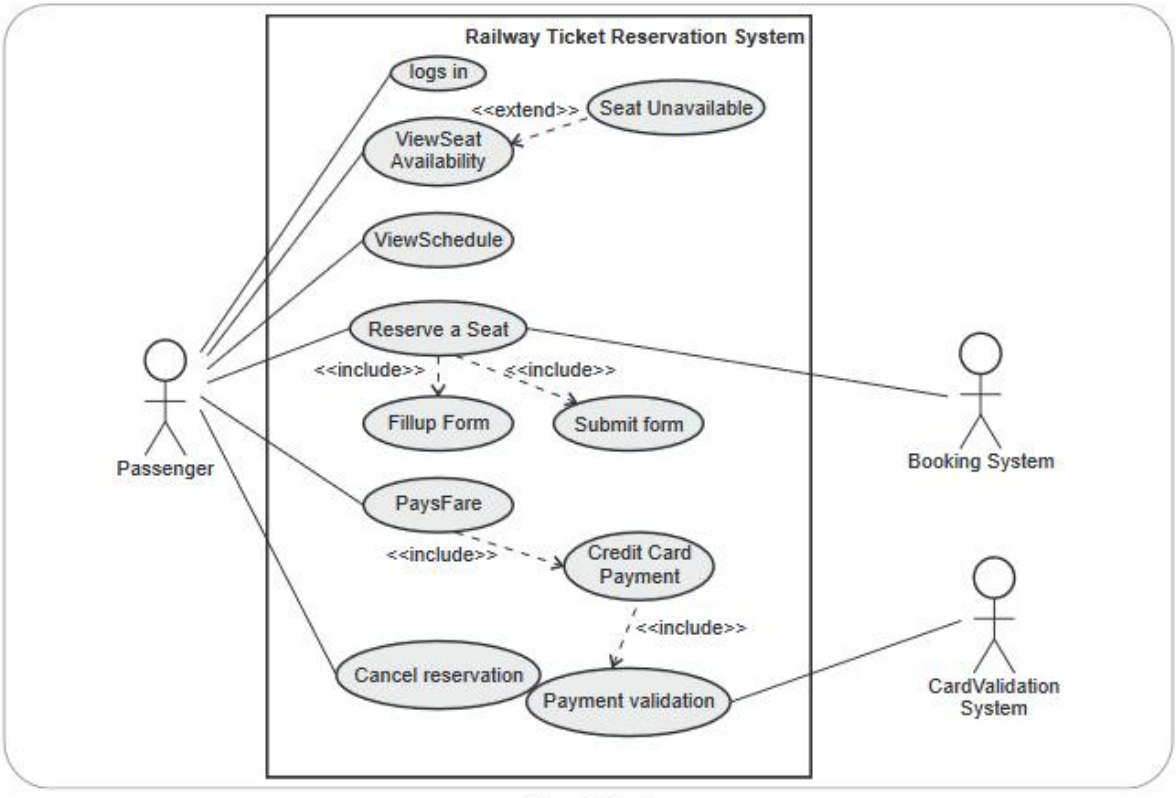


Fig. 2.11.4

Ex. 2.11.5 : *Consider an automated soda machine that gives cool drinks. Draw use case model of the soda machine.*

Sol. :

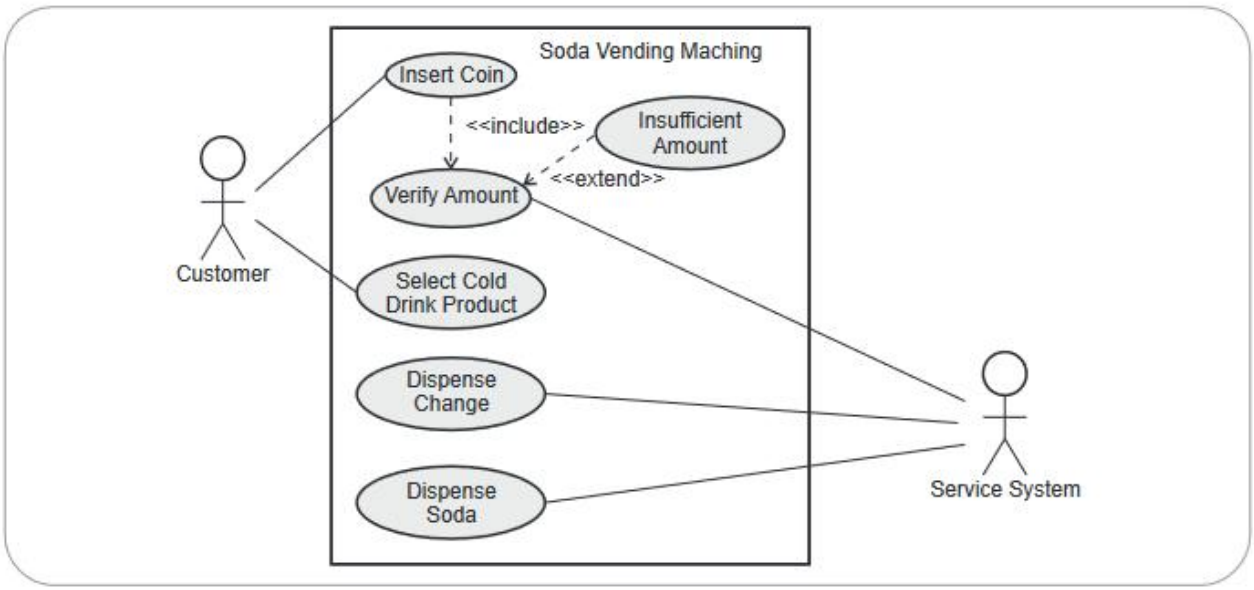


Fig. 2.11.5

2.12 Building Requirement Models

- **Requirement analysis** is an intermediate phase between **system engineering** and **software design**.
- Requirement analysis produces a software specification.

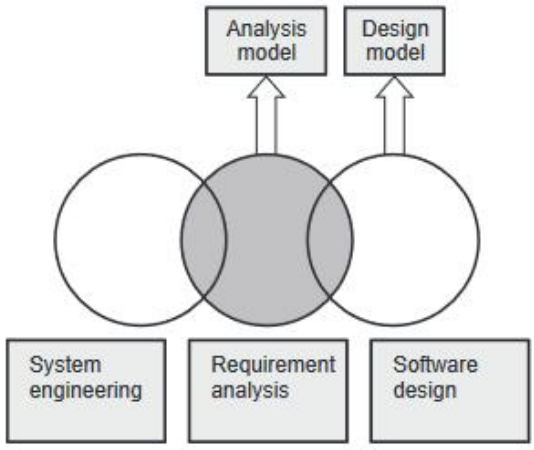


Fig. 2.12.1 Requirement analysis : An intermediate step

How is requirement analysis helpful ?

- **Analyst -** The requirement analysis helps the 'analyst' to refine software allocation. Using requirement analysis various models such as **data model**, **functional model** and **behavioral model** can be defined.
- **Designer -** After requirement analysis, the designer can design for data, architectural interface and component level designs.
- **Developer -** Using requirements specification and design the software can be developed.

What are requirement analysis efforts ?

1. Problem recognition

- The requirement analysis is done for understanding the need of the system. The scope of the software in context of a system must be understood.

2. Evaluation and synthesis

- Following are the tasks that must be done in evaluation and synthesis phase.
  - i) Define all externally observable data objects evaluate data flow.
  - ii) Define software functions.
  - iii) Understand the behaviour of the system.
  - iv) Establish system interface characteristics.
  - v) Uncover the design constraints.

3. Modelling

- After evaluation and synthesis, using data, functional and behavioral domains the data model, functional model and behavioral model can be built.

4. Specification

- The requirement specification (SRS) must be built.

5. Review

- The SRS must be reviewed by project manager and must be refined.

2.12.1 Overall Objectives

- Following are **three objectives** of analysis model -
  1. Describe customer requirement.
  2. Create a basis for software design.
  3. Prepare valid requirements list.
- Analysis model bridges the gap between **system description and design model.** The system description describes overall **system functionality** and design model describes the **software architecture**, user interface and component level structure.
- There is no clear division of analysis and design tasks. Some design can be carried out during analysis and some analysis might be conducted during the design of the software.

Part III : Software Requirement Specification

2.13 Definition of Software Requirement Specification (SRS)

- Software Requirements Specification (SRS) is a kind of document which includes both definition and specification of requirements.

2.14 Need for SRS

- Following are some advantages of SRS which clearly specify the need for SRS.
  1. SRS establishes an **agreement** between the client and supplier about the nature of software product.
  2. SRS can be used for **validating** the final product because the software requirements are specified in SRS. And whether the final product meets these requirements or not - this can be tested using SRS.
  3. For producing **high quality software** high quality SRS must be produced. Hence it is said that high quality SRS is a **prerequisite** for high quality software.
  4. High quality SRS **reduces the cost** of software development.

2.15 Format

- Tracking the team's progress throughout the development activity.
- Typically software designers use **IEEE STD 830-1998** as the basis for the entire Software Specifications. The standard template for writing SRS is as given below.

Document Title
Author(s)
Affiliation
Address
Date
Document Version

1. Introduction

1.1 Purpose of this document : Describes the purpose of the document.

1.2 Scope of this document : Describes the scope of this requirements definition effort. This section also details any constraints that were placed upon the requirements elicitation process, such as schedules, costs.

1.3 Overview : Provides a brief overview of the product defined as a result of the requirements elicitation process.

2. General Description

- Describes the general functionality of the product such as similar system information, user characteristics, user objective, general constraints placed on design team.
- Describes the features of the user community, including their expected expertise with software systems and the application domain.

3. Functional Requirements : This section lists the functional requirements in ranked order. A functional requirement describes the possible effects of a software system, in other words, what the system must accomplish. Each functional requirement should be specified in following manner -

1. **Description** : A full description of the requirement.
2. **Criticality** : Describes how essential this requirement is to the overall system.
3. **Technical issues** : Describes any design or implementation issues involved in satisfying this requirement.
4. **Cost and schedule** : Describes the relative or absolute costs of the system.
5. **Risks :** Describes the circumstances under which this requirement might not able to be satisfied.
6. **Dependencies with other requirements** : Describes interactions with other requirements.
7. Any other appropriate.

4. Interface Requirements : This section describes how the software interfaces with other software products or users for input or output. Examples of such interfaces include library routines, token streams, shared memory, data streams and so forth.

4.1 User Interfaces : Describes how this product interfaces with the user.

4.1.1 GUI : Describes the graphical user interface if present. This section should include a set of screen dumps to illustrate user interface features.

4.1.2 CLI : Describes the command-line interface if present. For each command, a description of all arguments and example values and invocations should be provided.

4.1.3 API : Describes the application programming interface, if present.

4.2 Hardware Interfaces : Describes interfaces to hardware devices.

4.3 Communications Interfaces : Describes network interfaces.

4.4 Software Interfaces : Describes any remaining software interfaces not included above.

5. Performance Requirements : Specifies speed and memory requirements.

6. Design Constraints : Specifies any constraints for the design team such as software or hardware limitations.

7. Other Non Functional Attributes : Specifies any other particular non functional attributes required by the system. Such as :

7.1 Security

7.2 Binary Compatibility

7.3 Reliability

7.4 Maintainability

7.5 Portability

7.6 Extensibility

7.7 Reusability

7.8 Application Compatibility

7.9 Resource Utilization

7.10 Serviceability

... others as appropriate.

8. Operational Scenarios : This section should describe a set of scenarios that illustrate, from the user's perspective, what will be experienced when utilizing the system under various situations.

9. Preliminary Schedule : This section provides an initial version of the project plan, including the major tasks to be accomplished, their interdependencies, and their tentative start/stop dates.

10. Preliminary Budget : This section provides an initial budget for the project.

11. Appendices

11.1 Definitions, Acronyms, Abbreviations : Provides definitions terms and acronyms, can be provided.

11.2 References : Provides complete citations to all documents and meetings referenced.

2.16 Characteristics

MSBTE : Summer-15,16,17, Winter-16, 17, Marks 4

- Following are some characteristics of a good SRS -

1. **Correct :** The SRS must be correct. That means all the requirements must be correctly mentioned, or the requirements must be realistic by nature. For instance : While developing a word processing software, if there is a requirement for spell check facility and if software cannot find the spelling errors from the document, then that means requirement is incorrect.
2. **Complete :** To make the SRS complete, it should specify the purpose of SRS.
3. **Unambiguous :** When requirements are understood correctly then only unambiguous SRS can be written. Unambiguous specification means only one interpretation can be made from the specified requirements. If for particular term there are multiple meanings then, those terms should be mentioned in glossary with proper meaning.
4. **Consistent :** If there are not conflicts in the specified requirements then SRS is said to be consistent.
5. **Stability :** In SRS, it is not possible to specify all the requirements. The SRS must contain all the essential requirements. Each requirement must be clear and explicit.
6. **Verifiable :** The SRS should be written in such a manner that the requirements that are specified within it must be satisfied by the software. For instance - "The GUI should look good". This requirement is not verifiable because one cannot specifically define "what is mean by good ?".
7. **Modifiable :** Writing SRS is an **iterative** process. Even after specifying the requirements they can be modified later on if there is a change in user requirements. While modifying the SRS it is important to preserve the completeness and consistency in requirements.
8. **Traceable :** If origin of requirement is properly given or references of the requirements are correctly mentioned then such a requirement is called as traceable requirement.

Board Questions

1. *What is SRS ?* MSBTE : Summer-15,17,Marks 4
2. *What is SRS ? Explain importance of SRS.* MSBTE : Summer-16, Marks 4
3. *Explain general format of Software Requirement Specification (SRS).* MSBTE : Winter-16, 17, Marks 4

❑❑❑

UNIT- III

3 Software Modelling and Design

3.1 Translating Requirement Model into Design Model

MSBTE : Winter-16,17, Summer-15, 16, 17, Marks 8

- Software design is at the core of software engineering and it is applied irrespective of any process model.
- After analysing and modelling the requirements, software design is done which serves as the basis for code generation and testing.
- Software designing is a process of translating analysis model into the design model.
- The analysis model is manifested by scenario based, class based, flow oriented and behavioural elements and feed the design task.
- The classes and relationships defined by CRC index cards and other useful class based elements provide the basis for the **data or class design.**
- The **architectural design** defines the relationship between major structural elements of the software. The architectural styles and design patterns can be used to achieve the requirements defined for the system.
- The **interface design** describes how software communicates with systems. These systems are interacting with each other as well as with the humans who operate them. Thus interface design represents the flow of information and specific type of behaviour. The usage scenarios and behavioural models of analysis modelling provide the information needed by the interface design.
- The component-level design transforms structural elements of software architecture into procedural description of software module. The information used by the component design is obtained from class based model, flow based model and behavioural model.
- Software design is **important** to assess the **quality** of software. Because design is the only way that we can accurately translate the user requirements into the finished software product.

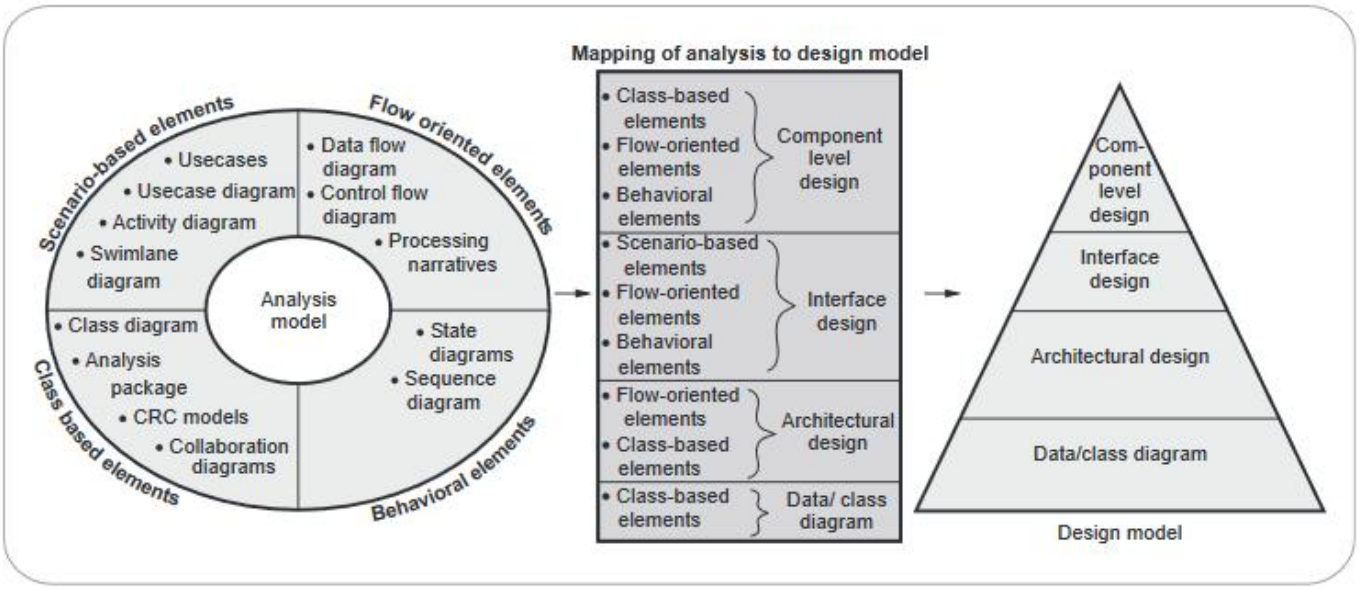


Fig. 3.1.1 Translating analysis model into the design model

- Without design unstable system may get developed. Even if some small changes are made then those changes will go fail. It will become difficult to test the product. The quality of the software product can not be assessed until late in the software process.

3.1.1 Data Modeling

- Data modelling is the basic step in the analysis modelling. In data modelling the data objects are examined independently of processing.
- The data domain is focused. And a model is created at the customer's level of abstraction.
- The data model represents how data objects are related with one another.

3.1.1.1 Data Object, Attributes and Relationships

What is data object ?

- Data object is a set of attributes (data items) that will be manipulated within the software (system).
- Each instance of data object can be identified with the help of unique identifier. **For example :** A student can be identified by using his roll number.
- The system cannot perform without accessing to the instances of object.
- Each data object is described by the attributes which themselves are data items.

Data object is a collection of attributes that act as an aspect, characteristic, quality or descriptor of the object.

Fig. 3.1.2 Object

The vehicle is a data object which can be defined or viewed with the help of set of attributes.

Typical data objects are

- External entities such as printer, user, speakers.
- Things such as reports, displays, signals.
- Occurrences or events such as interrupts, alarm, telephone call.
- Roles such as manager, engineer, customer.
- Organizational units such as division, departments.
- Places such as manufacturing floor, workshops.
- Structures such as student records, accounts, file.

What are attributes ?

Attributes define properties of data object

Typically there are three types of attributes -

1. **Naming attributes** - These attributes are used to name an instance of data object. For example : In a *vehicle* data object *make* and *model* are naming attributes.
2. **Descriptive attributes** - These attributes are used to describe the characteristics or features of the data object. For example : In a *vehicle* data object *color* is a descriptive attribute.
3. **Referential attribute** - These are the attributes that are used in making the reference to another instance in another table. For example : In a *vehicle* data object *owner* is a referential attribute.

What is relationship ?

Relationship represents the connection between the data objects. For example

The relationship between a shopkeeper and a toy is as shown below

Here the toy and shopkeeper are two objects that share following relationships -

- Shopkeeper orders toys.
- Shopkeeper sells toys.
- Shopkeeper shows toys.
- Shopkeeper stocks toys.

3.1.2 Cardinality and Modality

Cardinality in data modeling, cardinality specifies how the number of occurrences of one object is related to the number of occurrences of another object.

- One to one (1 : 1) - One object can relate to only one other object.
- One to many (1 : N) - One object can relate to many objects.
- Many to many (M : N) - Some number of occurrences of an object can relate to some other number of occurrences of another object.

Modality indicates whether or not a particular data object must participate in the relationship.

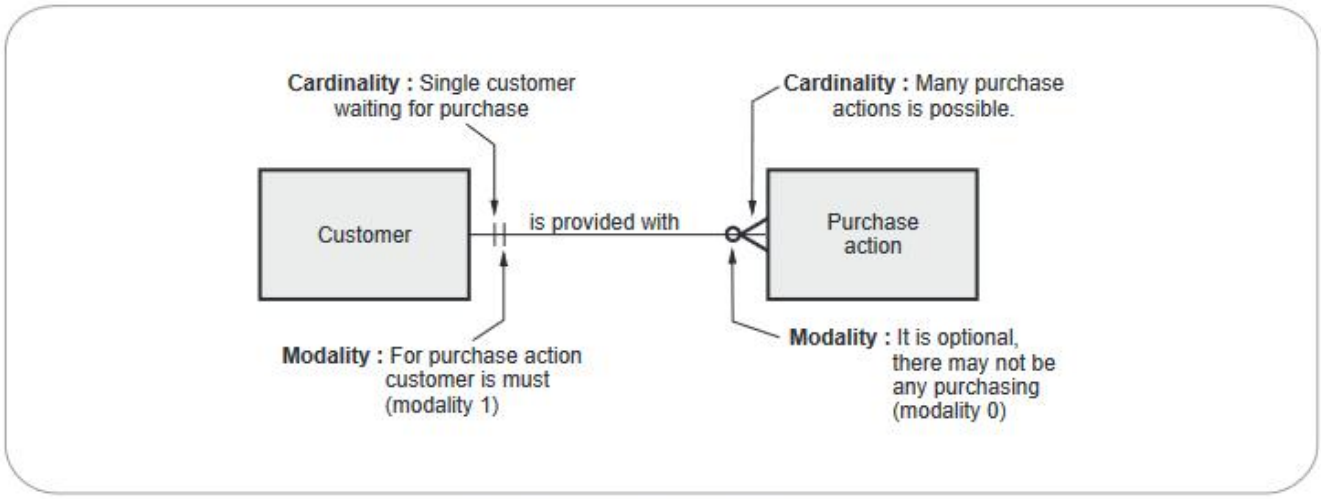


Fig. 3.1.3 Cardinality and modality

Modality of a relationship is 0 (zero) if there is no explicit need for the relationship to occur or the relationship is optional. The modality is 1 (one) if an occurrence of the relationship is mandatory.

Example : Refer Fig. 3.1.3.

Board Questions

1. *With neat diagram explain translation of analysis model into design model.* **MSBTE : Summer-15, Marks 8**
2. *Describe data objects and data attributes.* **MSBTE : Summer-15, Marks 4**
3. *Explain cardinality and modality with example.* **MSBTE : Summer-15, Marks 4**
4. *What is data modeling ? Explain the terms cardinality and modality.* **MSBTE : Summer-16, Marks 4**
5. *Compare cardinality and modality.* **MSBTE : Winter-16,Summer-17, Marks 4**
6. *Explain modality with the help of example.* **MSBTE : Winter-17, Marks 4**

3.2 Analysis Modeling

MSBTE : Winter-16,17, Summer-15, 16, 17, 18, Marks 8

Requirement anslysis produces a software specification.

The requirement analysis helps the 'analyst' to refine software allocation. Using requirement analysis various models such as data model, functional model and behavioral model can be defined.

- Following are **three objectives** of analysis model -
  1. Describe customer requirement.
  2. Create a basis for software design.
  3. Prepare valid requirements list.
- Analysis model bridges the gap between **system description and design model.** The system description describes overall **system functionality** and design model describes the **software architecture**, user interface and component level structure.
- There is no clear division of analysis and design tasks. Some design can be carried out during analysis and some analysis might be conducted during the design of the software.

3.2.1 Elements of Analysis Model

Analysis modelling approach	
Structured approach	**Object oriented approach**
The analysis is made on data and processes in which data is transformed as separate entities.	The analysis is made on the classes and interaction among them in order to meet the customer requirements.
Data objects are modelled in such a way that data attributes and their relationship is defined in structured approach	Unified Modelling Language (UML) and unified processes are used in object oriented modelling approach.

But the commonly used analysis model combines features of both these approaches because the best suitable analysis model bridges the software requirements and software design.

Following are the elements of analysis model -

- Scenario based elements
- Flow-oriented elements
- Behavioural elements
- Class based elements

Following Fig. 3.2.1 illustrates the elements of analysis model.

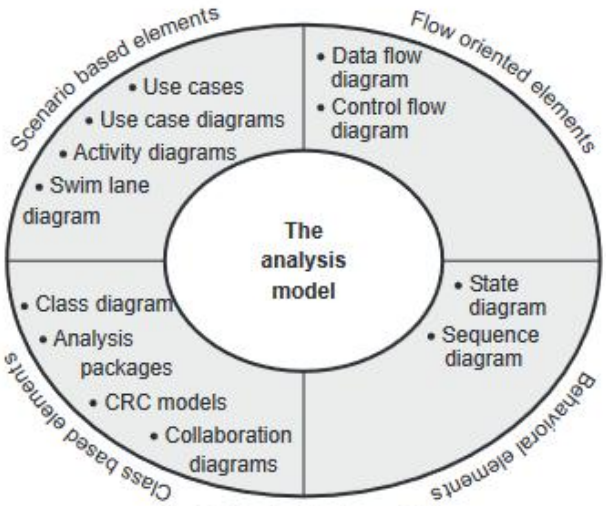


Fig. 3.2.1 Analysis model

Board Questions

1. *Write importance of analysis modeling.* **MSBTE : Summer-15, 17, Marks 4**
2. *With a neat diagram explain analysis model.* **MSBTE : Winter-15, Marks 4**
3. *Explain the various elements of analysis modeling in detail.* **MSBTE : Winter-15, Marks 6**
4. *Describe the terms : Analysis modeling and design modeling.* **MSBTE : Summer-16, Marks 4**
5. *Describe four principles of analysis modeling.* **MSBTE : Winter-16, Summer-18, Marks 4**
6. *Explain analysis modeling.* **MSBTE : Winter-17, Marks 4**
7. *List and explain the elements of analysis model with neat labeled diagram.* **MSBTE : Summer-18, Marks 8**

3.3 Design Modeling

MSBTE : Winter-15, 16,Summer-15, 17, Marks 4

Software design is model of software which translates the requirements into finished software product in which the details about software data structures, architecture, interfaces and components that are necessary to implement the system are given.

Characteristics of Good Design

1. The good design should **implement all the requirements** that are explicitly mentioned in the analysis model. It should accommodate all the implicit requirements demanded by the customer.
2. The design should be **simple enough** so that the code developer, code tester as well as those who are supporting the software will find it readable and understandable.
3. The design should be **comprehensive**. That means it should provide a complete picture of software, addressing the data, functional and behavioural domains from an implementation perspective.

3.3.1 Fundamental Design Concepts

The software design concept provides a framework for implementing the right software.

Various issues that must be considered during software design are -

(1) Abstraction

The abstraction means an ability to cope up with the complexity. Software design occurs at different levels of abstraction. At each stage of software design process levels of abstractions should be applied to refine the software solution. At the higher level of abstraction, the solution should be stated in broad terms and in the lower level more detailed description of the solution is given.

(2) Modularity

- The software is divided into separately named and addressable components that called as **modules.**
- Monolithic software is hard to grasp for the software engineer, hence it has now become a trend to divide the software into number of products. But there is a co-relation between the number of modules and overall cost of the software product.

(3) Architecture

- Architecture means representation of overall structure of an integrated system.

- In architecture various components interact and the data of the structure is used by various components.
- In architectural design various system models can be used and these are -

Model	Functioning
Structural model	Overall architecture of the system can be represented using this model.
Framework model	This model shows the architectural framework and corresponding applicability.
Dynamic model	This model shows the reflection of changes on the system due to external events.
Process model	The sequence of processes and their functioning is represented in this model.
Functional model	The functional hierarchy occurring in the system is represented by this model.

(4) Refinement

- Refinement is actually a process of elaboration.
- Stepwise refinement is a top-down design strategy proposed by Niklaus WIRTH.
- The architecture of a program is developed by successively refining levels of procedural detail.
- The process of program refinement is analogous to the process of refinement and partitioning that is used during requirements analysis.
- Abstraction and refinement are complementary concepts. The major difference is that - In the abstraction low-level details are suppressed. Refinement helps the designer to elaborate low-level details.

(5) Pattern

According to **Brad Appleton** the design pattern can be defined as - It is a named nugget (something valuable) of insight which conveys the essence of proven solution to a recurring problem within a certain context.

In other words, design pattern acts as a design solution for a particular problem occurring in specific domain. Using design pattern designer can determine whether-

- Pattern can be **reusable.**
- Pattern can be used for **current work**.
- Pattern can be used to **solve similar kind of problem** with different functionality.

(6) Information Hiding

Information hiding is one of the important property of effective modular design. The term information hiding means the modules are designed in such a way that information contained in one module cannot be accessible to the other module (the module which does not require this information). Due to information hiding only limited amount of information can be passed to other module or to any local data structure used by other module.

The **advantage** of information hiding is basically in testing and maintenance. Due to information hiding some data and procedures of one module can be hidden from another module. This ultimately **avoids** introduction of **errors** module from one module to another. Similarly one can make **changes** in the desired module without affecting the other module.

(7) Functional Independence

- The functional independence can be achieved by developing the functional modules with single-minded approach.
- By using functional independence functions may be compartmentalized and interfaces are simplified.
- Independent modules are easier to maintain with reduced error propagation.
- The major benefit of functional independence is in achieving effective modularity.
- The functional independence is assessed using two qualitative criteria - **Cohesion** and **coupling**.
- **Cohesion :** A cohesive module performs only "one task" in software procedure with little interaction with other modules. In other words cohesive module performs only one thing.
- **Coupling :** Coupling effectively represents how the modules can be "connected" with other module or with the outside world.Coupling is a measure of interconnection among modules in a program structure.

(8) Refactoring

- It is defined refactoring as "The process of changing a software system in such a way that the external behaviour of the design do not get changed, however the internal structure gets improved".

- Benefits of refactoring are -
  1. The redundancy can be achieved.
  2. Inefficient algorithms can be eliminated or can be replaced by efficient one.
  3. Poorly constructed or inaccurate data structures can be removed or replaced.
  4. Other design failures can be rectified.

Board Question

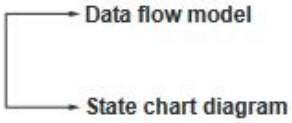
1. *What are the characteristics of good design ?* **MSBTE : Summer-15,17,Marks 4**
2. *With reference to software design give the meanings of*
   i) Modularity ii) Functional independence iii) Refactoring iv) Information hiding **MSBTE : Winter-15, Marks 4**
3. *Explain following with reference to design concepts in design modeling.*
   i) Abstraction ii) Functional independence **MSBTE : Winter-16, Marks 4**

3.4 Design Notations

MSBTE : Winter-17, Summer-15, 17, Marks 4

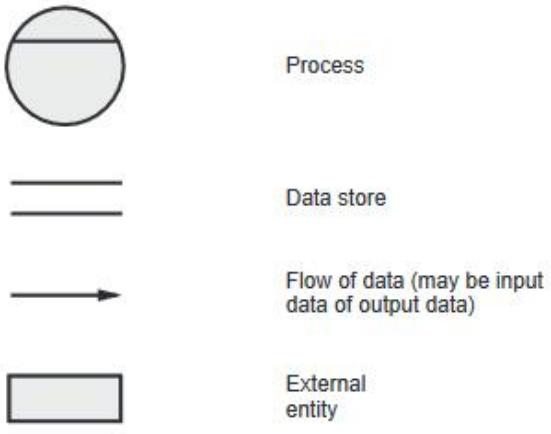
3.4.1 Data Flow Diagram (DFD)

- Behavioral models are used to describe the overall behavior of a system. There are two types of models that depict the behavior of the system
- The data flow model represents the flow of data and state chart diagram represent the states that are occurring in the system.
- These models can be used separately on together depending upon nature of application.
- Let us discuss these models in detail –



3.4.1.1 Data Flow Diagram

- The data flow diagrams depict the information flow and the transforms that are applied on the data as it moves from input to output.
- The symbols that are used in data flow diagrams are -
- The data flow diagrams are used to represent the system at any level of abstraction.
- The DFD can be partitioned into levels that represent increase in information flow and detailed functionality.
- A level 0 DFD is called as 'fundamental system model' or 'context model'. In the context model the entire software system is represented by a bubble with input and output indicated by incoming and outgoing arrows.
- Each process shown in level 1 represents the sub functions of overall system.
- The number of levels in DFD can be increased until every process represents the basic functionality.
- As the number of levels gets increased in the DFD, each bubble gets refined. The following figure shows the leveling in DFD. Note that the information flow continuity must be maintained.



Designing Data Flow Diagrams

- The data flow diagrams are used to model the information and function domain. Refinement of DFD into greater levels helps the analyst to perform functional decomposition.
- The guideline for creating a DFD is as given in Fig. 3.4.1.

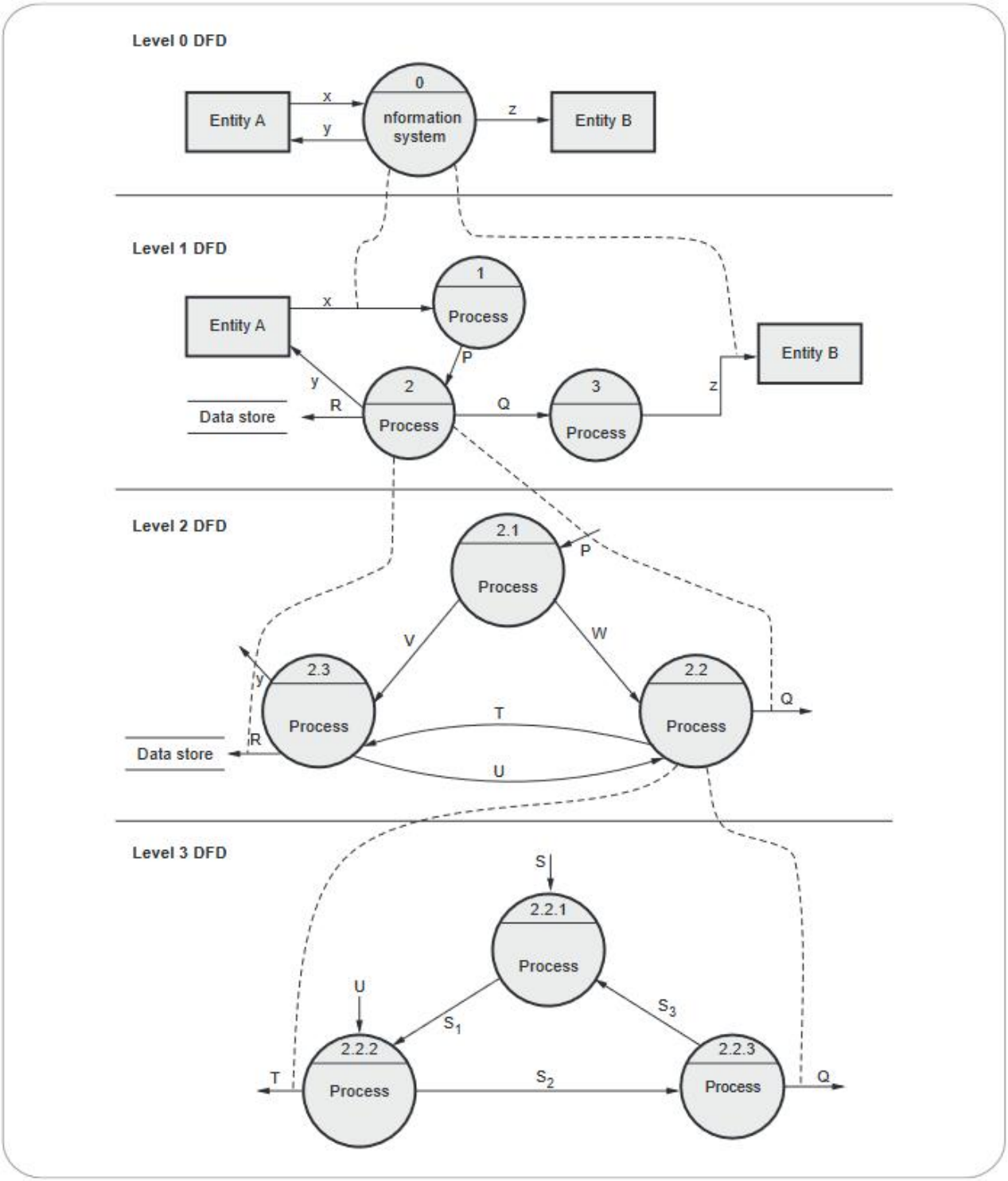


Fig. 3.4.1 Levels of DFD

1. Level 0 DFD i.e. Context level DFD should depict the system as a single bubble.
2. Primary input and primary output should be carefully identified.
3. While doing the refinement isolate processes, data objects and data stores to represent the next level.
4. All the bubbles (processes) and arrows should be appropriately named.
5. One bubble at a time should be refined.
6. Information flow continuity must be maintained from level to level.

- A simple and effective approach to expand the level 0 DFD to level 1 is to perform "grammatical parse" on the problem description. Identify nouns and verbs from it. Typically nouns represent the external entities, data objects or data stores and verbs represent the processes. Although grammatical parsing is not a foolproof but we can gather much useful information to create the data flow diagrams.

Rules for designing DFD

1. No process can have only outputs or only inputs. The process must have both outputs and inputs.

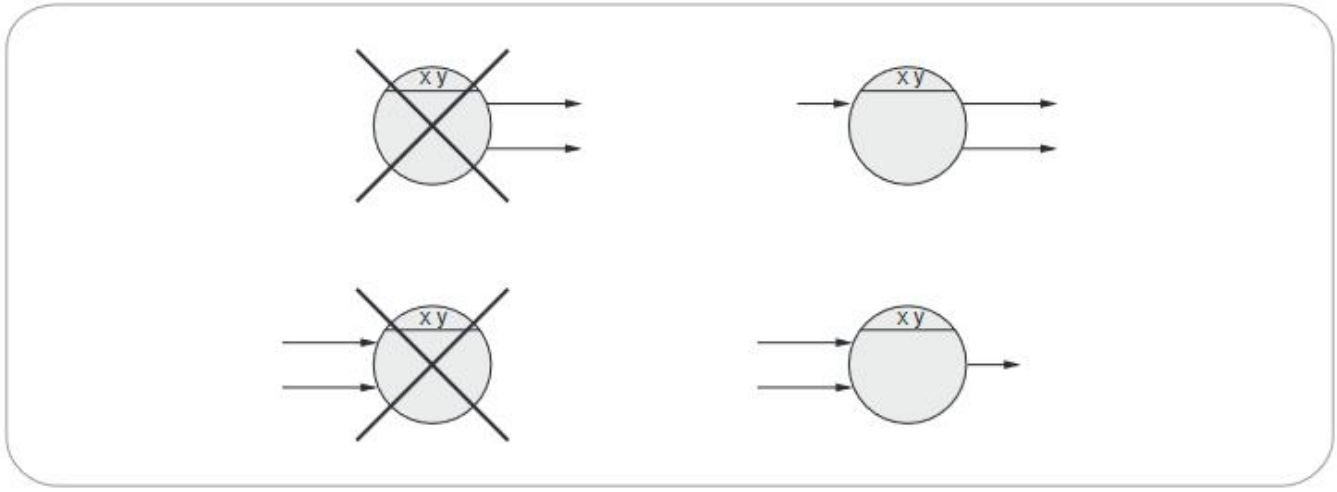


Fig. 3.4.2

2. The verb phrases in the problem description can be identified as processes in the system.

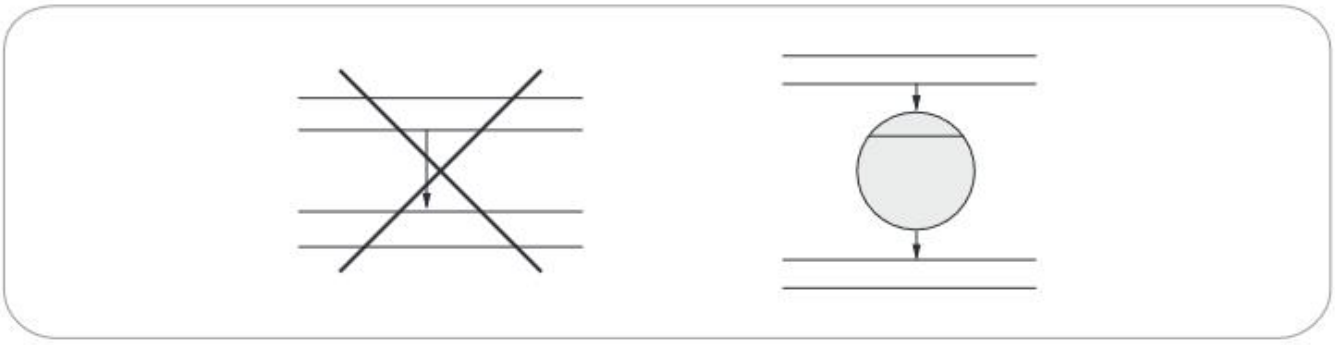


Fig. 3.4.3

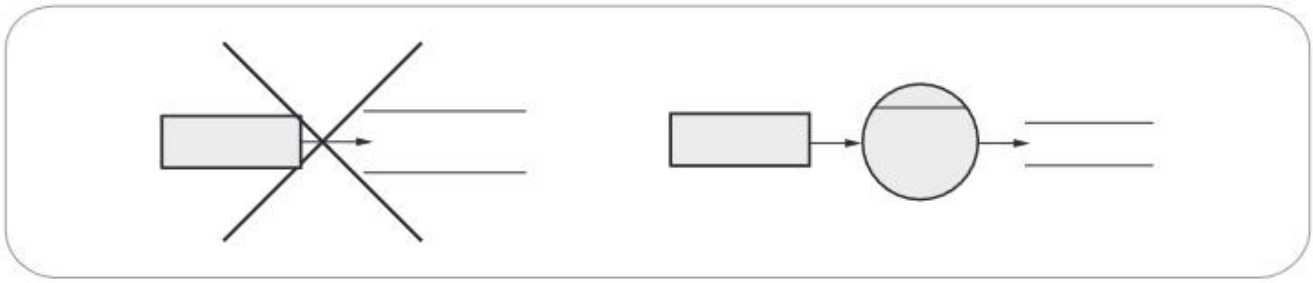


Fig. 3.4.4

3. There should not be a direct flow between data stores and external entity. This flow should go through a process.
4. Data store labels should be noun phrases from problem description.
5. No data should move directly between external entities. The data flow should go through a process.

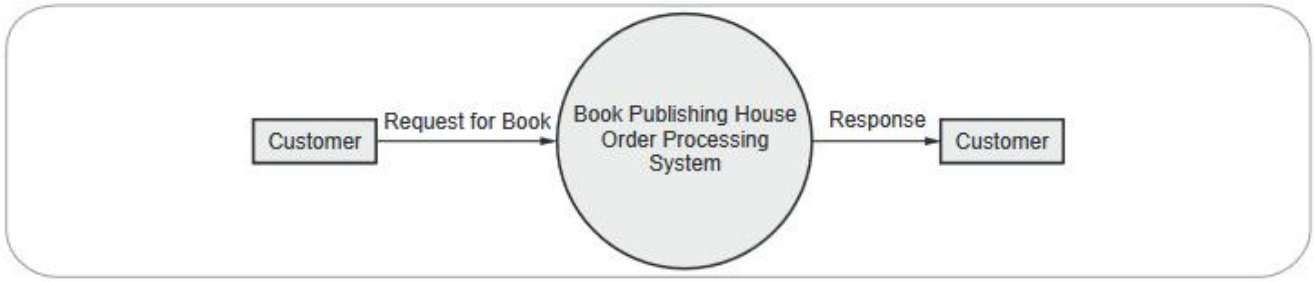
Fig. 3.4.5

6. Generally source and sink labels are noun phrases.
7. Interactions between external entities is outside scope of the system and therefore not represented in DFD.
8. Data flow from process to a data store is for updation/insertion/deletion.
9. Data flow from data store to process is for retrieving or using the information.
10. Data flow labels are noun phrases from problem description.

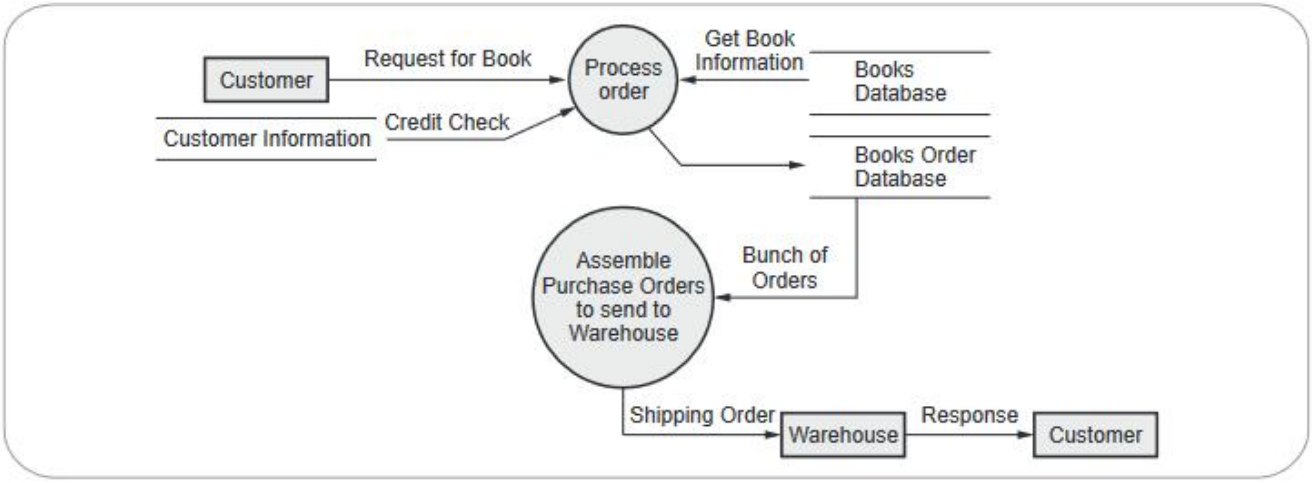
Ex. 3.4.1 : *Draw a dataflow diagram level 0 and level 1 for a Book Publishing House.* **MSBTE : Summer-16, Marks 6**

Sol.:

Level 0 DFD :



Level 1 DFD :



Ex. 3.4.2 *Draw DFD level 0 and level 1 for Hotel management system.* **MSBTE : Winter-16, Marks 6**

Sol. : **Level 0 DFD**

In this level, the system is designed globally with input and output. The input to food ordering system are -

1. As customer orders for the food. Hence food order is an input.

The output to food ordering system are -

1) Receipt.
2) The food order should be further given to kitchen for processing the order.
3) Bill and management report is given to restaurant manager.

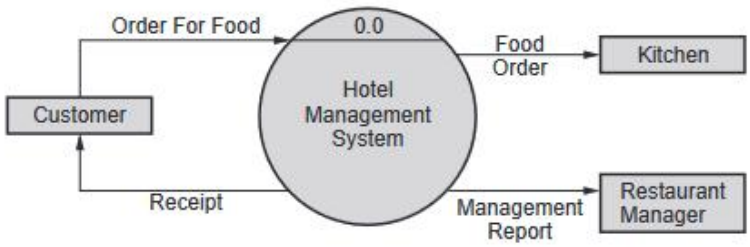


Fig. 3.4.6 Level 0 DFD

Level 1 DFD

In this level, the bubble 0.0 is shown in more detail by various processes. The process 1.0 is for processing an order. And processes 2.0, 3.0 and 4.0 are for housekeeping activities involved in food ordering system. To create a management report there should be some information of daily sold items. At the same time inventory data has to be maintained for keep track of 'instock' items. Hence we have used two data stores in this DFD -

1. Database of sold items 2. Inventory database.

Finally management report can be prepared using daily sold details and daily inventory deplition amount. This management report is given to restaurant manager.

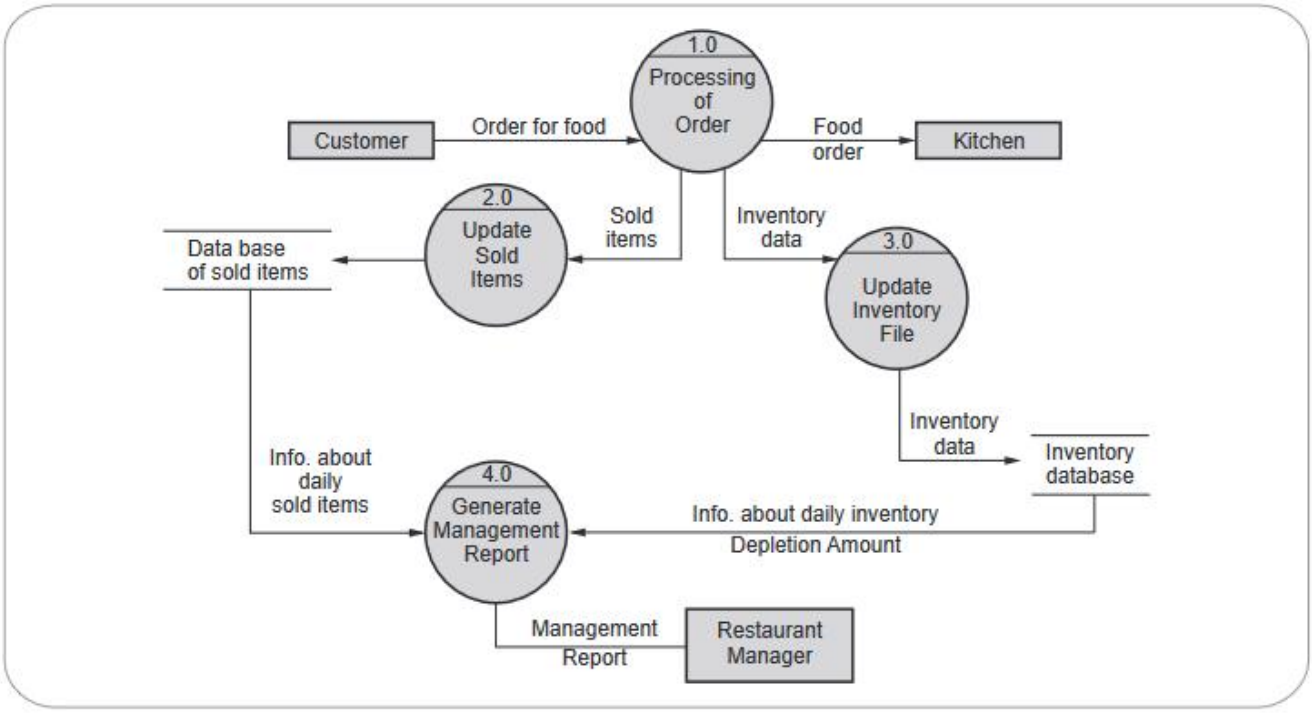


Fig. 3.4.7 Level 1 DFD

Ex. 3.4.3 *For library management system draw level 0 and level 1 DFD.* **MSBTE : Winter-15, Summer-17, Marks 4**

Sol. : In this DFD the whole system is represented with the help of input, processing and output. The input can be -

i) Student requests for a book hence **Book request**.

ii) To show identity of the student he/she has to submit his/her Library card, hence **Library card**.

The processing unit can be globally given as shown in Fig. 3.4.8.

Library information system

The system will produce following outputs -

i) The demanded book will be given to student. Hence **Book** will be the output.

ii) The library information system should display **demanded book information** which can be used by customer while selecting the book.

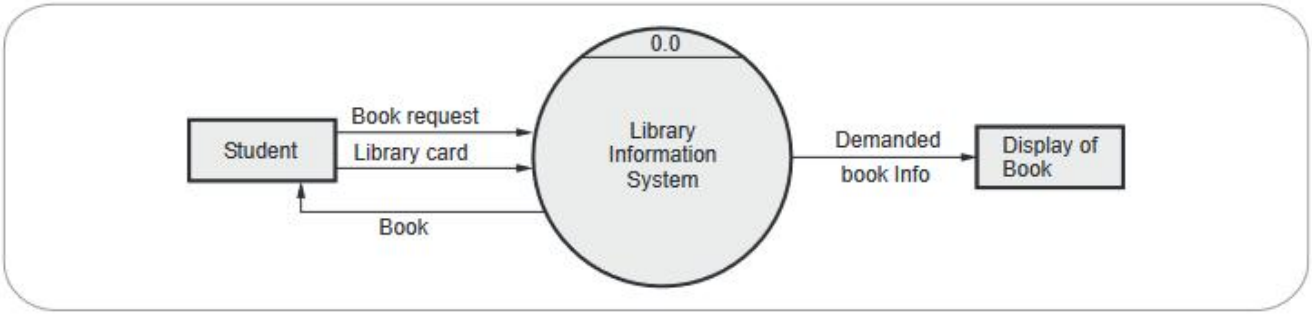


Fig. 3.4.8 Level 0 DFD (Context level DFD)

Level 1 DFD

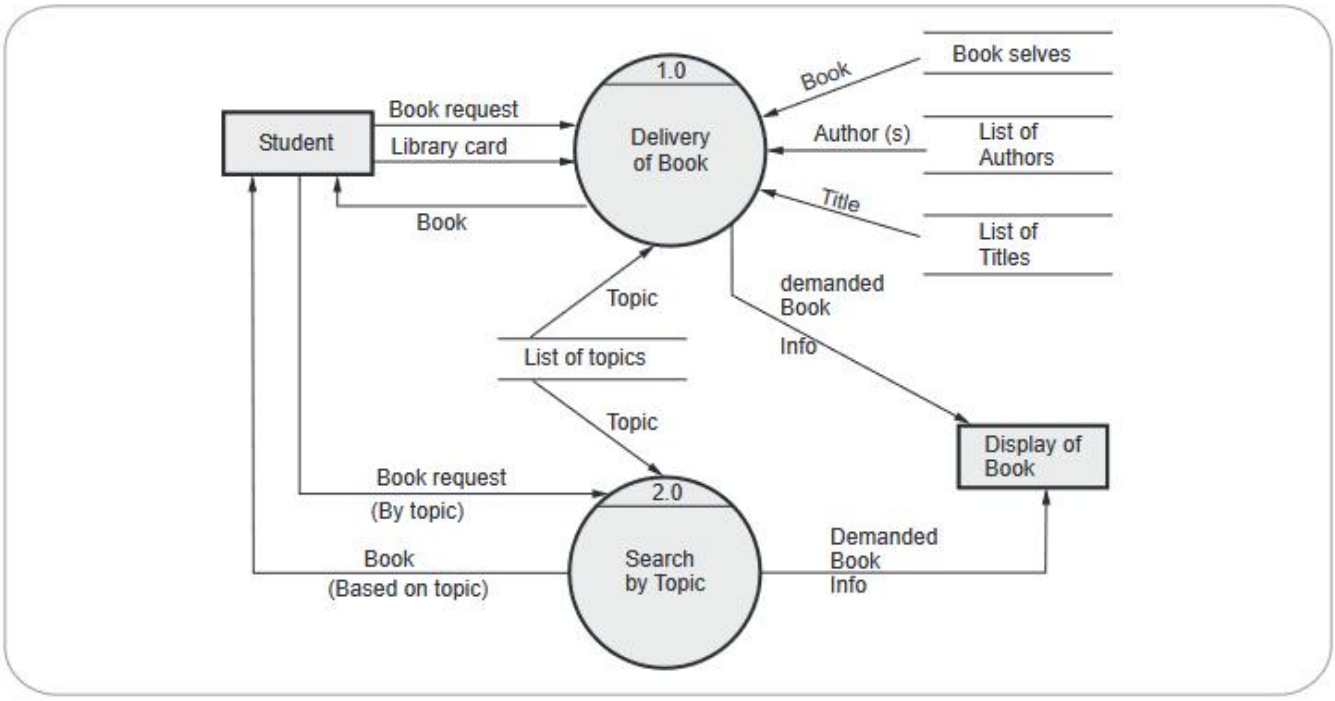


Fig. 3.4.9 Level 1 DFD

In this level, the system is exposed with more processing details. The processes that need to carried out are -

i) Delivery of Book.

ii) Search by Topic.

These processes require certain information such as List of Authors, List of Titles, List of Topics, the book selves from which books can be located. This kind of information is represented by **data store.**

Ex. 3.4.4 *Draw and explain level 0 and level 1 Dataflow diagram for "Online examination Win17 of form filling on MSBTE website".* **MSBTE : Summer-18, Marks 6**

Sol. :

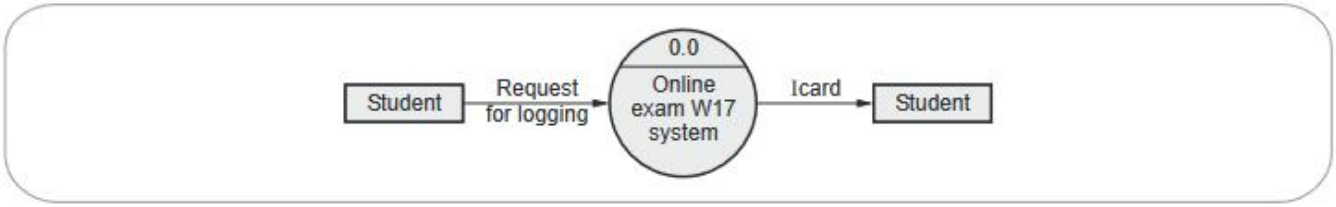


Fig. 3.4.10 Level 0 DFD

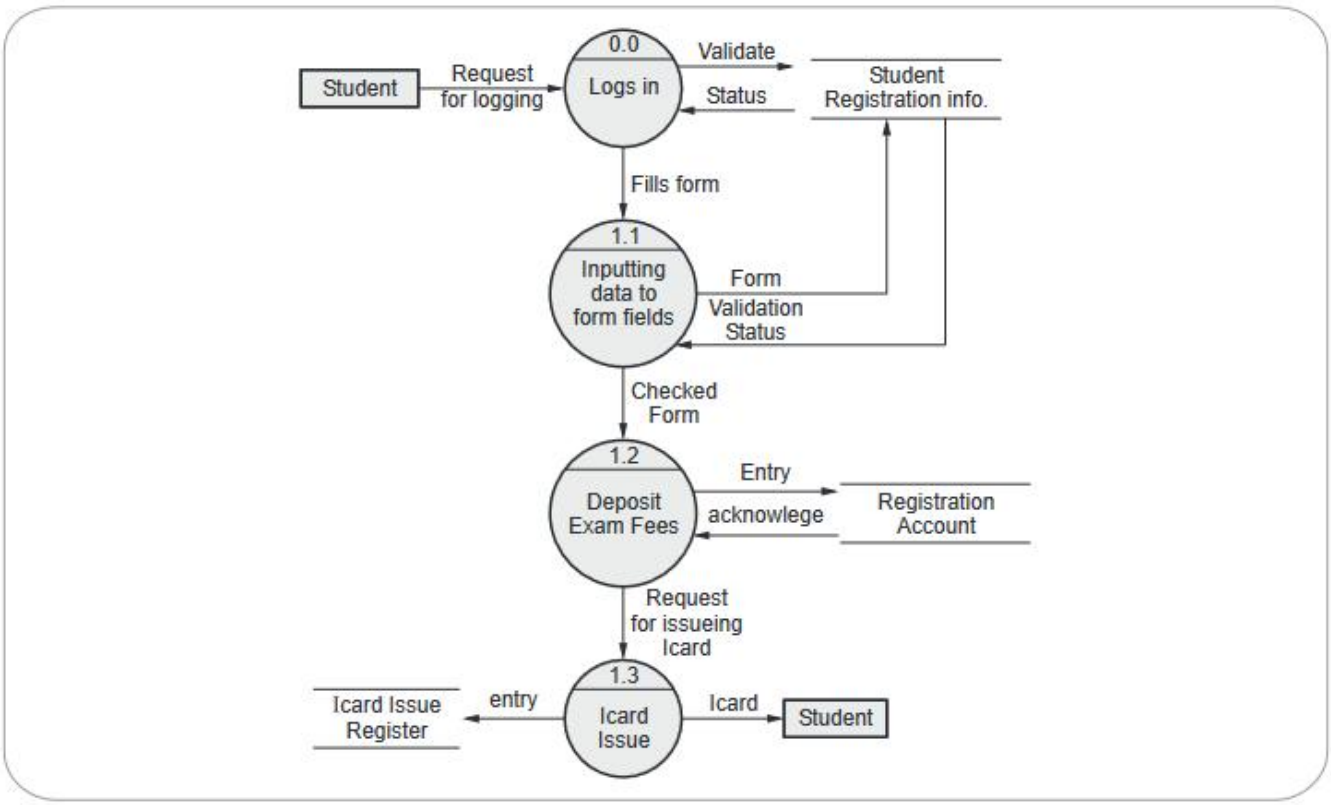


Fig. 3.4.11 Level 1 DFD

Ex. 3.4.5 : *Draw Level 0 and Level 1 DFD for railway reservation system*

Sol. : **Problem description :** For an online railway reservation system, passenger can make online booking for the seats, by specifying the requirements. These requirements are - type of reservation A/C coach, non A/C coach, two tier or three tier number of seats, name of the train, start and destination of journey. The system then checks for availability of such requirements. If such a reservation is possible then system generates **availability** of reservation. The passengers **checks** for availability and the **charges are then calculated** for the selected requirement, and these are acknowledged to the passenger. If the passenger is satisfied then he **confirms** the reservation.

The required DFD can be drawn in levels as follows -

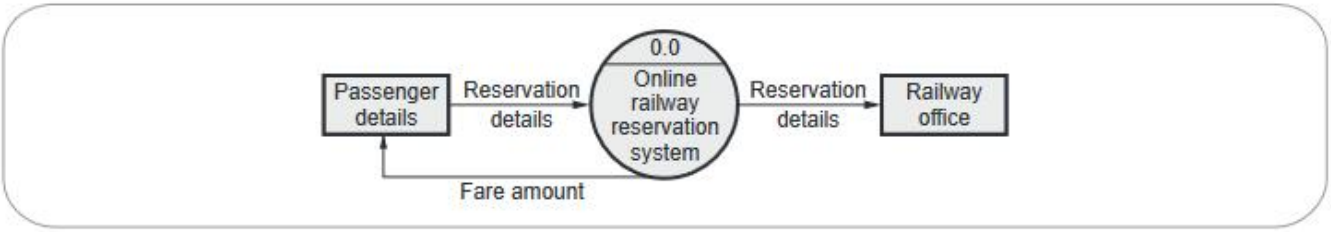


Fig. 3.4.12 Level 0 DFD

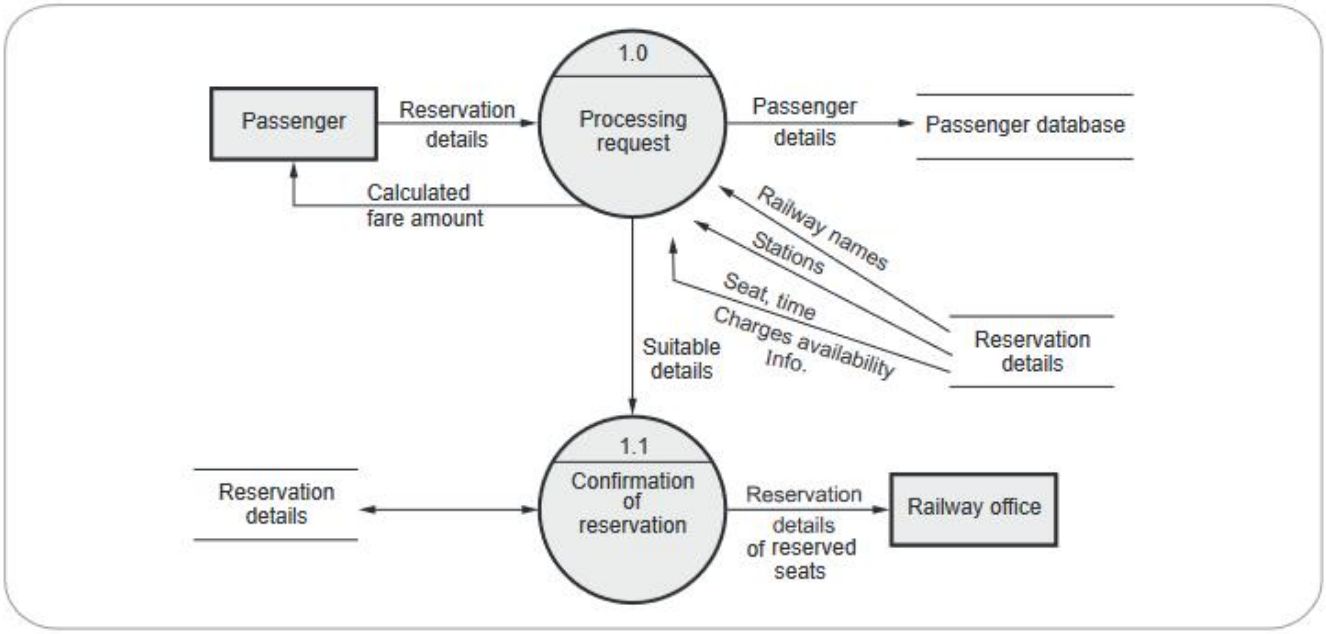


Fig. 3.4.13 Level 1 DFD

Ex. 3.4.6 : *Draw Level 0 and Level 1 DFD for ATM system.*

Sol.: The level 0 DFD is the context level DFD in which only inputs and outputs that are interacting with the system are given.

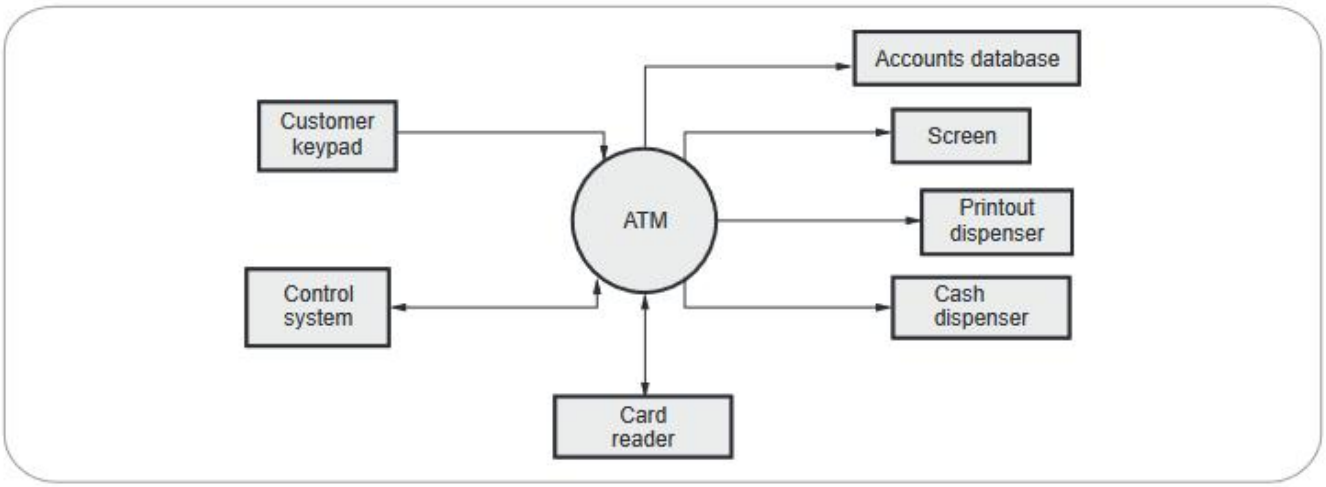


Fig. 3.4.14 Level 0 DFD

More detailing is done in level 1.

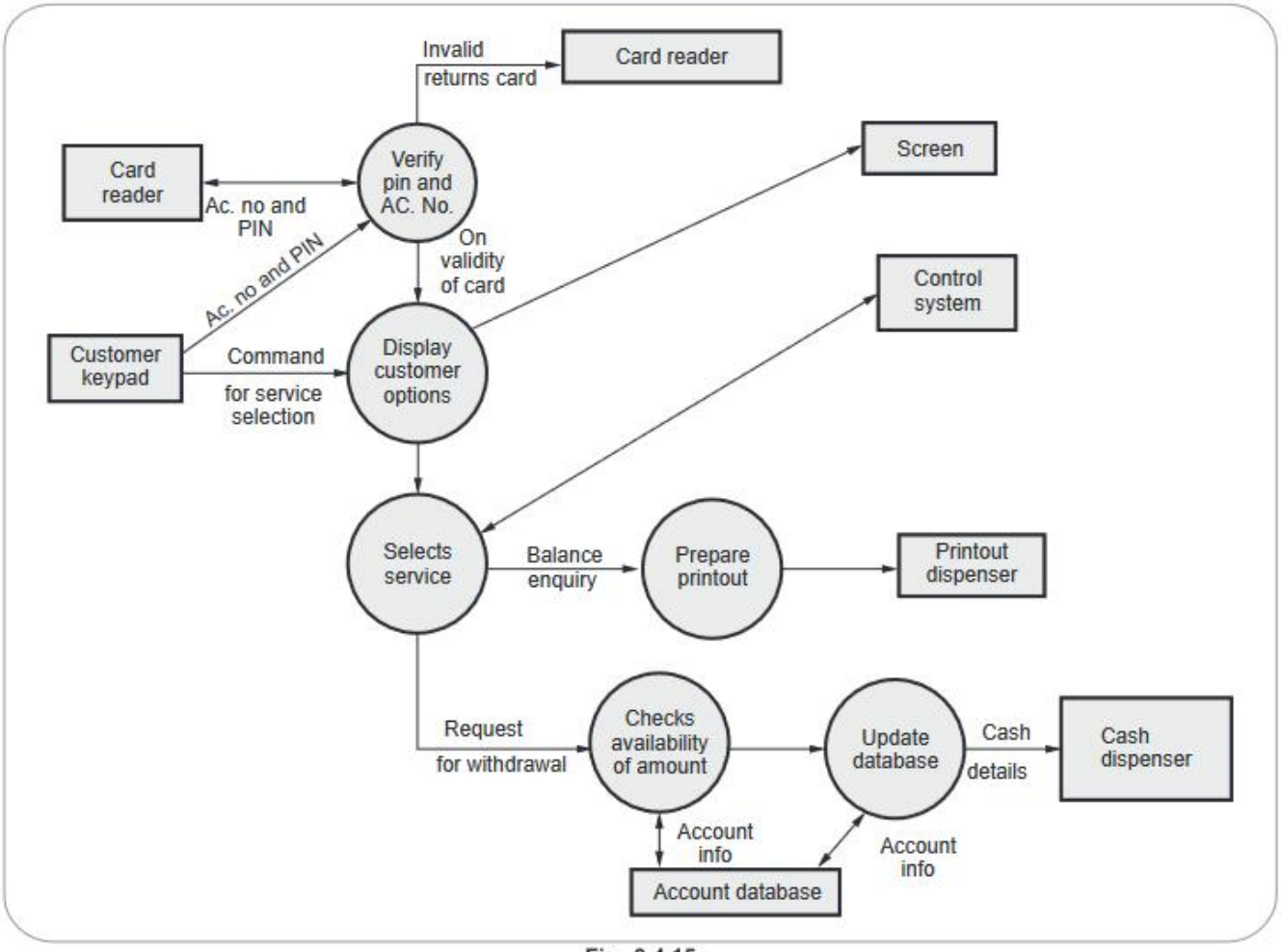


Fig. 3.4.15

3.4.2 Structured Flow Chart

- The structure chart is a principle tool of structured design.
- The basic element in the structured chart is module. Module is defined as a collection of program statement with four attributes.
- **Input and output :** What the module gets from the invoker is called input and what the receiver gets from the module is called output.
- **Function :** The function processes the input and produces the output.
- **Mechanics :** The code or the logic by which the function is carried out.
- **Internal data :** It is the own workspace.
- The two modules can be connected to each other by a connector as shown below.

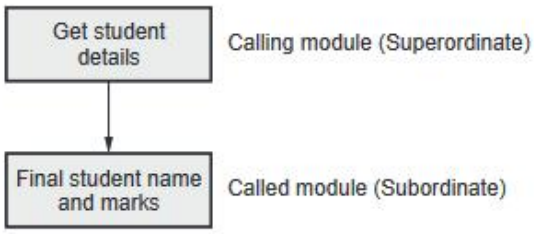


Fig. 3.4.16 Connector

- The module uses data and flags. The data is processed by different modules. The flag is used as a control signal. It can be set or reset.
- For example if we have two modules one for getting the employee detail(subordinate) and another module is for finding the employee name(superordinate). Then the caller module will sent the data as employee's ID and using that ID the called module will find the name of employee. If the employee ID is valid then that message will be given by the called module to a caller module. The use of data and flag can be as shown below.

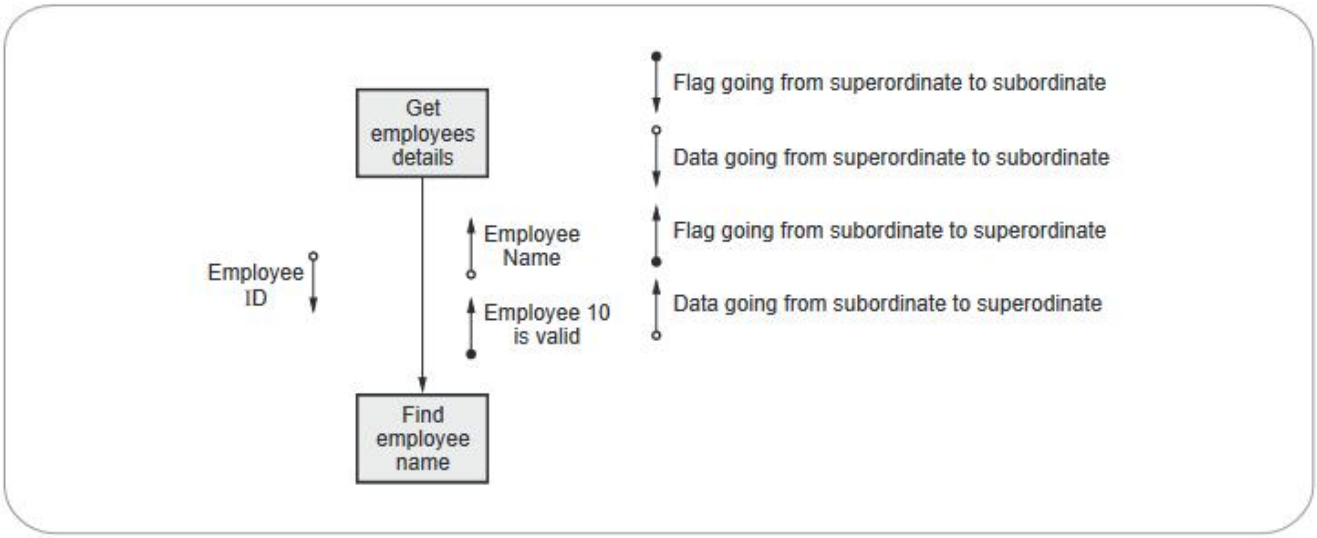


Fig. 3.4.17 Use of flag

- The iterations and decisions on a structured chart is as shown below.

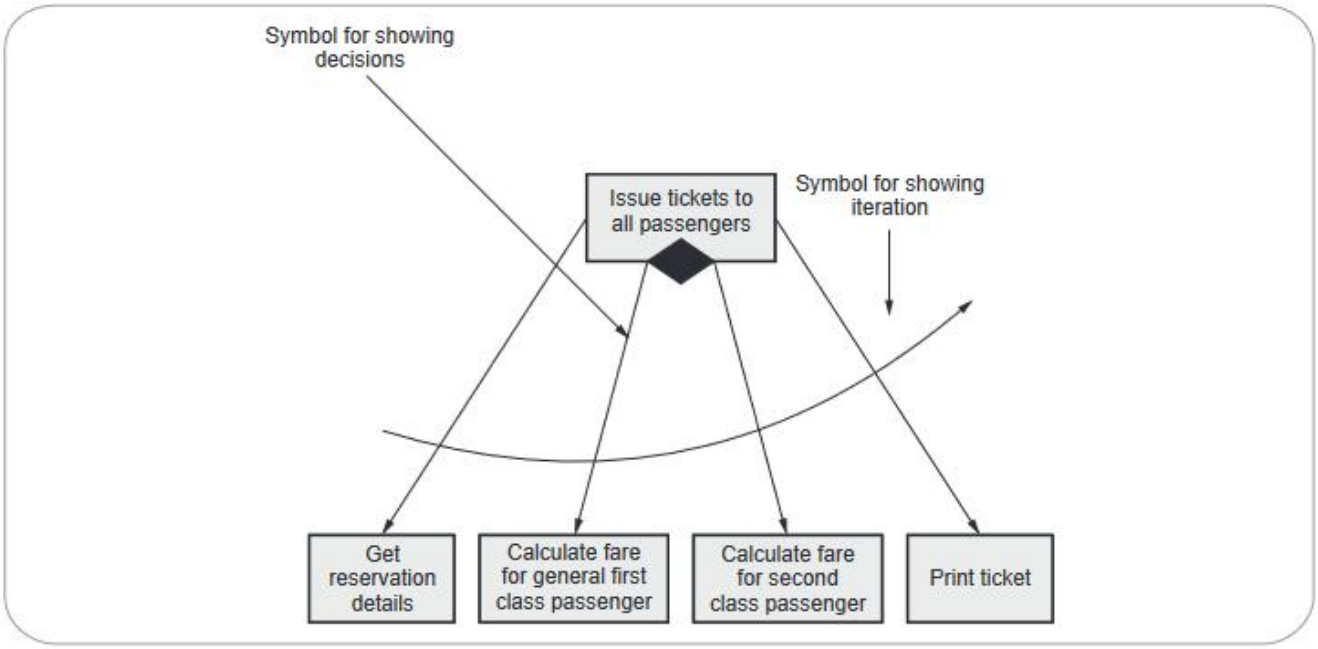


Fig. 3.4.18 Interaction

• The example of a structure chart is as given below.

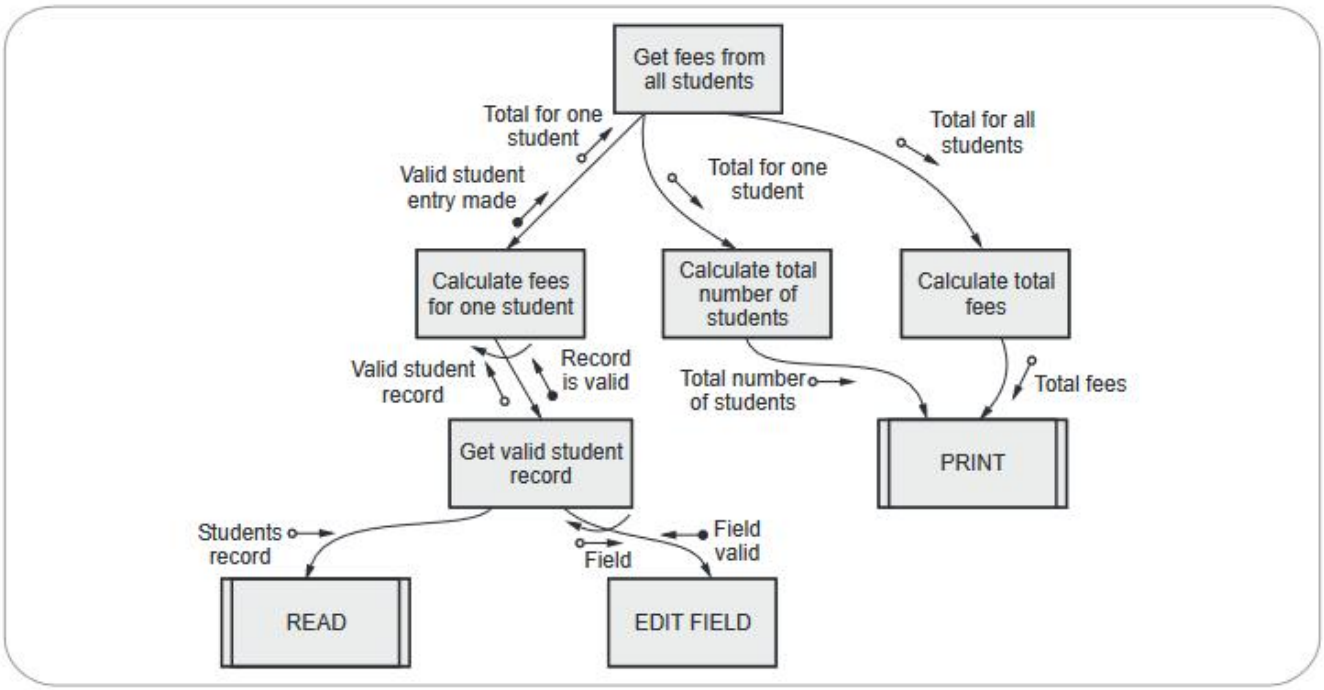


Fig. 3.4.19 Structure chart

3.4.3 Decision Tables

Many software applications possess certain modules in which certain complex conditions occur and such conditions should be responded by taking appropriate actions. Hence decision tables are used in which complex conditions and corresponding actions are stored in a tabular form. Such decision table can be used by software program to model the complicated logic. And therefore it is almost impossible to mis-interpret the decision table.

The decision table is divided into four sections as given below.

Rules

Conditions	Condition alternatives
Actions	Action entries

The decision tables can associate many independent conditions with several actions in an elegant way. The rules can be shown by numbering them.

For example : we can create a decision table for following scenario

" A nationalised bank gives housing loan to his customers with fixed and floating rate of interset. If the customer chooses the fixed rate of interest and takes a loan of an amount which is less than one lakhs then apply scheme A structure to him and if the customer chooses floating rate for a loan amount less than 1 lakhs then apply scheme B structure to him and at the same time apply monthly

charges to him.If the cutomer chooses floating rate with loan amount > 1 lakhs then apply monthly charges and give him other scheme".

Condition	1	2	3	4	5
Fixed rate	T	F	F	F	
Floating rate	F	T	T	T	
Loan amount < 1 lakh	F	F	T	F	
Action					
Min monthly charge	F	T	T	F	
Apply scheme A	T	F	F	F	
Apply scheme B	F	T	F	F	
Other mode of charge	F	F	T	F	

In above given decision table T indicates condition is true and F indicates condition is false. Thus decision table is useful for representing the conditions and corresponding actions.

Board Questions

1. *What is DFD ? Explain level 1 DFD with example.* **MSBTE : Summer-15, Marks 4**
2. *What is DFD ? Explain its symbol.* **MSBTE : Summer-17, Marks 4**
3. *Explain DFD with example.* **MSBTE : Winter-17, Marks 4**

3.5 Testing

MSBTE : Summer-15, 16, 17, 18, Winter-15, 16, 17, Marks 4

3.5.1 Meaning and Purpose

Definition : Software testing is an activity performed to uncover errors. It is a critical element of software quality assurance and represents the ultimate review of specification, design and coding.

The purpose of software testing is to ensure whether the software functions appear to be working according to specifications and performance requirements.

Goals of Testing

The testing goals are,

1. Testing is a process of executing a program with the intend of finding an error.
2. A good test case is one that has high probability of finding an undiscovered error.
3. A successful test is one that uncovers an as-yet undiscovered error.

The major testing objective is to design tests that systematically uncover types of errors with minimum time and effort.

Testing Principles

Every software engineer must apply following testing principles while performing the software testing.

1. All tests should be traceable to customer requirements.
2. Tests should be planned long before testing begins.
3. The Pareto principle can be applied to software testing - 80 % of all errors uncovered during testing will likely be traceable to 20 % of all program modules.
4. Testing should begin "in the small" and progress toward testing "in the large".
5. Exhaustive testing is not possible.
6. To be most effective, testing should be conducted by an independent third party.

3.5.2 Black Box and White Box Testing

Black Box Testing

- The black box testing is also called as behavioural testing.
- Black box testing methods focus on the functional requirements of the software. Test sets are derived that fully exercise all functional requirements.
- The black box testing is not an alternative to white box testing and it uncovers different class of errors than white box testing.
- **Objective :** Black box testing uncovers following types of errors.
  1. Incorrect or missing functions
  2. Interface errors 3. Errors in data structures

4. Performance errors
5. Initialization or termination errors

White Box Testing

- In white box testing the procedural details are closely examined.
- In this testing the internals of software are tested to make sure that they operate according to specifications and designs.
- Objective : The major focus of white box testing is on internal structures, logic paths, control flows, data flows, internal data structures, conditions, loops, etc.

Difference between black box testing and white box testing

Sr. No.	Black box testing	White box testing
1.	Black box testing is called behavioural testing.	White box testing is called glass box testing.
2.	Black box testing examines some fundamental aspect of the system with little regard for internal logical structure of the software.	In white box testing the procedural details, all the logical paths, all the internal data structures are closely examined.
3.	During black box testing the program cannot be tested 100 percent.	White box testing lead to test the program thoroughly.
4.	This type of testing is suitable for large projects.	This type of testing is suitable for small projects.

Advantages and disadvantages of black box testing

Advantages :

1. The black box testing focuses on fundamental aspect of system without being concerned for internal logical structure of the software.
2. The advantage of conducting black box testing is to uncover following types of errors.
   i. Incorrect or missing functions
   ii. Interface errors
   iii. Errors in external data structures
   iv. Performance errors
   v. Initialization or termination errors

Disadvantages :

1. All the independent paths within a module cannot be tested.
2. Logical decisions along with their true and false sides cannot be tested.
3. All the loops and the boundaries of these loops cannot be exercised with black box testing.
4. Internal data structure cannot be validated.

Advantages and disadvantages of white box testing

Advantages :

1. Each procedure can be tested thoroughly. The internal structures, data flows, logical paths, conditions and loops can be tested in detail.
2. It helps in optimizing the code.
3. White box testing can be easily automated.
4. Due to knowledge of internal coding structure it is easy to find out which type of input data can help in testing the application efficiently.

Disadvantages :

1. The knowledge of internal structure and coding is desired for the tested. Thus the skilled tester is required for whitebox testing. Due to this the testing cost is increased.
2. Sometimes it is difficult to test each and every path of the software and hence many paths may go untested.
3. Maintaining the white box testing is very difficult because it may use specialized tools like code analyzer and debugging tools are required.
4. The missing functionality can not be identified.

3.5.3 Level of Testing

We begin by 'testing-in-the-small' and move toward 'testing-in-the-large'.

Various testing strategies for conventional software are

1. Unit testing
2. Integration testing
3. Validation testing
4. System testing

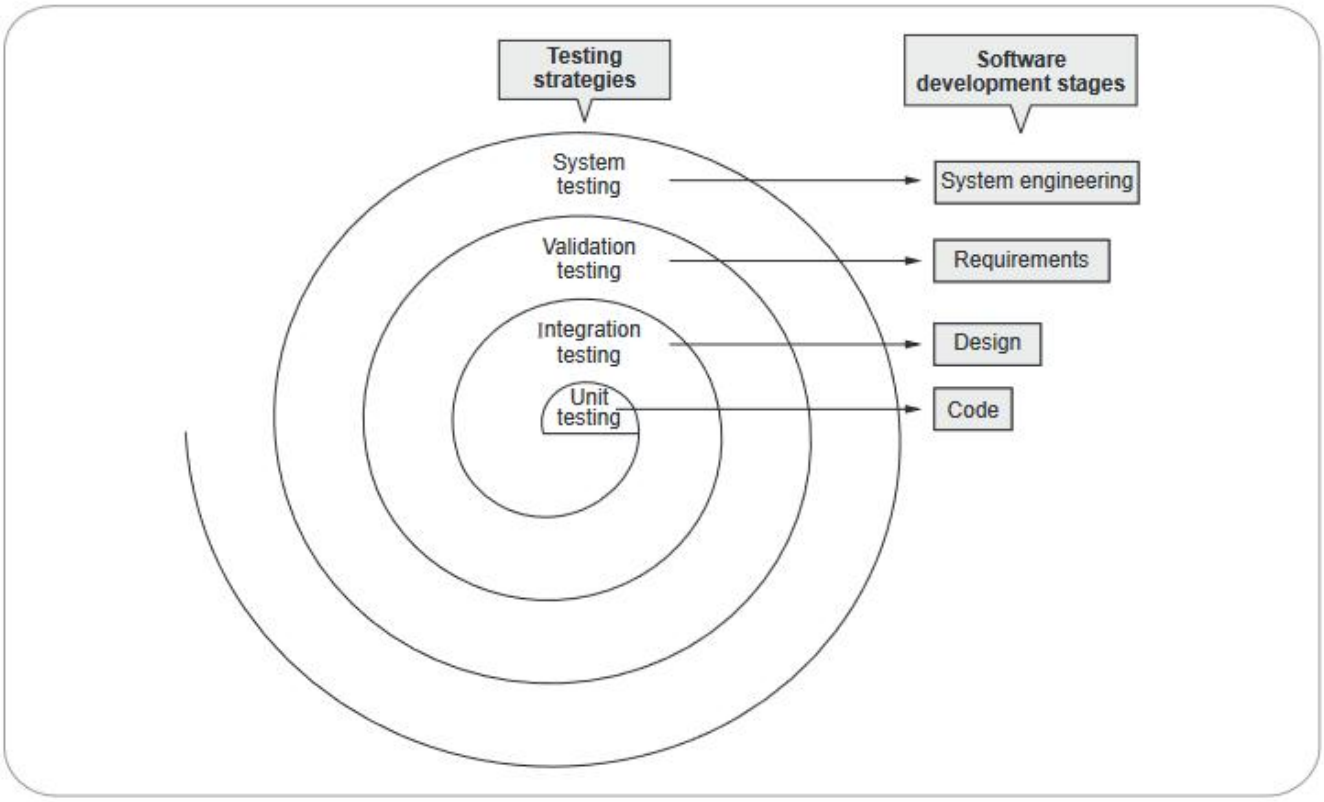


Fig. 3.5.1 Testing strategy

1. **Unit testing -** In this type of testing techniques are applied to detect the errors from each software component individually.
2. **Integration testing -** It focuses on issues associated with verification and program construction as components begin interacting with one another.
3. **Validation testing -** It provides assurance that the software validation criteria (established during requirements analysis) meets all functional, behavioural and performance requirements.
4. **System testing -** In system testing all system elements forming the system is tested as a whole.

3.5.3.1 Unit Testing

- In unit testing the individual components are tested independently to ensure their quality.
- The focus is to uncover the errors in design and implementation.
- The various tests that are conducted during the unit test are described as below.

1. Module interfaces are tested for proper information flow in and out of the program.
2. Local data are examined to ensure that integrity is maintained.
3. Boundary conditions are tested to ensure that the module operates properly at boundaries established to limit or restrict processing.
4. All the basis (independent) paths are tested for ensuring that all statements in the module have been executed only once.
5. All error handling paths should be tested.
6. Drivers and stub software need to be developed to test incomplete software. The "driver" is a program that accepts the test data and prints the relevant results. And the "stub" is a subprogram that uses the module interfaces and performs the

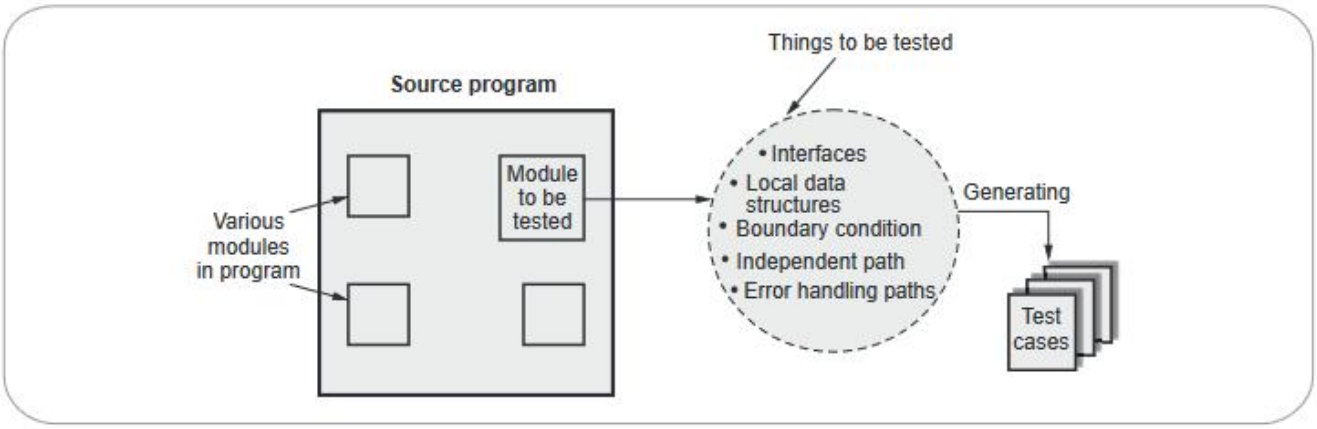


Fig. 3.5.2 Unit testing

minimal data manipulation if required. This is illustrated by following Fig. 3.5.3.

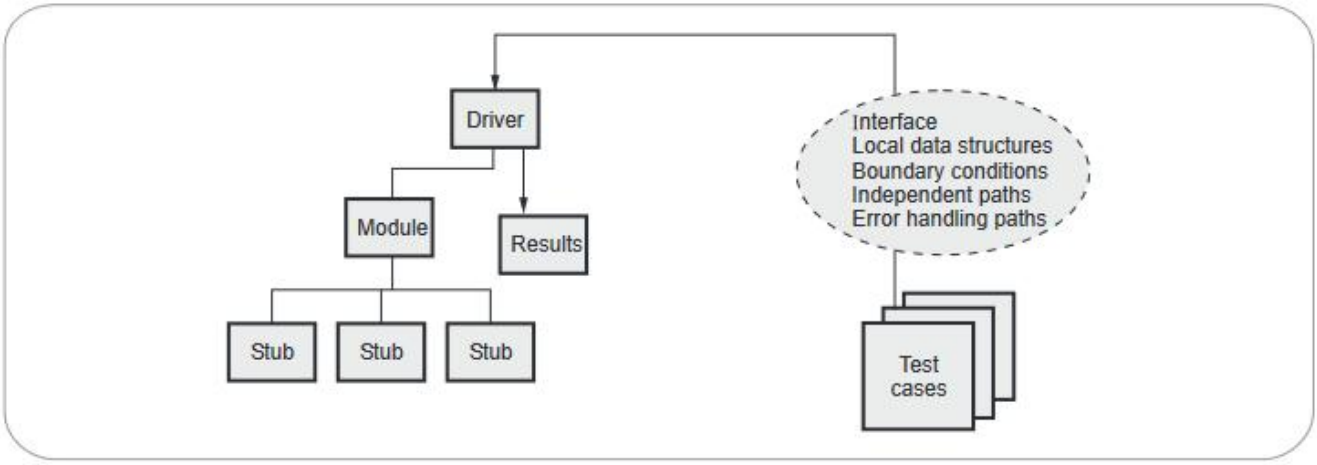


Fig. 3.5.3 Unit testing environment

7. The unit testing is simplified when a component with high cohesion (with one function) is designed. In such a design the number of test cases are less and one can easily predict or uncover errors.

3.5.4 Test Documentation

3.5.4.1 Test Case Template

- Test cases are used to determine the **presence of fault** in the program. Sometimes even if there is some fault in our program the correct output can be obtained for some inputs. Hence it is necessary to exercise those set of inputs for which faults (if any) can be exposed off.
- Executing test cases require money because - i) machine time is required to execute test cases ii) human efforts are involved in executing test cases. Hence in the project testing minimum number of test cases should be there as far as possible.

- The testing activity should involve two goals -
  i) Maximize the number of errors detected.
  ii) Minimize the number of test cases.
- The selection of test case should be such that faulty module or program segment must be exercised by at least one test case.
- Selection of test cases is determined by some criteria which is called as **test selection criterion**. Hence the test selection criterion T can be defined as the set of conditions that must be satisfied by the set of test cases.
- Testing criterion are based on two fundamental properties - **reliability** and **validity**.
- A test criterion is reliable if all the test cases detect same set of errors.
- A test criterion is valid if for any error in the program there is some set which causes error in the program.
- For testing criteria there is an important theorem - "if testing criterion is valid and reliable if a set satisfying testing criterion succeeds then that means program contains no errors".
- Generating test cases to satisfy criteria is complex task.
- The test case specification records the results of the testing, the conditions used for testing particular unit. It also specifies the expected test results. It records the outcome of test cases (Pass/Fail). The sample structure of a test case specification is as given below -

Precondition :

Test case id	Test case name	Test case de-scrip-tion	Test steps			Test case status (Pass/ Fail)	Test priority	Defect severity
			Step	Expected	Actual			
1								
2								
3								
4								

- Test case specification is the major activity in the testing process. Careful selection of test cases will help in conducting proper testing.

There are two basic reasons why test case specification should be before using them for testing - Firstly it will assist the tester to reveal as many errors as possible from the program and secondly the high quality code can be produced.

3.5.4.2 Test Plan

Test plan is not a static document. It gets generated during the development. The main purpose of test planning is to describe the product tests and to establish the standards in testing process. The structure of test plan is as given below.

1. Testing process

In this section various phases of testing process are described.

2. Requirements traceability

Testing should be planned to meet all the requirements.

3. Tested items

The tested products of software are tested.

4. Testing schedule

It includes testing schedule and resource allocation for this schedule.

5. Test recording procedures

After running the tests, it is necessary to record those tests in order to check whether tests are conducted systematically or not.

6. Hardware and software requirements

The required software tools and hardware utilization is specified under this section.

7. Constraints

All the factors affecting the testing process are listed under this section.

3.5.4.3 Introduction to Defect Report

Defect report is a document that identifies and describes the defects detected by the tester.

The purpose of defect report is to state the problem as clearly as possible so that the developer can fix it easily.

In most of the software companies defect report is used as a tool in defect management process.

The template of commonly used defect report is as follows -

ID	Unique identifier given to the defect.
Project	Name of the Project
Product	Name of the product
Release version	Specify the release version of the product.
Module	Specify the module name in which defect is detected.
Build version	Specify the build version of the product where the defect was detected.
Summary	Specify clear and concise summary of defect.
Description	Specify the detailed description of the defect. Describe the defect in simple words. It should be comprehensive.
Steps to replicate	Step by step description of the way to reproduce the defect.
Actual results	Specify the actual results you received when you followed the steps.
Expected results	Specify the expected results from the steps to replicate.
Attachment	Attach any additional information like screenshots and logs.
Remarks	Mention any additional comments on defect.
Defect severity	Mention the severity of defect.
Defect priority	Mention the priority of the defect.
Reported by	The name of the person who report the defect.
Assigned to	The name of the person who is assigned to fix the defect.
Status	Specify the status of the defect.
Fixed build no.	Specify the version of the product where the defect was fixed.

3.5.4.4 Test Summary Report

- Test summary report is a document which contains summary of test activities and final test results.
- After completion of testing cycle, it is very important to communicate the test results and findings to project stakeholders(Stakeholders are people who take part in developing the software product) so that decisions can be made for the software release.

Template : The IEEE template for Test Summary Report is as given below -

Test Summary Report Identifier

[Enter Some type of unique company generated number to identify this summary report, its level and the level of software that it is related to.]

Summary

[Include basic information about what was tested and what happened.]

Project Name : [Project name]

- **System Name :** [System name]
- **Version Number :** [Version number]
- **Test Item :** [This should match the item definitions from the appropriate level test plan that this report is covering. Any variance from the items specified in the test plan should be identified.]
- **Additional Comments :** [Enter any additional comments]

Variances

Document any changes from those areas agreed on in the reference documents, especially in areas that may cause concern to the group accepting the test results.

Support materials and documents for-

1) Change requests

2) Enhancement requests

3) Incident reports

Comprehensive assessment

[Enter a comprehensive assessment of your interpretation of how adequate the test was in light of how thorough the test plan said it should be ? What wasn't tested well enough ?]

Summary of resultss

[Summarize the test results. Include a detailed description of any deviations from the original test plan, design, test case, or expected results. Include any issues or bugs discovered during the test.]

Evaluation

- Based on evaluation of testing, each identified test item should be covered in the evaluation.
- Evaluate exist criteria mentioned in the test plan.
- Evaluate final defect status
- Identify and evaluate product risk, quality of test object, and testing process.

Summary of activities

Cover the planned activities and the changes to those plans especially in areas where the amount of actual effort greatly exceeded the planned effort. Include the reasons for the variances and the possible impact on the testing staff.

[Include

- Staff time used in Hours per day/week
- Elapsed time versus staff time
- Is staff working excess hours per week
- Costs - planned versus actual
- Variances and the reasons for the change such as - change in project scope, requirements changes, newly introduced defects, and so on.]

Approvals

The undersigned acknowledge they have reviewed the *<Project Name>* **Test Summary Report** and agree with the approach it presents. Changes to this **Test Summary Report** will be coordinated with and approved by the undersigned or their designated representatives.

[List the individuals whose signatures are desired. Examples of such individuals are Quality Manager or Tester. Add additional lines for signature as necessary. Although signatures are desired, they are not always required to move forward with the practices outlined within this document.]

Signature: Date: _____

Print Name: _____

Title: _____

Role: _____

Board Questions

1. *List four objectives of testing.* **MSBTE : Summer 15, Winter-17,Marks 4**
2. *State eight characteristics of software bugs.* **MSBTE : Summer-15,18, Winter-17, Marks 4**
3. *Describe the attributes of a good test.* **MSBTE : Winter-15, Marks 4**
4. *Explain the testing concept with its Testing Principles. (any four principles)* **MSBTE : Summer-18, Marks 4**
5. *Compare white box and black box testing.* **MSBTE : Summer 15,18,Marks 4**
6. *Explain white box testing.* **MSBTE : Winter-15, Marks 4**
7. *Describe white box and black box testing of software.* **MSBTE : Winter-16, Marks 4**
8. *List the objective of black box testing.* **MSBTE : Summer-17, Marks 4**
9. *Explain unit testing.* **MSBTE : Winter-15,17, Marks 4**
10. *What aspects of the software are tested in Unit Testing ?* **MSBTE : Summer-16, Marks 4**
11. *Describe in brief four level testing process in test execution.* **MSBTE : Winter-16, 17, Marks 4**
12. *Draw stub and driver mechanism of unit testing and enlist various types of errors detected by unit testing.* **MSBTE : Summer-18, Marks 4**
13. *Explain test case design in detail.* **MSBTE : Summer-17, Marks 4**
14. *List various testing characteristics.* **MSBTE : Summer-17, Marks 4**
15. *What do you mean by good test ?* **MSBTE : Winter-17, Marks 4**

❑❑❑

UNIT- IV

4 Software Project Estimation

4.1 Management Spectrum

MSBTE : Winter-16, Summer-18

Effective software project management focuses four P's i.e. **people, product, process and project.** The successful project management is done with the help of these four factors where the order of these elements is not arbitrary. Project manager has to motivate the **communication between stakeholders.** He should also prepare a **project plan** for the success of the product.

4.1.1 The People

People factor is an important issue in software industry. There is a strong need for motivated and highly skilled people for developing the software product. The **S**oftware **E**ngineering **I**nstitute (SEI) has developed the **P**eople **M**anagement **C**apability **M**aturity **M**odel (PM-CMM)

By using PM-CMM model software organizations become capable for undertaking complex applications which ultimately attracts or motivates the talented people.

Following are some key practice areas for software people -

- Recruitment
- Selection
- Performance management
- Training compensation
- Career development
- Organization and work design
- Culture development.

4.1.2 The Product

- Before planning the project three important tasks need to be done -
  (1) Product objectives and scope must be established.
  (2) Alternative solutions should be considered.
  (3) Technical and management constraints must be identified.
- The software developer and customer must communicate with each other in order to define the objectives and scope of the product.
- This is done as the first step in requirement gathering and analysis.
- The scope of the project identifies primary data, functions and behaviour of the product.
- After establishing the objectives and scope of the product the alternative solutions are considered.
- Finally, the constraints imposed by - delivery deadline or budgetary restrictions, personal availability can be identified.

4.1.3 The Process

- The software process provides the framework from which the software development plan can be established.
- There are various framework activities that needs to be carried out during the software development process. These activities can be of varying size and complexities.
- Different task sets-tasks, milestones, work products and quality assurance points enable framework activities to adapt the software requirements and certain characteristics of software project.
- Finally, umbrella activities such as software quality assurance (SQA) and Software Configuration Management (SCM) are conducted. These umbrella activities depend upon the framework activities.

4.1.4 Project

For a successful software project, it is necessary to understand the mistakes in the project and how to correct them. John Reel defined ten symptoms to indicate why the software projects fail -

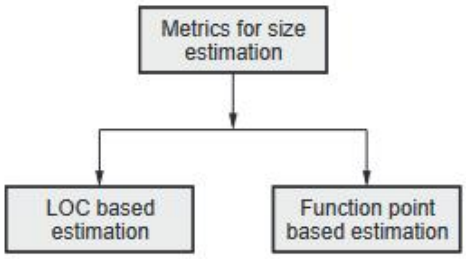
1. Software developers do not understand the customer's need.
2. The scope of the project is not defined properly.
3. Change management is done poorly.
4. Business needs change very often.
5. The technological changes are quite often.
6. Users are not co-operative.
7. Unrealistic deadlines are set.
8. Projects do not get sponsored or Sponsorship gets lost.
9. Lack of skilled people in the project.
10. Project managers do not adopt the best practices for the projects.

Board Questions

1. *Define management spectrum and enlist characteristics of software.*
   MSBTE : Summer-18, Marks 4
2. *Explain 4 P's software project spectrum.*
   MSBTE : Summer-18, Marks 4
3. *Explain the following management spectrum :*
   i) The Process ii) The Project
   MSBTE : Winter-16, Marks 4

4.2 Metrics for Size Estimation

There are two types of metrics used for size estimation of the project -



4.2.1 LOC based Estimation

- Size oriented measure is derived by considering the size of software that has been produced.
- The organization builds a simple record of size measure for the software projects. It is built on past experiences of organizations.
- It is a direct measure of software

Project	LOC	Effort	Cost ($)	Doc. (pgs.)	Errors	Defects	People
ABC	10,000	20	170	400	100	12	4
PQR	20,000	60	300	1000	129	32	6
XYZ	35,000	65	522	1290	280	87	7
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

Table 4.2.1 Size measure

- A simple set of size measure that can be developed is as given below :
  - Size = Kilo Lines of Code (KLOC)
  - Effort = Person/month
  - Productivity = KLOC/person-month
  - Quality = Number of faults/KLOC
  - Cost = $/KLOC
  - Documentation = Pages of documentation / KLOC
- The size measure is based on the lines of code computation. The lines of code is defined as one line of text in a source file.
- While counting the lines of code the Simplest Standard is :
  - Don't count blank lines.
  - Don't count comments.
  - Count everything else.
- The size oriented measure is not universally accepted method.

Advantages

1. Artifact of software development which is easily counted.
2. Many existing methods use LOC as a key input.
3. A large body of literature and data based on LOC already exists.

Disadvantages

1. This measure is dependent upon the programming language.
2. This method is well designed but shorter program may get suffered.
3. It does not accommodate non procedural languages.
4. In early stage of development it is difficult to estimate LOC.

4.2.2 Function Points

- The oriented model is based on functionality of the delivered application.
- These are generally independent of the programming language used.
- This method is developed by Albrecht in 1979 for IBM. It uses function points.
- Function points are derived using :
  1. Countable measures of the software requirements domain
  2. Assessments of the software complexity.

How to calculate function point ?

- The data for following information domain characteristics are collected :
  1. **Number of user inputs -** Each user input which provides distinct application data to the software is counted.
  2. **Number of user outputs -** Each user output that provides application data to the user is counted, e.g. screens, reports, error messages.
  3. **Number of user inquiries -** An on-line input that results in the generation of some immediate software response in the form of an output.
  4. **Number of files -** Each logical master file, i.e. a logical grouping of data that may be part of a database or a separate file.
  5. **Number of external interfaces -** All machine-readable interfaces that are used to transmit information to another system are counted.
- The organization needs to develop criteria which determine whether a particular entry is simple, average or complex.
- The weighting factors should be determined by observations or by experiments.

Domain Characteristics	Count		Weighting factor			Count
			Simple	Average	Complex	
Number of user input		X	3	4	6	
Number of user output		X	4	5	7	
Number of user inquiries		X	3	4	6	
Number of files		X	7	10	15	
Number of external interfaces		X	5	7	10	
Count Total						

- The count table can be computed with the help of above given table.
- Now the software complexity can be computed by answering following questions. These are **complexity adjustment values.**
  1. Does the system need reliable backup and recovery ?
  2. Are data communications required ?
  3. Are there distributed processing functions ?
  4. Is performance of the system critical ?
  5. Will the system run in an existing, heavily utilized operational environment ?
  6. Does the system require on-line data entry ?
  7. Does the on-line data entry require the input transaction to be built over multiple screens or operations ?
  8. Are the master files updated on-line ?
  9. Are the inputs, outputs, files or inquiries complex ?
  10. Is the internal processing complex ?
  11. Is the code which is designed being reusable ?
  12. Are conversion and installation included in the design ?
  13. Is the system designed for multiple installations in different organizations ?
  14. Is the application designed to facilitate change and ease of use by the user ?
- Rate each of the above factors according to the following scale :

- Function Points (FP) = Count total x (0.65 + (0.01 x Sum(F_i))

- Once the functional point is calculated then we can compute various measures as follows
  - **Productivity** = FP/person-month
  - **Quality** = Number of faults/FP
  - **Cost** = $/FP
  - **Documentation** = Pages of documentation/FP.

Advantages

1. This method is independent of programming languages.
2. It is based on the data which can be obtained in early stage of project .

Disadvantages

1) This method is more suitable for business systems and can be developed for that domain.
2) Many aspects of this method are not validated.
3) The functional point has no significant meaning. It is just a numerical value.

Comparison between size oriented and function oriented metrics

Sr. No.	Size oriented metrics	Function oriented metrics
1.	Size oriented software metrics is by considering the size of the software that has been produced.	Function oriented metrics use a measure of functionality delivered by the software.
2.	For a size oriented metric the software organization maintains simple records in tabular form. The typical table entries are : Project name, LOC, Effort, Pages of documents, errors, defects, total number of people working on project.	Most widely used function oriented metric is the function point (FP) computation of the function point is based on characteristics of software's information domain and complexity.

Ex. 4.2.1 *Study of requirement specification for ABC project has produced following results : Need for 7 inputs, 10 outputs, 6 inquiries, 17 files and 4 external interfaces. Input and external interface function point attributes are of average complexity and all other function points attributes are of low complexity.*
Determine adjusted function points assuming complexity adjustment value is 32.

Sol. : Given that :

7 inputs

10 Outputs

6 inquiries

17 files

4 external interfaces

Average complexity for inputs and external interfaces. Low complexity for remaining parameters.

Adjusted function point value $\sum (F_i) = 32$.

Let us calculate count total value.

Measurement parameters	Count		Weighting factor			
		X	Simple	Average	complex	
Number of user inputs.	7	X		4		28
Number of user outputs.	10	X	4			40
Number of user inquiries.	6	X	3			18
Number of files	17	X	7			119
Number of external interfaces.	4	X		7		28
Count total.						233

$$\text{Function point} = \text{Count total} \times [0.65 + 0.01 \times \sum(F_i)]$$

$$= 233 \times [0.65 + 0.01 \times 32]$$

$$= 233 \times [0.65 + 0.32]$$

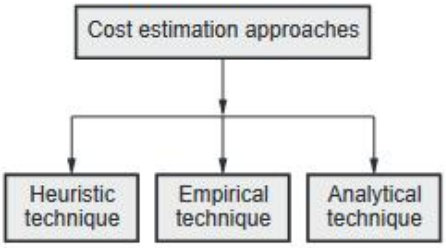
$$= 233 \times 0.97$$

$$FP = 226.01$$

Hence adjusted function point is **226.01.**

4.3 Project Cost Estimation Approach

Project estimation is one of the basic activity in project planning and measurement. There are three categories of estimation techniques -



4.3.1 Overview of Heuristic Technique

- This method makes use of previous project's estimated cost.
- This method assumes the relationships among the different project parameters. These parameters are then correlated using suitable mathematical expressions.
- Once the basic parameters are known, the other parameters can be easily determined by substituting the value of basic parameters in mathematical expression.
- COCOMO model is an example of Heuristic technique.

4.3.2 Analytical and Empirical Estimation

Empirical Estimation

- In empirical estimation technique an **educated guess of project parameters is made.** Hence empirical estimation model is based on common sense.
- However as there are many activities involved in empirical estimation techniques, this **technique is formalized.**
- **Delphi technique** is used for empirical estimation techniques.
- This method involves **more interactions and communications** between those who are participating. The procedure is as follows.

1. The co-ordinator presents a specification and estimation form to each expert.
2. Co-ordinator calls a group meeting in which the experts discuss estimation issues with the coordinator and each other.
3. Experts fill out forms anonymously.
4. Co-ordinator prepares and distributes a summary of the estimates.
5. The co-ordinator then calls a group meeting. In this meeting the experts mainly discuss the points where their estimates vary widely.
6. The experts again fill out forms anonymously.
7. Again co-ordinator edits and summarizes the forms, repeating the steps 5 and 6 until the coordinator is satisfied with overall prediction synthesized from experts.

- The key to this technique is in expert co-ordinator. The co-ordinator must be talented enough to synthesize the diverse and wide ranging statements.
- This method is successful for technical forecasting.

Analytical Estimation

In analytical estimation technique, the results are derived by making certain basic assumptions about the project. Hence analytical estimation technique have some **scientific basis**. **Halstead's software science** is based on analytical estimation model.

4.3.2.1 Halstead's Software Science

Halstead's complexity measurement was developed to measure a program module's complexity directly from source code, with emphasis on computational complexity.

The Halstead's measures are based on four scalar numbers derived directly from a program's source code :

n_1 is number of distinct operators,

n_2 is number of distinct operands,

N_1 is total number of operators,

N_2 is total number of operands,

From these numbers, five measures are derived :

Measure	Symbol	Formula
Program length	N	$N = N_1 + N_2$
Program vocabulary	n	$n = n_1 + n_2$
Volume	V	$V = N * (\log_2 n)$
Difficulty	D	$D = (n_1/2) * (N_2/2)$
Effort	E	$E = D * V$

Halstead's uses certain measures such as program length, program vocabulary, volume, difficulty and effort for the given algorithm. By this Halstead's is trying to show that the program length can be calculated, volume of the algorithm can be estimated. The above given table shows how actually these measures can be obtained.

The Halstead's measures are applicable to operational systems and to development efforts once the code has been written. Thus using Halstead's measurement experimental verification can be performed in software science.

Program length

The length of a program is total usage of operators and operands in the program.

$$\text{Length} = N_1 + N_2$$

Program vocabulary

The program vocabulary is the number of unique operators and operands used in the program.

$$\text{Vocabulary} \cdot n = n_1 + n_2$$

Program volume

The program volume can be defined as minimum number of bits needed to encode the program.

$$V = N \log_2 n$$

Length estimation

$$N = n_1 \log_2 n_1 + n_2 \log_2 n_2$$

Guideline for calculating operands and operators :

1. All the variables and constants are considered as operands.
2. Local variables with same name, if occuring in different functions are counted as unique operand.
3. Function calls are considered as operators.
4. The looping statements, do ... while, while, for are operators. The statements if, if ... else are operators. The switch ... case statements are considered as operators.
5. The reserve words, return, default, continue, break, sizeof are all operators.
6. The brackets, commas, semicolons are operators.
7. The unary and binary operators are considered as operators. The & is considered as operator.
8. In arrays, array name and index are considered as operands and [ ] is considered as operator.
9. All hash directives can be ignored.
10. Comments are not considered.
11. In Goto statement, goto is considered as operator and label as operand.

Ex. 4.3.1 *Obtain Halstead's length and volume measure for following C function.*

```
void swap (int a [ ], int i)
{
   int temp ;
   temp = a[i] ;
   a[i] = a[i+1] ;
   a[i+1] = temp ;
}
```

Sol. : We will first find out the operands and operators from above function along with their occurrences.

Operands	Occurrences	Operators	Occurrences
swap	1	( )	1
a	5	{ }	1
i	5	void	1
temp	3	int	3
1	2	[ ]	5
		,	1
		;	4
		=	3
		+	2
$n_1 = 5$	$N_1 = 16$	$n_2 = 9$	$N_2 = 21$

$N = N_1 + N_2 = 16 + 21 = 37$ $n = n_1 + n_2 = 14$

Estimated length $= n_1 \log_2 n_1 + n_2 \log_2 n_2$

$= 5 \log_2 5 + 9 \log_2 9$

$= 5 * 2.32 + 9 * 2.19$

$$= 11.60 + 19.77 = 31.37$$

$$\text{Volume} = N \times \log_2 n$$

$$= 37 * \log_2 (14)$$

$$\text{Volume} = 37 * 2.63 = 97.64$$

4.4 COCOMO

COCOMO is one of the most widely used software estimation models in the world. This model is developed in 1981 by **Barry Boehm** to give an estimate of the number of man-months it will take to develop a software product. COCOMO predicts the efforts and schedule of a software product based on **size** of the software. COCOMO stands for "**CO**nstructive **CO**st **MO**del".

COCOMO has three different models that reflect the complexity -

- Basic model
- Intermediate model
- Detailed model.

Similarly there are three classes of software projects.

1) Organic mode : In this mode, relatively small, simple software projects with a small team are handled. Such a team should have good application experience to less rigid requirements.

2) Semi-detached projects : In this class an intermediate projects in which teams with mixed experience level are handled. Such projects may have mix of rigid and less than rigid requirements.

3) Embedded projects : In this class, projects with tight hardware, software and operational constraints are handled.

Let us understand each model in detail.

1) Basic Model : The basic COCOMO model estimates the software development effort using only **Lines of Code**. Various equations in this model are -

$$E = a_b (KLOC)^{b_b}$$

$$D = C_b (E)^{d_b}$$

$$P = E/D$$

Where E is the **effort** applied in person-months.

D is the development time in chronological months.

KLOC means kilo line of code for the project.

P is total number of persons required to accomplish the project.

The coefficients a_b, b_b, c_b, d_b for three modes are as given below.

Software projects	a_b	b_b	c_b	d_b
Organic	2.4	1.05	2.5	0.38
Semi-detached	3.0	1.12	2.5	0.35
Embedded	3.6	1.20	2.5	0.32

Table 4.4.1

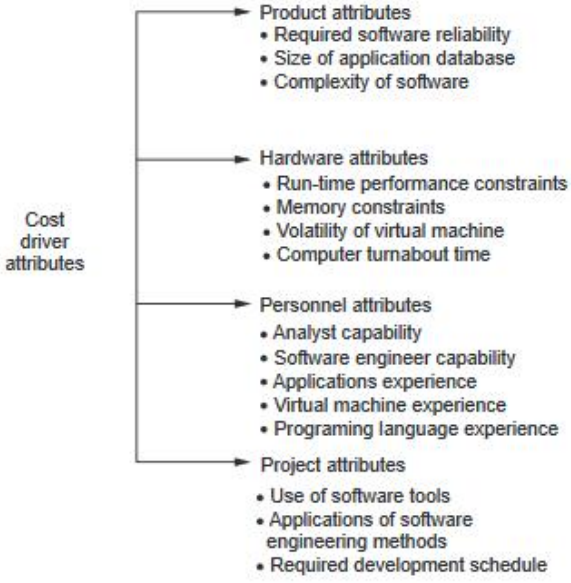
Merits of basic COCOMO model

Basic COCOMO model is good for quick, early, rough order of magnitude estimates of software project.

Limitations of basic model

1. The accuracy of this model is limited because it does not consider certain factors for cost estimation of software. These factors are hardware constraints, personal quality, and experience, modern tachniques and tools.
2. The estimates of COCOMO model are within a factor of 1.3 only 29 % of the time and within the factor of 2 only 60 % of time.

Example



Consider a software project using semi-detached mode with 30,000 lines of code. We will obtain estimation for this project as follows -

i) Effort estimation

$$E = a_b (KLOC)^{b_b}$$

i.e. $E = 3.0 (30)^{1.12}$ where lines of code = 30000 = 30 KLOC

$$E = 135 \text{ person-month}$$

ii) Duration estimation

$$D = C_b (E)^{d_b}$$

$$= 2.5 (135)^{0.35}$$

$$D = 14 \text{ months}$$

iii) Persons estimation

$$P = E/D$$

$$= 135/14$$

$$P = 10 \text{ persons approximately}$$

2) Intermediate Model

This is an extension of Basic COCOMO model. This estimation model makes use of set of "Cost driver attributes" to compute the cost of software.

Now these 15 attributes get a 6-point scale ranging from "very low" to "extra high". These ratings can be viewed as

The effort multipliers for each cost driver attribute is as given in following table. The product of all effort multipliers result in "Effort Adjustment Factor" (EAF).

Cost drivers	Ratings					
	Very low	Low	Nominal	High	Very high	Extra high
Product attributes						
Required software reliability	0.75	0.88	1.00	1.15	1.40	
Size of application database		0.94	1.00	1.08	1.16	
Complexity of software	0.70	0.85	1.00	1.15	1.30	1.65
Hardware attributes						
Run-time performance constraints			1.00	1.11	1.30	1.66
Memory constraints			1.00	1.06	1.21	1.56
Volatility of virtual machine		0.87	1.00	1.15	1.30	
Computer turnabout time		0.87	1.00	1.07	1.15	

Personnel attributes						
Analyst capability	1.46	1.19	1.00	0.86	0.71	
Software engineer capability	1.42	1.17	1.00	0.86	0.70	
Applications experience	1.29	1.13	1.00	0.91	0.82	
Virtual machine experience	1.21	1.10	1.00	0.90		
Programming language experience	1.14	1.07	1.00	0.95		
Project attributes						
Use of software tools	1.24	1.10	1.00	0.91	0.82	
Applications of software engineering methods	1.24	1.10	1.00	0.91	0.83	
Required development schedule	1.23	1.08	1.00	1.04	1.10	

Table 4.4.2

The formula for effort calculation can be -

$$E = a_i (KLOC)^{b_i} \cdot EAF \text{ person-months}$$

The values for a_i and b_i for various class of software projects are -

Software project	a_i	b_i
Organic	3.2	1.05
Semi-detached	3.0	1.12
Embedded	2.8	1.20

Table 4.4.3

The duration and person estimate is same as in basic COCOMO model. i.e.

$D = c_b (E)^{d_b}$ months i.e. use values of c_b and d_b coefficients that are in Table 4.4.1

$P = E/D$ persons

Merits of Intermediate Model

1. This model can be applied to almost entire software product for easy and rough cost estimation during early stage.
2. It can also be applied at the software product component level for obtaining more accurate cost estimation.

Limitations of Intermediate Model

1. The estimation is within 20 % of actual 68 % of the time.
2. The effort multipliers are not dependent on phases.
3. A product with many components is difficult to estimate.

Example

Consider a project having 30,000 lines of code which is an **embedded** software with critical area hence realiability is high. The estimation can be

$$E = a_i (KLOC)^{b_i} \cdot EAF$$

As reliability is high, EAF = 1.15 (product attribute)

$a_i = 2.8$, $b_i = 1.20$ } for embedded software

$\therefore \quad E = 2.8 (30)^{1.20} * 1.15$

$= 191$ person-month

$D = c_b (E)^{d_b} = 2.5 (191)^{0.32}$

$= 13$ months approximately

$P = E/D$

$= 191/13$

$P = 15$ persons approximately

3) Detailed COCOMO Model

The detailed model uses the same equations for estimation as the Intermediate Model. But detailed model can estimate the effort (E), duration (D) and persons (P) of each of development phases, subsystems, modules.

The experimentation with different development strategies is allowed in this model.

Four phases used in detailed COCOMO model are -

1. Requirements planning and product design (RPD)
2. Detailed design (DD)
3. Code and unit test (CUT)
4. Integrate and test (IT)

The effort multipliers for detailed COCOMO are

Phases	Very low	Low	Nominal	High	Very high
RPD	1.80	0.85	1.00	0.75	0.55
DD	1.35	0.85	1.00	0.90	0.75
CUT	1.35	0.85	1.00	0.90	0.75
IT	1.50	1.20	1.00	0.85	0.70

Using these detailed cost drivers, an estimate is determined for each phase of the lifecycle.

4.5 COCOMOII

COCOMO II is applied for modern software development practices addressed for the projects in 1990's and 2000's.

The sub-models of COCOMO II model are –

1. Application composition model

- For estimating the efforts required for the prototyping projects and the projects in which the existing software components are used application-composition model is introduced.
- The estimation in this model is based on the number of application points. The application points are similar to the object points.
- This estimation is based on the level of difficulty of object points.
  Boehm has suggested the object point productivity in the following manner.

Developers experience and capability	Very low	Low	Nominal	High	Very high
CASE maturity	Very low	Low	Nominal	High	Very high
Productivity (NOP/Month)	4	7	13	25	50

- Effort computation in application-composition model can be done as follows -

  $$PM = (NAP^{(1-\%reuse/100)}) / PROD$$

  where

  PM means effort required in terms of person-months.

  NAP means number of application points required.

  % reuse indicates the amount of reused components in the project. These reusable components can be screens, reports or the modules used in previous projects.

  PROD is the object point productivity. These values are given in the above table.

2. An early design model

- This model is used in the early stage of the project development. That is after gathering the user requirements and before the project development actually starts, this model is used. Hence approximate cost estimation can be made in this model.
- The estimation can be made based on the functional points.
- In early stage of development different ways of implementing user requirements can be estimated.
- The effort estimation (in terms of person month) in this model can be made using the following formula :

$$\textbf{Effort} = \mathbf{A} \times \textbf{size}^{\mathbf{B}} \times \mathbf{M}$$

where

Boehm has proposed the value of coefficient **A = 2.94**.

Size should be in terms of Kilo Source Lines Of Code i.e. KSLOC. The lines of code can be computed with the help of function point.

The value of **B** is varying from 1.1 to 1.24 and depends upon the project.

M is based on the characteristics such as

1. Product reliability and complexity (RCPX)
2. Reuse required (RUSE)
3. Platform difficulty (PDIF)
4. Personnel capability (PERS)
5. Personnel experience (PREX)
6. Schedule (SCED)
7. Support facilities (FCIL)

These characteristics values can be computed on the following scale -

- Hence the effort estimation can be given as

$$\mathbf{PM} = 2.94 \times \textbf{size}^{\mathbf{B}} \times \mathbf{M}$$

$$\mathbf{M} = \mathbf{RUSE} \times \mathbf{PDIF} \times \mathbf{PERS} \times \mathbf{PREX} \times \mathbf{SCED} \times \mathbf{FCIL}$$

3. A reuse model

This model considers the systems that have significant amount of code which is reused from the earlier software systems. The estimation made in reuse model is nothing but the efforts required to integrated the reused models into the new systems.

- There are two types of reusable codes : black box code and white box code. The black box code is a kind of code which is simply integrated with the new system without modifying it. The white box code is a kind of code that has to be modified to some extent before integrating it with the new system, and then only it can work correctly.
- There is third category of code which is used in reuse model and it is the code which can be generated automatically. In this form of reuse the standard templates are integrated in the generator. To these generators, the system model is given as input from which some additional information about the system is taken and the code can be generated using the templates.
- The efforts required for automatically the generated code is

$$PM = (ASLOC \times AT/100)/ATPROD$$

where

AT is percentage of automatically generated code.

ATPROD is the productivity of engineers in estimating such code

- Sometimes in the reuse model some white box code is used along with the newly developed code. The size estimate of newly developed code is equivalent to the reused code. Following formula is used to calculate the effort in such a case –
- ESLOC = ASLOC × (1–AT/100) X AAM

where

ESLOC means equivalent number of lines of new source code.

ASLOC means the source lines of code in the component that has to be adapted.

AAM is adaptation Adjustment multiplier. This factor is used to take into account the efforts required to reuse the code.

4. Post architecture model

- This model is a detailed model used to compute the efforts. The basic formula used in this model is

$$\text{Effort} = \mathbf{A} \times \mathbf{Size}^{\mathbf{B}} \times \mathbf{M}$$

- In this model efforts should be estimated more accurately. In the above formula **A** is the amount of code. This code size estimate is made with the help of three components –
  1. The estimate about **new** lines of code that is added in the program.
  2. Equivalent number of source lines of code (ESLOC) used in **reuse model**.
  3. Due to changes in requirements the lines of code get modified. The estimate of amount of **code** being **modified**.

The exponent term **B** has three possible values that are related to the levels of project complexity. The values of B are continuous rather than discrete. It depends upon the five scale factors. These scale factors vary from very low to extra high (i.e. from 5 to 0).

- These factors are –

Scale factor for component B	Description
Precedentedness	This factor is for previous experience of organisation. Very low means no previous experience and high means the organisation knows the application domain.
Development flexibility	Flexibility in development process. Very low means the typical process is used. Extra high means client is responsible for defining the process goals.
Architecture/risk resolution	Amount of risk that is allowed to carry out. Very low means little risk analysis is permitted and extra high means high risk analysis is made.
Team cohesion	Represents the working environment of the team. Very low cohesion means poor communication or interaction between the team members and extra high means there is no communication problem and team can work in a good spirit.
Process maturity	This factor affects the process maturity of the organisation. This value can be computed using Capacity Maturity Model (CMM) questionnaire, for computing the estimates CMM maturity level can be subtracted from 5.

- Add up all these rating and then whatever value you get, divide it by 100. Then add the resultant value to 1.01 to get the exponent value.
- This model makes use of 17 cost attributes instead of seven. These attributes are used to adjust initial estimate.

Cost attribute	Type of attribute	Purpose
RELY	Product	System reliability that is required
CPLX	Product	Complexity of system modules
DATA	Product	Size of the data used from database
DOCU	Product	Some amount of documentation used
RUSE	Product	Percentage of reusable components
TIME	Computer	Amount of time required for execution
PVOL	Computer	Volatility of development platform
STOR	Computer	Memory constraint
ACAP	Personnel	Project analyst's capability to analyse the project

PCAP	Personnel	Programmer capability
PCON	Personnel	Personnel continuity
PEXP	Personnel	Programmer's experience in project domain
LTEX	Personnel	Experience of languages and tools that are used
AEXP	Personal	Analyst's experience in project domain.
TOOL	Project	Use of software tools
SCED	Project	Project schedule compression
SITE	Project	Quality of inter-site and multi-site working

Ex. 4.5.1 *Using COCOMO, estimate time required for the following :*

1) A semi-detached model of software project of 2000 lines.

2) An embedded model of software of 30,000 lines.

3) An organic model of software of one lakh lines.

4) An organic model of software of 10 lakh lines.

Sol. : To estimate time using basic model of COCOMO following formula can be used.

$$E = a_b(KLOC)^{b_b}$$

where E is the effort in person-month.

$$D = c_b(E)^{d_b}$$

where D is development time in chronological months.

$$P = E/D$$

where P is total number of persons involved in the project. The constants are

System	a_b	b_b	c_b	d_b
Organic system	2.4	1.05	2.5	0.38
Semidetached system	3.0	1.12	2.5	0.35
Embedded system	3.6	1.20	2.5	0.32

1) Given that, System = Semi detached

Lines of code = 2000 lines = 2 KLOC

$\therefore$ $E = a_b(KLOC)^{b_b}$

$E = 3.0\ (2)^{1.15}$

$E = 6.65$ person-month

$\therefore$ $D = c_b(E)^{d_b}$

$D = 4.8$ months

$\therefore$ $P = E/D$

$P = 1.3 \approx 1$ person

Thus 1 person can handle this project within approximately **5 months.**

2) Given that, System = Embedded

Lines of code = 30,000 lines = 30 KLOC

$\therefore$ $E = a_b(KLOC)^{b_b}$

$= 3.6\ (30)^{1.20}$

$E \approx 213$ person - month

$D = c_b(E)^{d_b}$

$= 2.5\ (213)^{0.32}$

$\therefore$ $D \approx 14$ months

$P = E/D$

$\approx 213/14$

≈ 15 persons.

That means 15 persons can complete this project within approximately **14 months.**

3) Given that, System = Organic

Lines of code = 1 lakh = 100 KLOC

$\therefore$ $E = a_b(KLOC)^{b_b}$

$= 2.4\ (100)^{1.05}$

$E \approx 302$ person-month

$D = c_b(E)^{d_b} = 2.5\ (302)^{0.38}$

$\therefore$ ≈ 21 months

$P = E/D$

$= 302/21$

≈ 14 persons.

That means this project can be completed within 21 months by 14 persons, approximately,

4) Given that, System = Organic

Lines of code = 10 lakh = 1000 KLOC

$\therefore$ $E = a_b(KLOC)^{b_b}$

$= 2.4\ (1000)^{1.05}$

$= 3390$ person - month

$D = c_b(E)^{d_b}$

$= 2.5\ (3390)^{0.38}$

≈ 55 months

$\therefore$ $P = E/D$

$\approx 3390/55$

$= 61$ persons

This project can be completed within 55 months by 61 people approximately.

4.6 Risk Management

MSBTE : MSBTE : Summer-15, 16, 17, 18, Winter-15, 16

Definition of risk : The risk denotes the uncertainty that may occur in the choices due to past actions and risk is something which causes heavy losses.

Definition of risk management : Risk management refers to the process of making decisions based on an evaluation of the factors that threats to the business.

Various activities that are carried out for risk management are -

1. Risk identification 2. Risk projection
3. Risk refinement
4. Risk mitigation, monitoring and management.

4.6.1 Software Risks

There are two characteristics of the risks

1. The risk may or may not happen. It shows the **uncertainty** of the risks.
2. When risks occur, unwanted consequences or **losses** will occur.

Different types of risk

1. **Project risk**

   Project risks arise in the software development process then they basically affect budget, schedule, staffing, resources, and requirements. When project risks become severe then the total cost of project gets increased.

2. **Technical risk**

   These risks affect quality and timeliness of the project. If technical risks become reality then potential design implementation, interface, verification and maintenance problems gets created. Technical risks occur when problem becomes harder to solve.

3. **Business risk**

   When feasibility of software product is in suspect then business risks occur. Business risks can be further categorized as

   i) Market risk - When a quality software product is built but if there is no customer for this product then it is called market risk (i.e. *no market for the product).*

   ii) Strategic risk - When a product is built and if it is not following the company's business policies then such a product brings strategic risks.

   iii) Sales risk - When a product is built but how to sell is not clear then such a situation brings sales risk.

   iv) Management risk - When senior management or the responsible staff leaves the organization then management risk occurs.

   v) Budget risk - Losing the overall budget of the project is called budget risk.

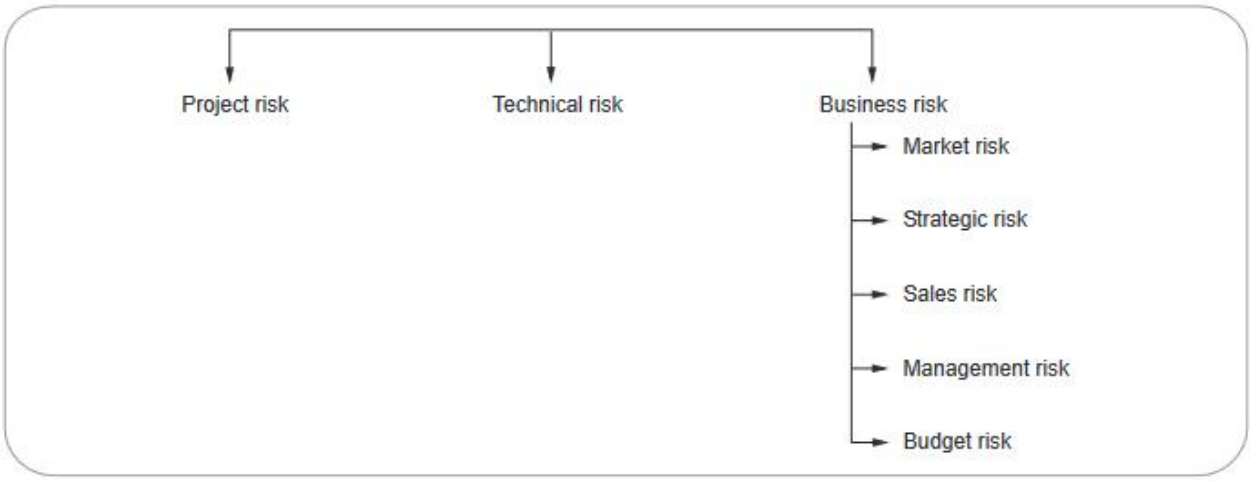


Fig. 4.6.1 Categorization of risk

Another categorization of risk proposed by **Charette** is -

Known risks are those risk that are identified after evaluating the project plan. These risks can also be identified from other sources such as environment in which the product gets developed, unrealistic dead lines, poor requirement specification and software scope. There are two types of known risks - *predictable* and *unpredictable* risks.

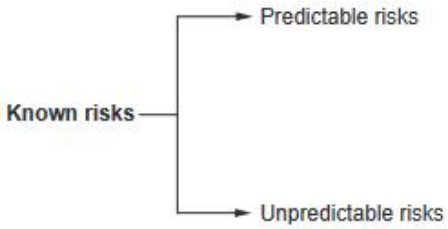


Fig. 4.6.2

Predictable risks are those risks that can be identified in advance based on past project experience. For example : Experienced and skilled staff leaving in between or improper communication with customer resulting in poor requirement specification.

Unpredictable risks *are* those risks that can not be guessed earlier.

For example certain changes in Government policies may affect the business project.

4.6.2 Risk Identification

Risk identification can be defined as the efforts taken to specify threats to the project plan. Risks identification can be done by identifying the known and predictable risks.

The risk identification is based on two approaches

1. Generic risk identification - It includes potential threat identification to software project.
2. Product-specific risk identification - It includes product specific threat identification by understanding people, technology and working environment in which the product gets built.

Normally the risk identification is done by the project manager who follows following steps -

Step 1 : Preparation of risk item check list

The risk items can be identified using following known and predictable components

i) Product size - The risk items based on overall size of the software product is identified.

ii) Business impact - Risk items related to the marketplace or management can be predicted.

iii) Customer characteristics - Risks associated with customer-developer communication can be identified.

iv) Process definition - Risks that get raised with the definition of software process. This category exposes important risks items because whichever

is the process definition made, is then followed by the whole team.

v) Development environment - The risks associated with the technology and tool being used for developing the product.

vi) Staff size and experience - Once the technology and tool related risks items are identified it is essential to identify the risk associated with sufficient highly experienced and skilled staff who will do the development.

vii) Technology to be built - complexity of the system should be understood and related risk items needs to be identified.

After preparing a risk item checklist a questionnaire is prepared. These set of questions should be answered and based on these answers the impact or seriousness of particular risk item can be judged.

Step 2 : Creating risk components and drivers list.

The set of risk components and drivers list is prepared along with their probability of occurrence. Then their impact on the project can be analysed.

Let us understand which are the risk components and drivers.

4.6.3 Risk Projection

The risk projection is also called **risk estimation**.

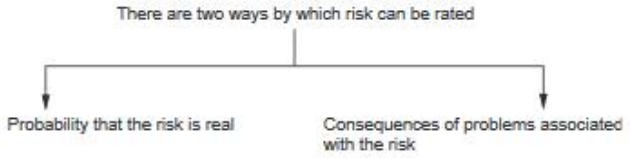


Fig. 4.6.3

The project planner, technical staff, project manager performs following steps to perform following steps for risk projection -

- Establish a scale that indicates the probability of risk being real.
- Enlist the consequences of the risk.
- Estimate the impact of the risk on the project and product.
- Maintain the overall accuracy of the risk projection in order to have clear understanding of the software that is to be built.

These steps help to prioritize the risks. Once the risks are prioritized then it becomes easy to allocate the resources for handling them.

4.6.4 Risk Assessment

- Risk assessment is done during the **project planning.**
- In this phase the risks **are identified, analysed and then prioritized** on the basis of analysis.
- The risk assessment is done throughout the project. It is most needed at the starting phases of project.
- The **goal of the risk assessment** is to prioritize the risks that require an attention.

Risk identification

- Risk identification is the first step in risk assessment, which identifies all the different risks for a particular project.
- Various methods that can be used to identify the risks are -
  - Preparing **checklists** for identifying the risks,
  - Conducting **surveys** and meetings,
  - Having **brainstorming sessions,**
  - **Reviewing** of plans, processes, and work products.
- Based on the surveys, **Bohem** has produced a list of top 10 risks items. This list helps in identifying the risks in the project.

Sr. No.	Risk Item	Management Technique
1.	Personnel Shortfall	Training the people, recruiting top talent people, Key personnel agreement.
2.	Unrealistic schedule or budget.	Detailed schedule and cost estimation, Software reuse.
3.	Developing wrong functions.	Making user survey, understanding the concepts, adopting prototyping techniques.
4.	Gold Plating i.e. adding features to the project.	Reviewing the requirements, prototyping.
5.	Developing wrong user interface.	Performing prototyping, task analysis

6.	Requirement changes occur frequently.	Incremental development.
7.	Shortfalls in externally done tasks.	Reference checking, auditing, prototyping.
8.	Shortfall in externally developed components.	Reference checking and inspections.
9.	Real-time performance shortfalls.	Simulation, modelling, prototyping.
10.	Straining computer knowledge capability.	Technical analysis, reference checking, prototyping.

- Apart from preparing a checklist, other methods of risk identification are **decision driver analysis, assumption analysis** and **decomposition.**
- In **decision driver analysis** technique, there is questioning and analyzing of all the major decisions taken for the project. If the decisions are driven by some non technical factors such as politics or marketing then the probability of occurrence of risk is very high.
- For identifying the risks assumptions about the project are compared with the pat experiences of the project.
- In **decomposition technique** the large project is broken down into small parts and analysis is made in order to identify the risks.
- The next task after risk identification is risk analysis and prioritization.

Risk Analysis

- Risk analysis is the process in which probability of occurrence of risk and the corresponding losses are estimated.
- If **cost models** are used for cost and schedule estimation, then the same models can be used to assess the cost and schedule risk. For example COCOMO model can be used for to analyse the cost and schedule risks.
- Risk analysis can be done by estimating the **worst-case value of size** and all the **cost drivers.** From these values the **project cost** can be estimated. This will give us the **worst-case analysis.** Then using the **worst-case effort estimate,** the **worst-case schedule** can easily be obtained.
- Another approach of risk analysis is making analysis of various things like -
  - Outcome of various decisions(Decision analysis)
  - Risks on various factors such as reliability and usability(quality factor analysis)
  - Performance constraints(performance analysis).

Risk Prioritization

- After risk analysis the impact of each risk on the project can be analysed. Based on this impact risks can be prioritised.
- **Risk exposure** computing is done for prioritising the risks. Risk exposure is also called as **risk impact.**
- The risk exposure can be calculated by following formula,

Risk Exposure = Probability of occurrence of risk *Loss due to unsatisfactory outcome

- Thus risk exposure for each risk from risk table is calculated. The total risk exposure of all risks helps in determining the final cost of the project.

4.6.5 Risk Containment

- Risk containment means reduction of risk.
- The project manager and team will be able to identify strategies to minimize or eliminate the risk factors.
- For example - If a project is facing high risk due to lack of experience in development platform, then the recruiter or hiring expert contractor can control this risk by hiring the skilled and experienced employee for the desired project.
- Lot of high risk factors can be eliminated or reduced during the risk assessment.

4.6.6 RMMM Strategy

RMMM stands for **risk mitigation**, **monitoring** and **management**. There are three issues in strategy for handling the risk is

1. Risk avoidance 2. Risk monitoring
3. Risk management.

Risk mitigation

Risk mitigation means preventing the risks to occur(risk avoidance). Following are the steps to be taken for mitigating the risks.

1. Communicate with the concerned staff to find of probable risk.
2. Find out and eliminate all those causes that can create risk before the project starts.
3. Develop a policy in an organization which will help to continue the project even though some staff leaves the organization.
4. Everybody in the project team should be acquainted with the current development activity.
5. Maintain the corresponding documents in timely manner. This documentation should be strictly as per the standards set by the organization.
6. Conduct timely reviews in order to speed up the work.
7. For conducting every critical activity during software development, provide the additional staff if required.

Risk monitoring

In risk monitoring process following things must be monitored by the project manager,

1. The approach or the behaviour of the team members as pressure of project varies.
2. The degree in which the team performs with the spirit of "team-work".
3. The type of co-operation among the team members.
4. The types of problems that are occurring.
5. Availability of jobs within and outside the organization.

The project manager should monitor certain mitigation steps. For example.

If the current development activity is monitored continuously then everybody in the team will get acquainted with current development activity.

The **objective** of risk monitoring is

1. To check whether the predicted risks really occur or not.
2. To ensure the steps defined to avoid the risk are applied properly or not.
3. To gather the information which can be useful for analyzing the risk.

Risk management

Project manager performs this task when risk becomes a reality. If project manager is successful in applying the project mitigation effectively then it becomes very much easy to manage the risks.

For example, consider a scenario that many people are leaving the organization then if sufficient additional staff is available, if current development activity is known to everybody in the team, if latest and systematic documentation is available then any 'new comer' can easily understand current development activity. This will ultimately help in continuing the work without any interval.

Board Questions

1. *What is risk projection ? Enlist steps of risk projection.* **MSBTE : Summer-15, Marks 4**
2. *Describe RMMM strategy in detail.* **MSBTE : Summer-15,Winter-15, Marks 4**
3. *What is software risk ? Explain types of software risks.* **MSBTE : Summer-16, Marks 4**
4. *Describe the following with respect to Risk assessment :*
   i) Risk identification ii) Risk analysis
   iii) Risk prioritization **MSBTE : Winter-16, Marks 8**
5. *Explain following terms w.r.t. risk management :*
   i) Risk identification ii) Risk analysis
   MSBTE : Summer-17, Marks 4
6. *Enlist and explain different types of Software Risks. (four points)*
   MSBTE : Summer-18, Marks 4

❑❑❑

Notes

UNIT- V

5 Software Quality Assurance and Security

5.1 Project Scheduling

MSBTE : Winter-15, 16, 17, Summer-15, 16, 17, Marks 4

Definition : Project scheduling is an activity the estimated efforts are distributed across the planned project duration by allocating effort to specific software engineering tasks.

In project the scheduling can be done using two simple activities -

1. Determining overall schedule by using different important milestones.
2. Developing detailed schedule for different tasks.

5.1.1 Basic Principle

Following are basic principles used in project scheduling -

(1) Compartmentalization : The project is partitioned or compartmentalized into number of activities, actions and tasks.

(2) Interdependency : The interdependency of each compartmentalized activity, task and action must be determined. Some tasks execute in sequence and some might execute in parallel.

(3) Time Allocation : For each scheduled task, some number of work units must be allocated. The start date and end date of each allocated task must be specified.

(4) Effort Validation : The number of people allocated for the scheduled tasks must be validated by the project manager.

(5) Defined Responsibilities : Every task that is scheduled should be assigned to specific team member.

(6) Defined Outcomes : Every scheduled task must have definite outcome.

(7) Defined Milestones : Every task or group of tasks should be associated with a project milestone.

5.1.2 Work Breakdown Structure

- Work breakdown Structure(WBS) is a process of dividing the complex projects to simpler and manageable tasks.
- In WBS, the large tasks are broken down into manageable chunks of work. These chunks can be easily examined and analysed.
- The project manager is responsible for creation of WBS.

Construction of Work Breakdown Structure

(1) The first step in creation of work breakdown structure is to identify the main deliverable of the project.

(2) Then the high level tasks are identified and they are broken down into smaller chunks of work.

(3) In the process of breaking down the tasks, one can break them down into different levels of detail. One can detail a high-level task into ten sub-tasks while another can detail the same high-level task into 20 sub-tasks. Thus there is no standard rule for breaking down of task into chunks. In general there is a simple rule is that - The smallest tasks of WBS is should not be smaller than two weeks worth of work.

Another rule is - no task should be smaller than 8 hours of work and should not be larger than 80 hours of work.

(4) There are many forms of displaying the work breakdown structure - it can be in tree like form, table or list form. For example -

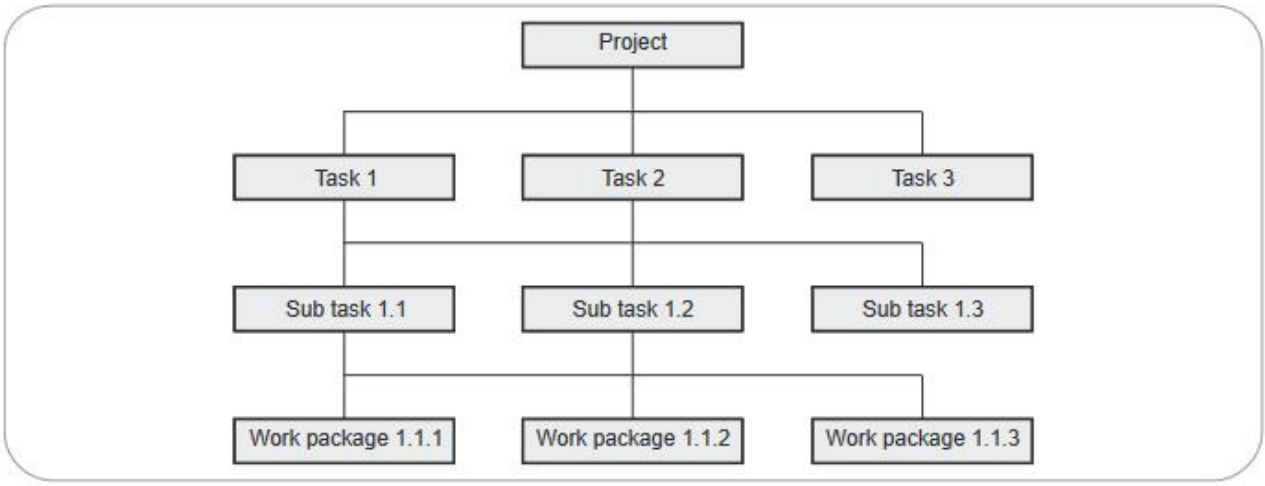


Fig. 5.1.1 Work breakdown structure in tree form

Example of WBS : The work breakdown structure for a simple product development is as shown below -

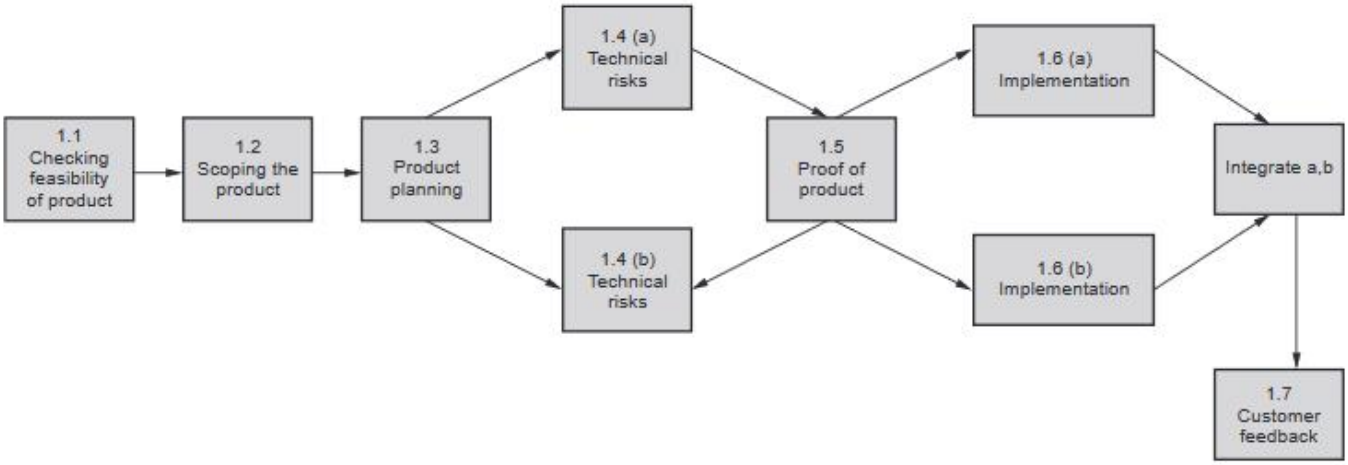


Fig. 5.1.2 Work breakdown structure in tree form

5.1.3 Activity Network and Critical Path Method

Bar charts and activity networks are graphical representation of project schedule. The beginning and end of the activity, responsible staff for corresponding activity is shown by bar chart.

Activity network show dependencies between different activities. There are some automating project management tools using which bar chart and activity network can be generated. These tools require database of project information as an input.

Ex. 5.1.1 *Consider, some hypothetical tasks, duration and dependencies. Draw the activity network and find the critical path.*

Task	Duration	Dependencies
T1	10	
T2	15	T1(M1)

T3	15	T1(M2)
T4	7	T1,T2(M3)
T5	20	T1(M1)
T6	5	T4,T5(M4)
T7	4	T6(M6)
T8	8	T7(M5)
T9	10	T6,T8(M7)
T10	12	T9(M8)

Sol. : The T1, T2, ... represent various tasks. The milestones of various tasks is shown by M1, M2, ... The activity network for these activities can be as shown below.

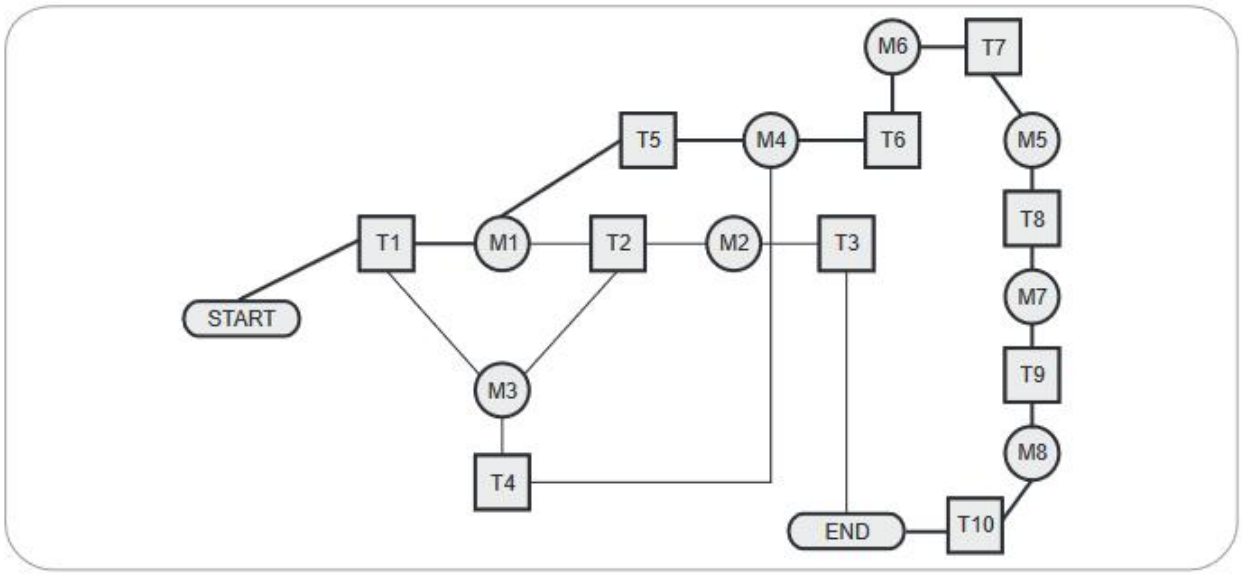


Fig. 5.1.3 Activity network

The minimum amount of time required by the project can be computing the longest possible path to reach from begin to end. This path is referred as the **critical path** in the project and the activities along this path are referred as the critical activities. In Fig. 5.1.3 critical path is shown by thick edge.

Ex. 5.1.2 *For a software project different activities and their durations are listed as below. Draw the activity network and find the critical path.*

Task	T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8	T_9	T_{10}	T_{11}	T_{12}
Duration (in days)	8	15	15	10	10	5	20	25	15	15	7	10
Dependencies	–	–	T_1	–	T_2, T_4	T_1, T_2	T_3	T_4	T_3, T_6	T_5, T_7	T_9	T_{11}

Sol. :

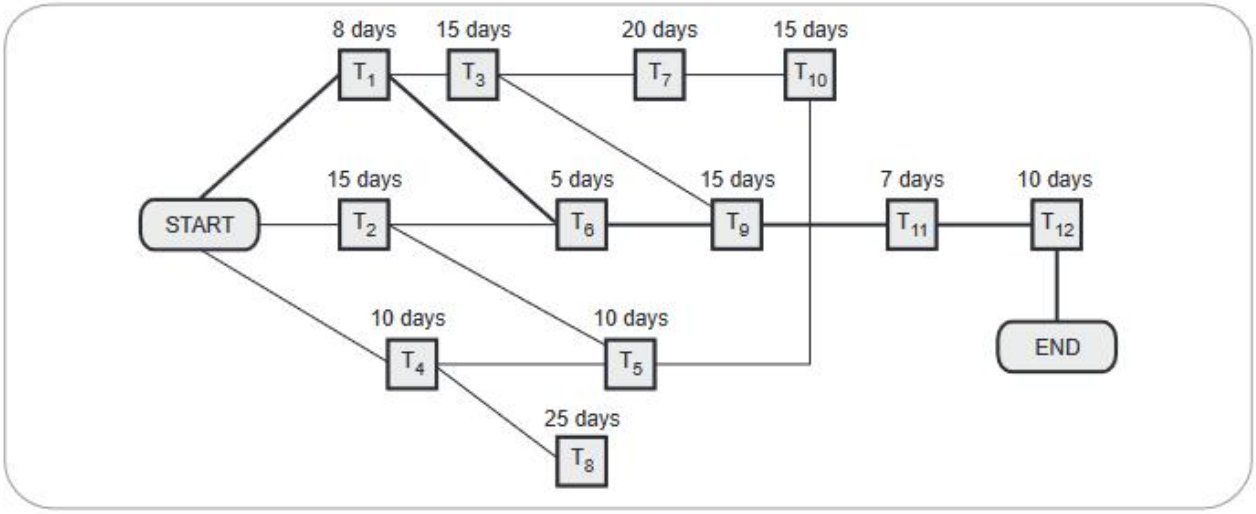


Fig. 5.1.4 Activity chart

A thick line denotes the critical path. A critical path is a longest possible path which starts from the beginning and reaches at the end and which requires minimum amount of time.

5.1.4 Scheduling Techniques

- There are two commonly used scheduling techniques - PERT and CPM.
- The PERT stands for Project management and Review Technique. This technique is used for the projects where time needed to complete different activities are not known.
- The CPM stands for Critical Path Method. This technique is used in conjunction with PERT and is used for managing the well defined activities of the project.
- The PERT and CPM technique is used for -
  - Prediction of deliverables.
  - Planning resource requirements
  - Controlling resource allocation
  - Internal and external program review
  - Performance evaluation
- Various framework activities in PERT/CPM are -
  - Define the project. The project should have single start activity and single finish activity.
  - Develop relationships among the activities.
  - Draw the network diagram for connecting all the activities.
  - Assign time and cost to each activity.
  - Compute critical path.
  - Use network to help plan, schedule, monitor and control the project.
- **Example -**
- The formula used in PERT to calculate number of working days is based on three point estimate.
- The three point estimates are optimistic, most likely and pessimistic estimates.
- **Formula** is

PERT weighted average

$$= \frac{\text{Optimistic time} + 4 \times \text{Most likely time} + \text{Pessimistic time}}{6}$$

- **Example :** Given that

Optimistic time = 6 days

Most likely time = 9 days

Pessimistic time = 12 days

then

$$\text{PERT weighted average} = \frac{6 + (4 \times 9) + 12}{6} = 9$$

- Thus it would be 9 days using PERT.
- The **main advantage** of PERT is that it attempts to address the risk associated with duration estimates.

Difference between PERT and CPM

Sr. No.	PERT	CPM
1.	PERT is a project management technique used to manage uncertain activities of project.	CPM is a statistical technique of project management that manages well defined activities.
2.	This technique of planning and control of time.	A method to control cost and time.
3.	This technique is event oriented.	This technique is activity oriented.
4.	It is non repetitive in nature.	It is repetitive in nature.

Board Questions

1. *List four basic principles of project scheduling.*
   MSBTE : Summer-15,16, Winter-15, Marks 4
2. *Differentiate between PERT and CPM.*
   MSBTE : Summer-15, Winter-16, 17, Marks 4
3. *With an example, explain how CPM and PERT are useful in software project management.*
   MSBTE : Winter-15, Marks 4
4. *What is the concept of task network ?*
   MSBTE : Summer-17, Marks 4
5. *Write meaning of PERT and CPM.*
   MSBTE : Summer-17, Marks 4
6. *What is project scheduling ?*
   MSBTE : Winter-17, Marks 4

5.2 Project Tracking

MSBTE : Winter-17, Summer-16, 18, Marks 8

- Project schedule is the most important factor for software project manager. It is the duty of project manager to decide the project schedule and track the schedule.
- Tracking the schedule means determine the tasks and milestones in the project as it proceeds.
- Following are the **various activities conducted during tracking** of the project schedule -

1. Conduct periodic meetings. In this meeting various problems related to the project get discussed. The progress of the project is reported to the project manager.
2. Evaluate results of all the project reviews.
3. Compare 'actual start date' and 'scheduled start date' of each of the project task.
4. Determine if the milestones of the project is achieved on scheduled date.
5. Meet informally the software practioners. This will help the project manager to solve many problems. This meeting will also be helpful for assessing the project progress.
6. Assess the progress of the project quantitatively.

- Thus for tracking the schedule of the project the project manager should be an experienced person. In fact project manager is the only responsible person who is controlling the software project.
- When some problems occur in the project then addition resources may be demanded, skilled and experienced staff may be employed or project schedule can be redefined.

5.2.1 Time Line Chart (Gantt Chart)

- In software project scheduling the timeline chart is created. The purpose of timeline chart is to emphasize the scope of individual task. Hence set of tasks are given as input to the time line chart.
- The time line chart is also called as Gant chart.
- The time line chart can be developed for entire project or it can be developed for individual functions.
- In time line chart

1) All the tasks are listed at the leftmost column.
2) The horizontal bars indicate the time required by the corresponding task.
3) When multiple horizontal bars occur at the same time on the calendar, then that means concurrency can be applied for performing the tasks.
4) The diamonds indicate the milestones.

- In most of the projects, after generation of time line chart the project tables are prepared. In project tables all the tasks are listed along with actual start and end dates and related information.

Example

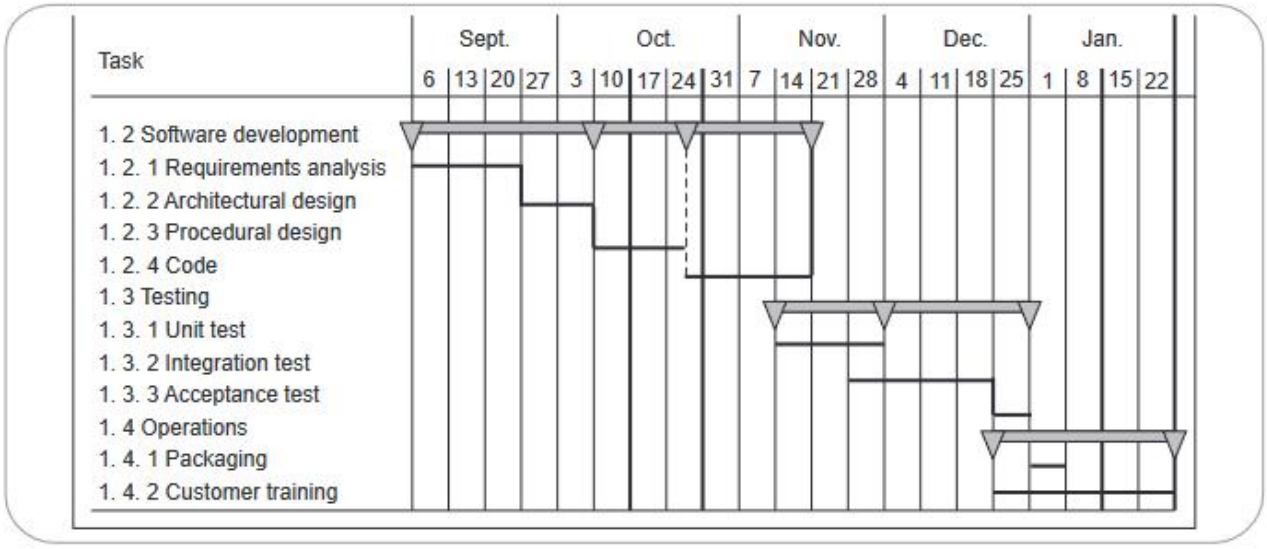


Fig. 5.2.1 Time line chart

Project table

Tasks	Planned start	Actual start	Planned end	Actual end	Effort assignment
Requirement analysis	6th Sept'05	6th Sept'05	20th Sept'05	22nd Sept'05	Jayashree, Padma, Lucky
Architectural design	27th Sept'05	27th Sept'05	3rd Oct'05	7th Oct'05	Trupti, Varsha
Procedural design	10th Oct'05	12th Oct'05	24th Oct'05	25th Oct'05	Varsha, Sachin, Devendra
⋮	⋮	⋮	⋮	⋮	⋮
Customer training	1st Jan'06	4th Jan'06	22nd Jan'06	25th Jan'06	Smita, Yogita

5.2.2 Earned Value Analysis

- The Earned Value Analysis (EVA) takes into consideration the project context for planned and actual expenditure.
- This analysis is made to find out project scope, schedule and resource characteristics.
- The EVA acts as a measure for software project progress.
- Various measures are determined during EVA. These measures are -

1) **Planned Value (PV) :** It denotes the planned cost of the work. The planned value is developed by first determining all of the work, that must be accomplished for successful project result.
2) **Actual Cost (AC) :** It represents the actual amount that the business has to expend on the project.
   $AC = \sum$ Efforts expended on work task that have been completed by time t.
3) **Earned Value (EV) :** It is project manager's estimate of the amount of originally budgeted work completed.
4) **Budget At Completion (BAC) :** It represents total budget for the project.
5) **Schedule Variance (SV) :** It indicates the status of the schedule. It represents whether work is ahead or behind the plan. If SV is negative the project is behind schedule, if it is positive then project is ahead of schedule. If SV is equal to zero, then project is on schedule.

6) **Cost Variance (CV) :** It is the difference between earned value and actual cost.

If CV = 0 then project is on budget.

If CV < 0 then project is over budget

If CV > 0 then project is under budget

7) **Schedule Performance Index (SPI) :** It is a measure of schedule efficiency on a project. If SPI = 1.0, project is on schedule.

If SPI is greater than 1 then project is ahead of the schedule.

IF SPI is less than 1 then project is getting delayed and it is behind the schedule.

8) **Cost Performance Index (CPI) :** It is a measure of cost efficiency on a project. The value 1.0 represents that project is within the given budget.

If CPI < 1.0 then that means project is over budgeted.

If CPI > 1.0 then that means project is under budgeted and we require most cost to accomplish the project.

Formula used during (EVA) :

PV = Planned Completion (%) * Budget At Completion (BAC)

EV = Actual Completion (%) * BAC

SV = EV – PV

CV = EV – AC

SPI = EV/PV

CPI = EV/AC

Ex. 5.2.1 *Mr. Koushan is the project manager on a project to build a new cricket stadium in Mumbai, India. After six months of work, the project is 27 % complete. At the start of the project, Koushan estimated that it would cost $ 50,000.000, What is the Earned value ?*

Sol. : The formula for Earned value is -

Earned value = % of work × Budget

= 27 % × 50000

$$= \frac{27}{100} \times 50000 = 13,\ 500$$

∴ The earned value is **$ 13,500**

Board Questions

1. *List different ways in which the project schedule can be tracked.* **MSBTE : Summer-16, Marks 4**
2. *What is project scheduling and tracking ? State four reasons why project deadlines cannot be met ?* **MSBTE : Summer-16, Marks 8**
3. *Explain the concept of Gantt chart.* **MSBTE : Winter-17, Marks 4**
4. *Explain different activities done to track the software.* **MSBTE : Summer-18, Marks 4**

5.3 Software Quality Management Vs. Software Quality Assurance

- **Definition of Quality :** The International Organization for Standardization (ISO) defines quality as the totality of characteristics of an entity that bear on its ability to satisfy stated or implied needs.
- The project quality management is a process which ensures that the project will satisfy the needs for which it was undertaken.
- The main principle of project quality management is to ensure the project will meet or exceed stakeholder's needs and expectations.
- The project team must develop a good relationship with key stakeholders.
- The project quality management is performed using following three key processes.

1. Planning Quality Management
2. Performing Quality Assurance
3. Controlling Quality

5.4 Phases of Software Quality Assurance

MSBTE : Winter-16, 17, Summer-15, 16, 17, 18, Marks 8

- **Definition of quality assurance :** It is planned and systematic pattern of activities necessary to provide a high degree of confidence in the quality of a product. It provides quality assessment of the quality control activities and determines the validity of the data or procedures for determining quality.
- The quality assurance consists of set of reporting and auditing functions.
- These functions are useful for assessing and controlling the effectiveness and completeness of quality control activities.
- The goal of quality assurance is to ensure the management of data which is important for product quality.

Statistical Quality Assurance

- Statistical software quality assurance is a simple concept which represents that changes in the software can be made in order to improve those elements of the process that introduce error.
- Statistical software quality assurance can be performed with the help of following steps -
  1. Collect the information about software defects. Categorize them.
  2. Make an attempt to trace each defect to its root cause.
  3. Isolate the vital few causes of the major source of all errors by using the 80-20 principle(known as Pareto principle). This principle is *"80 % of the defects can be traced to 20 % of all possible causes"*.
  4. Then move to correct the problems that have caused the defects

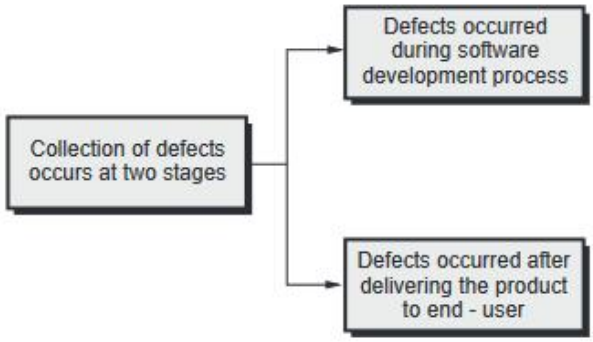


Fig. 5.4.1

Software Quality Assurance Activities

Let us list out the SQA activities conducted by SQA group.

1. Create a SQA plan.

A SQA plan is developed while planning the project. Quality assurance activities are conducted that are indicated in this plan. This plan basically

- Identifies evaluations to be performed.
- Audits and reviews to be performed, standards that should be adopted for the project.
- Procedures for error reporting and tracking.
- It also specifies documents to be produced by SQA group.
- Amount of feedback provided to the software project team.

2. Participates in description of software process.

The process selected by the software team is reviewed by the SQA group. This review is for

- Process description to ensure that it follows the organizational policy.

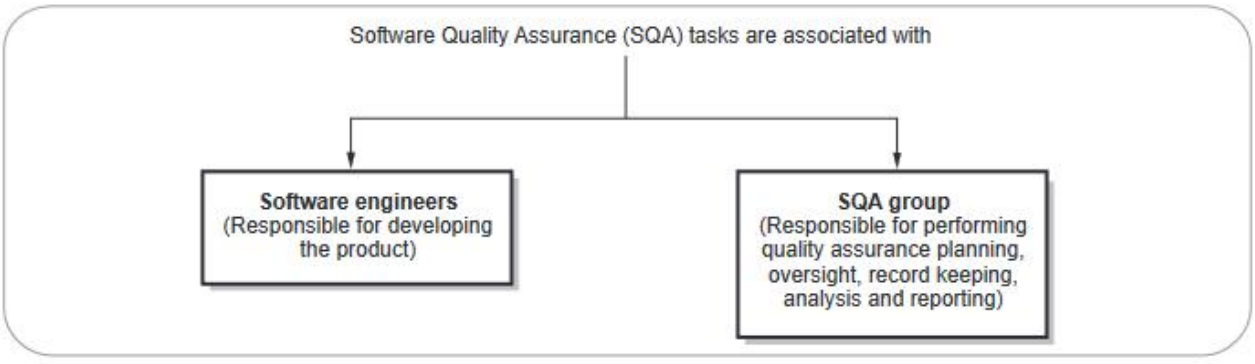


Fig. 5.4.2

- Internal software standards.
- Some standards that are adopted by the organization.

3. Reviews software engineering activities.

The SQA group identifies and documents the processes. The group also verifies the correctness of software process.

4. Authenticate designated software work products.

The SQA group performs following tasks -

- Reviews selected work product
- Identifies the process
- Documents them
- Tracks deviations
- Verifies the correctness made in the processes
- Regular reporting of results of its work to the project manager.

5. Ensure the deviations in software work. Document work products

The deviations in software work are identified from project plan. These processes are identified and handled according to documented procedure.

6. Identify any noncompliance and reports to senior management.

Non compliance items are identified and pursued until they get resolved. The periodic reporting about it is done to project manager.

Board Questions

1. *What steps are required to perform statistical SQA ?* MSBTE : Summer-15, Marks 4
2. *What is Software Quality Assurance ? What are the activities carried out in SQA.* MSBTE : Summer-16, Marks 4
3. *List various activities of SQA.* MSBTE : Summer-17, Marks 4
4. *What is Quality Assurance ? Describe various SQA activities.* MSBTE : Winter-16, Marks 8
5. *Explain about software quality assurance.* MSBTE : Winter-17, Marks 4
6. *Prepare any four software quality assurance guidelines and describe them.* MSBTE : Summer-18, Marks 4

5.5 Software Quality Control

MSBTE : Summer-17, Marks 4

- Quality control is a process in which activities are conducted in order to maintain the quality of product. These activities are series of inspections, reviews and tests used throughout the software process. These activities ensure whether each work product is satisfying the requirements imposed on it.
- While applying the quality control there should be a **feedback loop** to the process which generates the work product. With the help of such feedback we can tune the process if it does not satisfy the requirements. The feedback loop helps in minimizing the defects in the software product.
- The quality control activities can be fully automated or it can be completely manual or it can be a combination of automated tools and manual procedures.

Board Question

1. *What is quality control ?* MSBTE : Summer-17, Marks 4

5.6 Quality Evaluation Standards

MSBTE : Winter-15, 16, 17, Summer-15, 16, 17, 18, Marks 8

5.6.1 Six Sigma

Six sigma is widely used statistical software quality assurance strategy. It is a business driven approach to process improvement, reduced costs and increased profit. The word "six sigma" is derived from six standard deviations - 3.4 defects per million occurrences. Six Sigma originated at Motorola in the early 1980s.

There are three core steps in six sigma method -

Define - The customer requirements, project goals and deliverables are defined by communicating the customers.

Measure - The existing process and its output is measured in order to determine current quality performance.

Fig. 5.6.1 Six sigma framework

Analyze - In this phase defect metrics are analyzed in order to determine the few causes.

If an improvement is needed to an existing software then there are additional two methods in six sigma -

Improve - By eliminating the root causes of defects the process can be improved.

Control - The process can be controlled in such a way that the causes of defects can not be reintroduced.

These steps can sometimes be referred as DMAIC.

For a newly developing software, some organizations are suggesting following two alternating steps -

Design - In this step avoid root causes of defects and meet the customer requirements.

Verify - To verify the process, avoid defects and meet customer requirements.

These steps can sometimes be referred as DMADV.

5.6.2 ISO for Software

- In order to bring quality in the product and service, many organizations are adopting the *quality assurance system.*
- The *quality assurance systems* are the organizational structures that are used to bring quality in responsibilities, procedures, processes and resources.
- ISO 9000 is a family of *quality assurance system.* It can be applied to all types of organizations. It doesn't matter what size they are or what they do. It can help both product and service oriented organizations to achieve standards of quality.
- ISO 9000 is maintained by ISO, the **International Organization for Standardization** and is administered by accreditation and certification bodies.
- In ISO 9000, company's quality system and operations are scrutinized by third-party auditors for a complaince to the standard and effective operation. This process is called **registration to ISO 9000.**
- On successful registration, the company gets a certification from accreditation bodies of ISO. Such a company is then called **"ISO certified company".**
- **ISO 9001:2000** is a quality assurance standard which is applied to software engineering systems.
- It focuses on process flows, customer satisfaction, and the continual improvement of quality management systems.
- ISO 9001:2000 specifies requirements for a quality system that can be applied to any size or type of organization.
- The guideline steps for ISO 9001:2000 are
  - *Establish quality management system* - Identify and manage the processes in the quality management system.
  - *Document the quality management system*
  - *Support the quality*
  - *Satisfy the customers*
  - *Establish quality policy*
  - *Conduct quality planning*
  - *Control quality systems*
  - *Perform management reviews*
  - *Provide quality resources*
  - *Provide quality personnel*
  - *Provide quality infrastructure*
  - *Provide quality environment*
  - *Control realization planning*
  - *Control customer processes*
  - *Control product development*
  - *Control purchasing functions*
  - *Control operational activities*
  - *Control monitoring devices*
  - *Control non confirming products*
  - *Analyze quality information*

- *Make quality improvement*

- The ISO 9000 helps in creating organisational **quality manuals**. These quality manuals identify the organisational quality processes.
- Using these quality manuals, the project quality plan can be prepared for every individual project. Thus project quality management can be done.

This is illustrated by following Fig. 5.6.2.

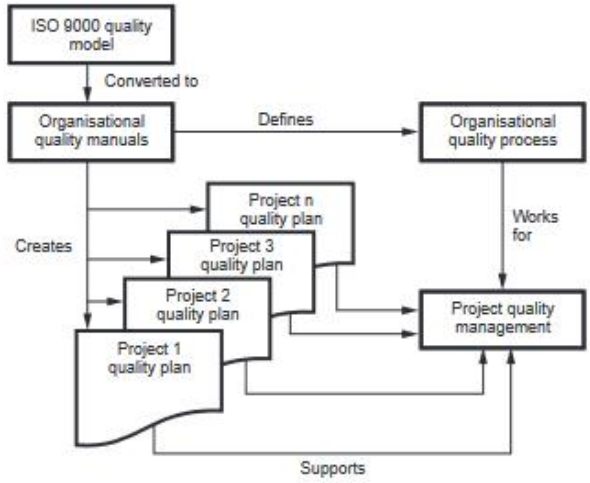


Fig. 5.6.2 ISO 9000 helps in quality management

5.6.3 CMMI

- The Software Engineering Institute (SEI) has developed a comprehensive **process meta-model** emphasizing process maturity. It is predicated on a set of system and software capabilities that should be present when organizations reach different levels of process capability and maturity.
- The Capability Maturity Model (CMM) is used in assessing how well an organization's processes allow to complete and manage new software projects.
- Various process maturity levels are

Level 1 : Initial - Few processes are defined and individual efforts are taken.

Level 2 : Repeatable - To track cost schedule and functionality basic project management processes are established. Depending on earlier successes of projects with similar applications necessary process discipline can be repeated.

Level 3 : Defined - The process is standardized, documented and followed. All the projects use documented and approved version of software process which is useful in developing and supporting software.

Level 4 : Managed - Both the software process and product are quantitatively understood and controlled using detailed measures.

Level 5 : Optimizing - Establish mechanisms to plan and implement change. Innovative ideas and technologies can be tested.

Thus CMM is used for improving the software project.

Comparison between ISO and SEI CMM Models

Sr. No.	ISO	CMM
1.	ISO 9001 addresses minimum criteria for an acceptable quality system.	In CMM, emphasis is on continuous process improvement.
2.	ISO 9001 focuses on hardware, software, processed material and services.	CMM focuses strictly on software.
3.	The basis of ISO 9001 is : "Say what you do and do what you say."	The basis of CMM is : "Say what you do and do what you say".
4.	Servicing activities are considered as separate maintenance process in ISO.	CMM does not have maintenance as a separate process.
5.	ISO 9001 maintains a quality record document which clearly shows whether or not required quality is achieved. This document also indicates whether or not existing quality system operates effectively.	The CMM emphasizes the need to record information for later use in the process and for improvement of the process.

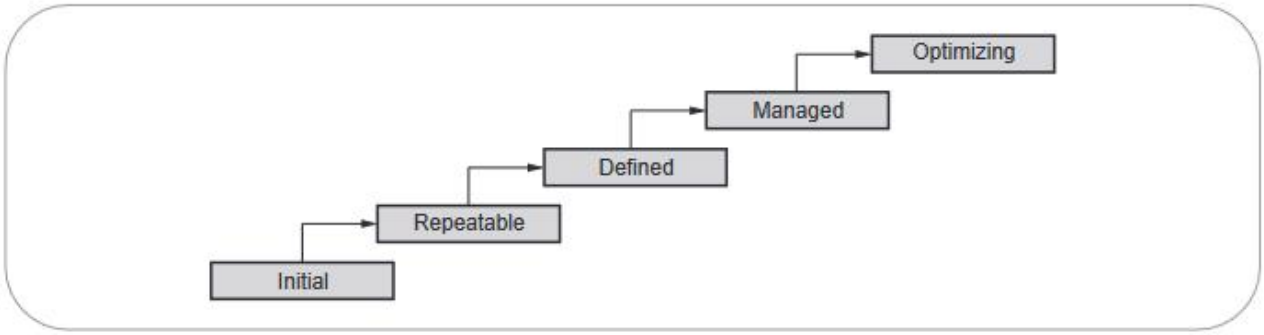


Fig. 5.6.3

Board Questions

1. *Describe six sigma for software engineering.* **MSBTE : Summer-15,18, Marks 4**
2. *Explain CMMI model with neat diagram.* **MSBTE : Summer-15,Marks 8**
3. *Give the outline that defines basic elements of ISO 9001 : 2000 for software quality assurance.* **MSBTE : Winter-15, Marks 4**
4. *What is six sigma ? Describes the core steps of DMAIC in detail.* **MSBTE : Winter-15, Marks 4**
5. *What is CMMI ? State two objectives of CMMI. Briefly explain the CMMI maturity levels.* **MSBTE : Summer-16, Marks 8**
6. *Explain CMMI with its levels and neat diagram.* **MSBTE : Winter-16, Marks 4**
7. *What is philosophy of six sigma ? Explain six sigma strategies.* **MSBTE : Summer-17, Marks 8**
8. *Describe six sigma for software engineering.* **MSBTE : Winter-17, Marks 4**
9. *State eight benefit of ISO standards.* **MSBTE : Winter-17, Marks 4**
10. *Explain different levels of Capability Maturity Model Integration technique. (CMMI)* **MSBTE : Summer-18, Marks 8**

5.7 Software Security

Software security is a concept which is implemented to protect software against malicious attack and other hacker risks so that the software continues to function correctly under such potential risks.

By providing software security - the integrity, authentication and availability is provided to the software system.

Basic Principles of Software Security

1. Protection from disclosure
2. Protection from alteration
3. Protection from destruction
4. Who is making the request
5. What rights and privileges does the requester have
6. Ability to build historical evidence
7. Management of configuration, sessions and errors/exceptions

5.8 Introduction to DEVOPs

Definition : Devops is a practice in which development and operation engineers participate together in entire lifecycle activities of system development from design, implementation to product support.

- The term Devops is derived from "Software **DEV**elopment " and "information technology **OP**eration**S**".
- Devops promotes a set of processes and methods from the three department **Development**, IT **operations** and **Quality assurance** that communicate and collaborate together for development of software system.

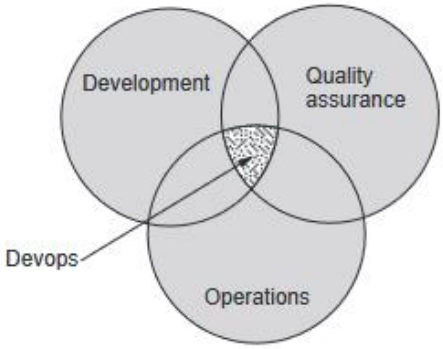


Fig. 5.8.1

Need for DEVOPs

- Devops enhances the organization's performance, improves the productivity and efficiency of development and operations teams.
- Bringing the two teams together centralizes the responsibility on the entire team and not specific individuals working.
- Devops is more than just a tool or a process change. It inherently requires an organizational culture shift.
- This cultural change is especially difficult, because of the conflicting nature of departmental roles :

1. Operations - seeks organizational stability;
2. Developers - seek change;
3. Testers - seek risk reduction.

- Adoption of Devops is driven by various factors. These factors are -

1. Demand for an increased rate of production releases - from application and business unit stakeholders.
2. Increased usage of data center automation and configuration management tools.
3. Use of agile and other development processes and methods.
4. Increased focus on test automation and continuous integration methods.
5. Wide availability of virtualized and cloud infrastructure.

Goals

The Goals of Devops are as follows -

1. To make simple processes increasing programmable and dynamic.
2. Fast delivery of product.
3. Lower failure rate of new releases.
4. Shortened lead time between fixes.
5. Faster mean time to recovery.
6. Increases net profit of organization.
7. To standardize development environment.
8. To reduce work in progress.
9. To reduce operating expenses.
10. To set up automated environment.

Benefits

Various benefits of Devops are -

- **Technical Benefits**
  1. Continuous software delivery is possible
  2. There is less complexity to manage the project.
- The problems in the project gets resolved faster.
- **Cultural benefits**
  1. The productivity of teams get increased.
  2. There is higher employee engagement.
  3. There arise greater professional development opportunities.
- **Business benefits**
  1. The faster delivery of the product is possible.
  2. The operating environment becomes stable.
  3. The communication and collaboration gets improved among the teams members and customer.
  4. More time is available for innovation rather than fixing and maintaining.

5.9 Secure Software Engineering

- Secure Software engineering is concerned with developing and maintaining software systems that behave reliably and efficiently by satisfying all the requirements of its customers.
- The secure software -
  (1) prevents loss of data.
  (2) prevents premature leaks of data.
  (3) Prevents downtime of resources.

Secure Software Engineering Life Cylce

- A Software Development Life Cycle (SDLC) is a framework that defines the process used by organizations to build an application. Over the years, multiple standard SDLC models have been proposed (Waterfall, Iterative, Agile, etc.) and used in various ways to fit individual circumstances.
- In general, SDLCs include the following phases:
  1. Planning and requirements.
  2. Architecture and design.
  3. Test planning.
  4. Coding.
  5. Testing and results.
  6. Release and maintenance.
- In the past, it was common practice to perform security-related activities only as part of testing. This technique usually resulted in a high number of issues discovered too late.
- It is better to integrate security concerning activities across the SDLC to help discover and reduce vulnerabilities early, effectively building security in.
- A Secure SDLC process ensures that security assurance activities such as penetration testing, code review, and architecture analysis are an integral part of the development effort.
- The primary advantages of secure software development Life cycle is -
  (1) More secure software as security is a continuous concern.
  (2) Awareness of security considerations by stakeholders.
  (3) Early detection of flaws in the system.
  (4) Cost reduction as a result of early detection and resolution of issues.
  (5) Overall reduction of business risks for the organization.

❑❑❑

SUMMER - 2015 **Software Engineering** Semester - IV (Civil) (21415)	**Solved Paper**

Time : 3 Hours] **[Total Marks : 100**

Note : 1) All questions are compulsory.

2) Answer each next main question on a new page.

3) Illustrate your answers with neat sketches wherever necessary.

4) Figures to the right indicate full marks.

5) Assume suitable data, if necessary.

Q.1 *Answer any five of the following :* **20**

a) *Explain software engineering as a layered technology approach.*

b) *Enlist core principles of software engineering practice.*

c) *Describe data objects and data attributes.*

d) *List four objectives of testing.*

e) *List four basic principles of project scheduling.*

f) *What steps are required to perform statistical SQA ?*

g) *State any four attributes of a good software.*

Q.2 *Answer any four of the following :* **16**

a) *Differentiate between waterfall model and incremental model.*

b) *What is SRS ?*

c) *Write importance of analysis modeling.*

d) *State eight characteristics of software bugs.*

e) *Enlist the features of SCM.*

f) *Describe six sigma for software engineering.*

Q.3 *Answer any four of the following :* **16**

a) *What do you mean by process framework ? Explain with suitable diagram.*

b) *Write four drawback of RAD model.*

c) *Explain deployment principle.*

d) *What are the characteristics of good design ?*

e) *Differentiate between validation and verification.*

f) *What is risk projection ? Enlist steps of risk projection.*

Q.4 *Answer any four of the following :* **16**

a) *Explain different decomposition techniques.*

b) *Describe integration testing.*

c) *What is DFD ? Explain level 1 DFD with example.*

d) *Explain cardinality and modality with example.*

e) *Explain spiral model with neat diagram.*

f) *Describe Agile process models in detail.*

Q.5 *Answer any two of the following :* **16**

a) *Describe eight principles of good planning.*

b) *With neat diagram explain translation of analysis model into design model.*

c) *Explain CMMI model with neat diagram.*

Q.6 *Answer any four of the following :* **16**

a) *Compare PSP and TSP.*

b) *List seven tusk of requirement engineering.*

c) *Differentiate between alpha and beta testing.*

d) *Compare white box and black box testing.*

e) *Describe RMMM strategy in detail.*

f) *Differentiate between PERT and CPM.*

WINTER - 2015

Software Engineering

Semester - IV (Civil) (15116)

Solved Paper

Time : 3 Hours] **[Total Marks : 100**

Note : 1) All questions are compulsory.

2) Illustrate your answers with neat sketches wherever necessary.

3) Figures to the right indicate full marks.

4) Assume suitable data, if necessary.

5) Preferably write the answers in sequential order.

Q.1 A) Attempt any THREE of the following : **12**

a) *Describe the characteristics of software.*

b) *Briefly explain software engineering as a layered technology.*

c) *With reference to requirement engineering, explain*
i) Inception and ii) Elicitation

d) *With reference to software design give the meanings of*
i) Modularity ii) Functional independence iii) Refactoring iv) Information hiding

b) Answer any ONE of the following : **6**

i) With a neat diagram, explain the nature and general steps of spiral model. Also give its advantages and disadvantages.*ii)Explain the various elements of analysis modeling in detail.*

Q.2 Answer any four of the following : **16**

a) *Define PSP and TSP. Give advantages of TSP.*

b) *Explain the features of Agile software development approach.*

c) *In which situation RAD model is applicable ? Give its advantages and disadvantages.*

d) *Describe the principles of deployment.*

e) *Explain PSPEC with an example.*

f) *Draw level 'O' and level 'I' DFD for library management system. Make suitable assumptions.*

Q.3 Attempt any FOUR of the following : **16**

a) *Explain the basic process framework activities.*

b) *Briefly describe the principles of communication.*

c) *Briefly describe the principles of coding.*

d) *With a neat diagram explain analysis model.*

e) *Explain domain analysis with a neat diagram.*

f) *Explain unit testing.*

Q.4 A) *Attempt any THREE of the following : (12)*

a) *Differentiate between alpha-testing and beta-testing.*

b) *Describe the following debugging strategies :*
i) Brute force ii) Back tracking

c) *With an example, explain how CPM and PERT are useful in software project management.*

d) *Give the outline that defines basic elements of ISO 9001 : 2000 for software quality assurance.*

B) Answer any ONE : **6**

a) *Explain the basic principles of project scheduling.*

b) *Explain the McCall's Quality factors.*

Q.5 Answer any TWO of the following : **16**

a) *Explain the core principles of software engineering in detail.*

b) *Explain in detail RMMM strategy.*

c) *What is six sigma ? Describes the core steps of DMAIC in detail.*

Q.6 Answer any FOUR of the following : **16**

a) *Explain white box testing.*

b) *Explain Top-Down integration testing.*

c) *Describe the attributes of a good test.*

d) *Why do the software projects fail ? Give reasons.*

e) *Describe the four elements of software configuration management system.*

SUMMER - 2016 **Software Engineering** **Semester - IV (Civil) (15162)**	**Solved Paper**

Time : 3 Hours] **[Total Marks : 100**

Note : 1) All questions are compulsory.

2) Answer each next main question on a new page.

3) Illustrate your answers with neat sketches wherever necessary.

4) Figures to the right indicate full marks.

5) Assume suitable data, if necessary.

Q.1 A) Attempt any three of the following : **12**

a) *Define software. State three characteristics of software.*

b) *What is software coding ? State three principles of code validation.*

c) *Describe the terms : Analysis Modeling and Design Modeling.*

d) *Differentiate between Prescriptive Process Modell and Agile Process Model (any four points).*

Q. B) Attempt any one of the following : **6**

a) *Describe the layered technology approach of Software Engineering.*

b) *Draw a dataflow diagram level 0 and level 1 for a Book Publishing House.*

Q.2 Attempt any four of the following : **16**

a) *Define the terms software process, software product, software work product and software engineering.*

b) *What is SRS ? Explain importance of SRS.*

c) *What is domain analysis ? Explain with suitable examples.*

d) *Describe the relationship between systems engineering and software engineering.*

e) *Draw a use case diagram for a Bank Management System.*

f) *What is Waterfall Model ? State the practical situations in which it can be used.*

Q.3 Attempt any four of the following : **16**

a) *State and explain any four types of software.*

b) *What is Requirements Elicitation ? What are the problems faced in eliciting requirements ?*

c) *Explain the importance of SRS.*

d) *What is Data Modeling ? Explain the terms cardinality and modality.*

e) *Draw a use case diagram for music system.*

Q.4 A) Attempt any three of the following : **16**

a) *What aspects of the software are tested in Unit Testing ?*

b) *State any four basic principles to be followed for project scheduling.*

c) *Define the terms : Software Reliability and Software Availabiliy.*

d) *Compare Alpha Testing and Beta Testing.*

Q.4 B) Attempt any one of the following : **6**

a) *What are the activities involved in SCM ?*

b) *What is Software Quality Assurance ? What are the activities carried out in SQA.*

Q.5 Attempt any two of the following : **16**

a) *What is Software deployment ? State the principles to be followed while preparing to deliver the software increment.*

b) *What is project scheduling and tracking ? State four reasons why project deadlines cannot be met ?*

c) *What is CMMI ? State two objectives of CMMI. Briefly explain the CMMI maturity levels.*

Q.6 Attempt any four of the following : **16**

a) *Compare top - down and bottom - up approach used for integrating testing.*

b) *Describe different debugging strategies.*

c) *What is software risk ? Explain types of software risks.*

d) *List different ways in which the project schedule can be tracked.*

e) *Compare software verification and software validation.*

WINTER - 2016

Software Engineering

Semester - IV (Civil) (16117)

Solved Paper

Time : 3 Hours] **[Total Marks : 100**

Note : 1) All questions are compulsory.

2) Answer each next main question on a new page.

3) Illustrate your answers with neat sketches wherever necessary.

4) Figures to the right indicate full marks.

5) Assume suitable data, if necessary.

6) Mobile Phone, Pager and any other Electronic Communication devices are not permissible in Examination Hall.

Q.1 A) Answer any three of the following : **12**

a) *Describe any four categories of software.*

b) *State and describe six principles of communication practices.*

c) *With neat diagram, describe inputs and output of domain analysis.*

d) *Write any four features of Agile Software Development approach.*

B) Answer any one of the following : 6

a) *With neat diagram, explain RAD model with its advantages and disadvantages. (2 each)*

b) *Draw DFD level 0 and level 1 for Hotel management system.*

Q.2 Answer any four of the following : 16

a) *Explain process framework with suitable diagram.*

b) *Describe in brief four level testing process in test execution.*

c) *Differentiate between cardinality and modality. (Any four points)*

d) *Differentiate between waterfall and incremental model. (any four points)*

e) *Explain following with reference to design concepts in design modeling.i) Abstractionii) Functional independence*

f) *Explain software engineering as a layered technology approach with neat diagram.*

Q.3 Answer any four of the following : 16

a) *Describe Team Software Process (TSP) model in detail.*

b) *Describe dour principles of analysis modeling.*

c) *Explain general format of Software Requirement Specification (SRS).*

d) *Describe Data Dictionary. Write any four advantages.*

e) *Explain :i) Component - level design elementsii) Deployment level design elementswith respect to design model.*

Q.4 A) Answer any three of the following : 12

a) *What do you mean by testing strategies ?*

b) *Explain basic principles of project planning.*

c) *Write steps to perform statistical SQA.*

d) *Explain smoke testing with its advantages and disadvantages (2 each).*

Q.4 B) Answer any one of the following : 4

a) *Explain CPM. How is it different from PERT ?*

b) *Explain CMMI with its levels and neat diagram.*

Q.5 Answer any two of the following : 16

a) *State and describe various core principles of software engineering.*

b) *Describe the following with respect to Risk assessment :*
i) Risk identification ii) Risk analysis iii) Risk prioritization

c) *What is Quality Assurance ? Describe various SQA activities.*

Q.6 Answer any four of the following : 16

a) *Differentiate between verification and validation.*

b) *Explain Brute force approaches used in debugging strategies.*

c) *Explain any four features of SCM.*

d) *Explain the following management spectrum :*
i) The Process ii) The Project

e) *Describe white box and black box testing of software.*

SUMMER - 2017 **Software Engineering** **Semester - IV (Civil) (16172)**	**Solved Paper**

Time : 3 Hours] **[Total Marks : 100**

Note : 1) All questions are compulsory.
2) Answer each next main question on a new page.
3) Illustrate your answers with neat sketches wherever necessary.
4) Figures to the right indicate full marks.
5) Assume suitable data, if necessary.
6) Mobile Phone, Pager and any other Electronic Communication devices are not permissible in Examination Hall.

Q.1 Attempt any five of the following : **20**

a) *What is software ? What is embedded software ?*

b) *Explain the term scrum.*

c) *List core principle of Software Engineering.*

d) *Write importance of analysis modeling.*

e) *List various testing characteristics.*

f) *What is change control ?.*

g) *List various activities of SQA.*

Q.2 Attempt any four of the following : **16**

a) *What is quality control ?*

b) *Explain following terms w.r.t. risk management :*
i) Risk identification ii) Risk analysis

c) *Describe debugging process.*

d) *Compare cardinality and modality.*

e) *Explain essence of practice.*

f) *What do you mean by process framework ? Explain with suitable diagram.*

Q.3 Attempt any four of the following : **16**

a) *Explain software engineering as a layered approach.*

b) *Explain following requirements engineering tasks :*
i) Negotiation ii) Specification

c) *What is DFD ? Explain its symbol.*

d) *How can project scheduling affect integration testing ?*

e) *What is the concept of task network ?*

f) *Explain SCM in short.*

Q.4 Attempt any four of the following : **16**

a) *Give possible reasons of why software is delivered late.*

b) *Explain test case design in detail.*

c) *Explain scenario based modeling in detail.*

d) *What is meant by software deployment ?*

e) *What is SRS ?*

f) *What is agile process ?*

Q.5 Attempt any two of the following : **16**

a) *Explain RAD model with its advantages and disadvantages.*

b) *Describe in detail eight principles of good planning.*

c) *What is philosophy of six sigma ? Explain six sigma strategies.*

Q.6 Attempt any four of the following : **16**

a) *Write meaning of PERT and CPM.*

b) *Explain the steps of bottom up integration.*

c) *List the objective of black box testing.*

d) *For library management system draw level 0 and level 1 DFD.*

e) *What are the characteristics of good design ?*

f) *State advantages of PSP and TSP.*

WINTER - 2017
Software Engineering
Semester - IV (Civil) (11718)

Solved Paper

Time : 3 Hours] **[Total Marks : 100**

Note : 1) All questions are compulsory.

2) Answer each next main question on a new page.

3) Illustrate your answers with neat sketches wherever necessary.

4) Figures to the right indicate full marks.

5) Assume suitable data, if necessary.

Q.1 Answer any five of the following : **20**

a) *Explain changing nature of software.*

b) *What are communication principles ? Explain their meaning.*

c) *List four objectives of testing.*

d) *Explain briefly unit testing.*

e) *What is alpha - beta testing ?*

f) *Describe six sigma for software engineering.*

g) *Explain analysis modeling.*

Q.2 Answer any four of the following : **16**

a) *Explain the waterfall model.*

b) *Explain modeling practice in software engineering with principles.*

c) *What do you mean by good test ?*

d) *Describe integration testing approach.*

e) *Explain Mccalls quality factor.*

f) *What is an object oriented analysis ?*

Q.3 Answer any four of the following : **16**

a) *Difference between prescriptive and agile process model.*

b) *Describe any two core principles of software engineering.*

c) *What is test plan ?*

d) *Describe regression testing.*

e) *Explain modality with the help of example.*

f) *What is SPM ? Why is it needed ?*

Q.4 Answer any four of the following : **16**

a) *Explain the concept of software requirement specification.*

b) *Explain characteristics of software testing.*

c) *State eight benefit of ISO standards.*

d) *Explain DFD with example.*

e) *Explain the concept of Gantt chart.*

f) *Explain CPM. How is it different from pert ?*

Q.5 Answer any two of the following : **16**

a) *What is software ? What are its characteristics ?*

b) *What are major task of requirement engineering ?*

c) *Explain the term debugging. Explain different debugging.*

Q.6 Answer any four of the following : **16**

a) *Explain Deployment principle .*

b) *Differentiate between validation and verification.*

c) *Explain about software quality assurance.*

d) *Describe behavioral model.*

e) *What is project scheduling ?*

f) *Explain SCM.*

SUMMER - 2018 **Software Engineering** **Semester - IV (Civil) (21718)**	**Solved Paper**

Time : 3 Hours] **[Total Marks : 100**

Note : 1) All questions are compulsory.

2) Answer each next main question on a new page.

3) Illustrate your answers with neat sketches wherever necessary.

4) Figures to the right indicate full marks.

5) Assume suitable data, if necessary.

6) Mobile Phone, Pager and any other Electronic Communication devices are not permissible in Examination Hall.

Q.1 a) Attempt any three of the following : **12**

i) *Define management spectrum and enlist characteristics of software.*

ii) *Draw stub and driver mechanism of unit testing and enlist various types of errors detected by unit testing.*

iii) *Describe five steps for successfulness of project.*

iv) *Draw the neat labeled diagram of spiral model and list two disadvantages of spiral model.*

Q.1 b) Attempt any one of the following : **6**

i) *Elaborate any six types of software considering the changing nature.*

ii) *Draw and explain level 0 and level 1 Dataflow diagram for "Online examination Win17 of form filling on MSBTE website".*

Q.2 Attempt any four of the following : **16**

a) *Elaborate the software characteristic "Software does not wear out".*

b) *List and explain three principles of analysis modeling.*

c) *Draw the usecase diagram for taking "photocopy of ansbooks from msbte" website.*

d) *Enlist advantages and disadvantages of smoke testing. (four points)*

e) *Enlist and explain different types of Software Risks. (four points)*

f) *Explain qualities of software considering :*
i) Quality of design ii) Quality of conformance

Q.3 Attempt any four of the following : **16**

a) *Give difference between waterfall model and incremental model. (four points)*

b) *Explain following requirements of engineering tasks :*
i) Negotiation ii) Validation

c) *Explain Architectural Design Elements.*

d) *State eight characteristics of software bugs.*

e) *Explain the functions of Software Configuration Management repository. (SCM)*

Q.4 a) Attempt any three of the following : **12**

i) *Explain the testing concept with its Testing Principles. (any four principles)*

ii) *Compare Bottom - up integration testing and top - down integration software testing. (four points)*

iii) *Explain the factors that Delay Project Schedule.*

iv) *Explain the six sigma for software engineering.*

b) Attempt any one of the following : **6**

i) *List and explain five framework activities defined in PSP (Personal software process).*

ii) *Explain 4 P's software project spectrum.*

Q.5 Attempt any two of the following : **16**

a) *Draw the behavioral analysis model for small hospital management system and illustrate the working of it.*

b) *List and explain the elements of analysis model with neat labeled diagram.*

c) *Explain different levels of Capability Maturity Model Integration technique. (CMMI)*

Q.6 Attempt any four of the following : **16**

a) *Explain principles of planning practices in software engineering (any four)*

b) *Explain input and output of domain analysis.*

c) *Define white box testing and black box testing with its need and characteristics. (two points)*

d) *Explain different activities done to track the software.*

e) *Prepare any four software quality assurance guidelines and describe them.*

❑❑❑

Solved Sample Test Paper - I
Software Engineering
S.Y. Diploma Semester - IV (Computer Engineering Group & Information Technology) (CO/CM/IF/CW)

Time : 1 Hour] **[Total Marks : 20**

Instructions :

(1) All questions are compulsory.

(2) Illustrate your answers with neat sketches wherever necessary.

(3) Figures to the right indicate full marks.

(4) Assume suitable data if necessary.

(5) Preferably, write the answers.

Q.1 ***Attempt Any Four.*** **[8]**

(a) *What is application software ?* **(Refer section 1.4(2))**

(b) *Give any two advantages of RAD model.* **(Refer section 1.7.1.2)**

(c) *Explain the concept of agile process in brief.* **(Refer section 1.8)**

(d) *What is the meaning of the term software elicitation ?* **(Refer section 2.8.2)**

(e) *What is use case ?* **(Refer section 2.11)**

(f) *What are the elements of analysis model ?* **(Refer section 3.2)**

Q.2 ***Attempt any THREE.*** **[12]**

(a) *List and explain any four attributes of a good software.* **(Refer section 1.1)**

(b) *Explain waterfall model in detail.* **(Refer section 1.7.1.1)**

(c) *Explain functional and non functional requirements.* **(Refer section 2.9)**

(d) *What is SRS ? Explain its importance.* **(Refer section 2.13)**

(e) *Explain the concept of cohesion and coupling in software design.* **(Refer section 3.3.1(7))**

(f) *Give any four rules used during data flow design.* **(Refer section 3.4.1.1)**

Solved Sample Test Paper - II
Software Engineering
S.Y. Diploma Semester - IV (Computer Engineering Group & Information Technology) (CO/CM/IF/CW)

Time : 1 Hour] **[Total Marks : 20**

Q.1 ***Attempt Any Four.*** **[8]**

(a) *Enlist four P's in management spectrum.* **(Refer section 4.1)**

(b) *Give any two causes of software project failure.* **(Refer section 4.1.4)**

(c) *Explain the term - functional point.* **(Refer section 4.2.2)**

(d) *What are three commonly used cost estimation approaches used in software projects ?* **(Refer section 4.3)**

(e) *Explain - work break down structure.* **(Refer section 5.1.2)**

(f) *Give the full forms of PERT and CPM.* **(Refer section 5.1.4)**

Q.2 ***Attempt any THREE.*** **[12]**

(a) *Explain LOC based estimation method.* **(Refer section 4.2.1)**

(b) *Enlist and explain different types of software risks (Four Points).* **(Refer section 4.6.1)**

(c) *Explain the term risk mitigation.* **(Refer section 4.6.6)**

(d) *List four basic principles of project scheduling.* **(Refer section 5.1.4)**

(e) *Write short note on - Earned Value Analysis.* **(Refer section 5.2.2)**

(f) *What are steps required to perform statistical SQA ?* **(Refer section 5.4)**

❑❑❑

Solved Sample Question Paper
Software Engineering

S.Y. Diploma Semester - IV (Computer Engineering Group & Information Technology) (CO/CM/IF/CW)

Time : 3 Hours] **[Total Marks : 70**

Instructions :

(1) All questions are compulsory.

(2) Illustrate your answers with neat sketches wherever necessary.

(3) Figures to the right indicate full marks.

(4) Assume suitable data if necessary.

(5) Preferably, write the answers in sequential order.

Q.1 ***Attempt Any FIVE of the Following :*** **[10]**

(a) *Define the terms - Software and Software engineering* **(Refer section 1.1)**

(b) *State and explain any two types of software* **(Refer section 1.4)**

(c) *List core principles of software engineering* **(Refer section 2.2)**

(d) *What is data object?* **(Refer section 3.1.1.1)**

(e) *What are two types of size estimation* **(Refer section 4.2)**

(f) *Define the term- Risk* **(Refer section 4.6)**

(g) *What is software quality assurance* **(Refer section 5.4)**

Q.2 ***Attempt Any THREE of the Following :*** **[12]**

(a) *Explain software engineering as a layered technology approach.* **(Refer section 1.2)**

(b) *Explain the essence of software engineering practice* **(Refer section 2.1)**

(c) *Explain cardinality and modality with example.* **(Refer section 3.1.2)**

(d) *Prepare any four software quality assurance guidelines and describe them.* **(Refer section 5.4)**

Q.3 ***Attempt Any THREE of the Following :*** **[12]**

(a) *Give the difference between size oriented metrics and function oriented metrics.* **(Refer section 4.2.2)**

(b) *Explain following with reference to design concepts in design modeling.*
i) Abstraction ii) Functional independence **(Refer section 3.3.1)**

(c) *Explain deployment principle* **(Refer section 2.7)**

(d) *Enlist basic principles of software security* **(Refer section 5.7)**

Q.4 ***Attempt Any THREE of the Following :*** **[12]**

(a) *Enlist various functional and non functional requirements for the bank ATM system* **(Refer example 2.9.1)**

(b) *Elaborate the software characteristic: "Software is engineered, not manufactured "* **(Refer section 1.3)**

(c) *Explain 4 P's software project spectrum* **(Refer section 4.1)**

(d) *Write short note on - Unit testing* **(Refer section 3.5.3.1)**

Q.5 ***Attempt any TWO of the Following :*** **[12]**

(a) *Explain COCOMO estimation model in detail* **(Refer section 4.4)**

(b) *Draw DFD level 0 and level 1 for Hotel Management System* **(Refer example 3.4.2)**

(c) *Describe six sigma for software engineering* **(Refer section 5.6.1)**

Q.6 ***Attempt any TWO of the Following :*** **[12]**

(a) *What is DEVOPs ? Explain its needs and benefits of DEVOPs* **(Refer section 5.8)**

(b) *Explain Risk management procedure in detail* **(Refer section 4.6)**

(c) *What are major tasks of requirement engineering?* **(Refer section 2.8)**

❑❑❑

SUMMER - 2019

Software Engineering

S.Y. Diploma Semester - IV (Computer Engg. Group & Information Technology) (CO/CM/IF/CW) (21819)

Solved Paper
I - Scheme

Time : 3 Hours] **[Total Marks : 70**

Instructions :

1) All questions are compulsory.
2) Answer each next main question on a new page.
3) Illustrate your answers with neat sketches wherever necessary.
4) Assume suitable data if necessary.
5) Use of Non-programmable Electronic Pocket Calculator is permissible.
6) Mobile phone, pager and any other electronic communication devices are not permissible in Examination Hall.
7) Preferably, write the answers in sequential order.

Q.1 ***Attempt any Five of the following :*** **[10]**

a) *Enlist and explain software characteristics (any two)* **(Refer section 1.3)**

b) *Define software on engineering.* **(Refer section 1.1)**

c) *State need of Software Requirement Specification (SRS).* **(Refer section 2.14)**

d) *Define Reactive Risk strategies.*

Ans. : Reactive Risk Strategy is a strategy in which certain action occurs in response to the risk.

e) *Specify following cost directives of cocomo :* **(Refer section 4.4)**

i) Product attributes (any two) *ii) Hardware attributes (any two).*

f) *Differentiate between software quality management and software quality assurance (any two points).* **(Refer section 5.3)**

g) *Define software quality assurance.* **(Refer section 5.4)**

Q.2 ***Attempt any Three of the following :*** **[12]**

a) *Explain software engineering as layered technology approach.* **(Refer section 1.2)**

b) *Explain with example decision table.* **(Refer section 3.4.3)**

c) *Explain following elements of management spectrum :* **(Refer section 4.1)**

i) People *ii) Process* *iii) Product* *iv) Project*

d) *List and explain basic principles of project scheduling.* **(Refer section 5.1.1)**

Q.3 ***Attempt any Three of the following :*** **[12]**

a) *Distinguish between perspective process model and agile process model.* **(Refer section 1.8)**

b) *Describe any four principles of communication for software engineering.* **(Refer section 2.3)**

c) *Draw proper labeled "LEVEL I Data Flow Diagram" (DFD) for student attendance system.*

Ans. :

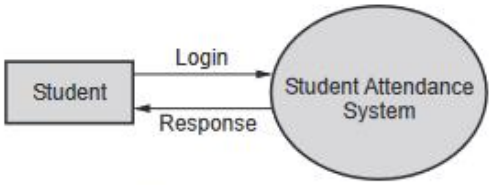


Fig. 1 Level 0 DFD

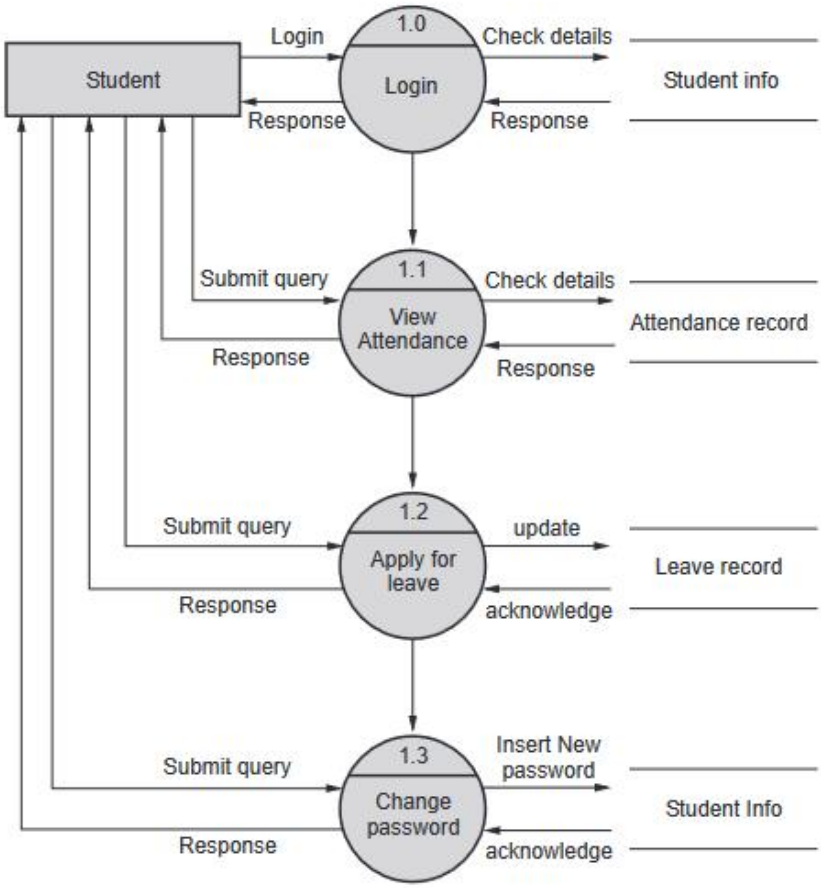


Fig. 2 Level 1 DFD

d) *State importance of "Function Point (FP)" and "Lines of Codes (LOC)" in concerned with project estimation.* **(Refer section 4.2)**

Q.4 ***Attempt any Three of the following :*** **[12]**

a) *Describe extreme programming with proper diagram.* **(Refer section 1.8.2)**

b) *List and explain any "four core principles" of software engineering.* **(Refer section 2.2)**

c) *Explain RMMM plan with example.* **(Refer section 4.6)**

d) *Explain any one project cost estimation approach.* **(Refer section 4.3)**

e) *Prepare time line chart for Library Managements System (five days a week) Consider phases of SDLC.* **(Refer section 5.2.1)**

Q.5 ***Attempt any Two of the following :*** **[12]**

a) *Enlist Requirement Gathering and Analysis for web based project for registering candidates for contest (any six points).*

Ans. :

(1) The candidate must enter personal information for user name, and password, email_id.

(2) The system should validate his/her email_id.

(3) The system should sent the system generated user name and password to the user through his email.

(4) The system should sent the "Successful registration" message to the user. If the registration gets failed the system should send the appropriate message to the user and it should consider such candidate as invalid.

(5) The candidate must be able to log-in with the user name and password.

(6) If the user forgets the password, the system should provide the assistance to the candidate with the help of his/her authorized email-id

b) *Differentiate between White box and Black box testing (any six points).* **(Refer section 3.5.2)**

c) *Describe Co-Como II model for evaluating size of software project with any three parameters in detail.* **(Refer section 4.6)**

Q.6 ***Attempt any Two of the following :*** **[12]**

a) *Draw and explain transition diagram from requirement model to design model.* **(Refer section 5.3.1)**

b) *Describe CMMI. Give significance of each level.* **(Refer section 5.6.3)**

c) *Identify and enlist requirement for given modules of employee management software :*

i) Employee detail ii) Employee salary iii) Employee performance

Ans. : (i) Employee details

1. The employee details fields must be completely filled up by the employee.
2. The employee information must be validated and then the information is entered in the database.

(ii) Employee salary

1. The appropriate salary information must be entered in the employee database of employee management software.
2. The salary must be non-negative and non-zero.

(iii) Employee performance

1. Employee performance record must be updated periodically in the employee management software.
2. On the lowering the performance, the employee should get the system generated notification.

❑❑❑

Notes