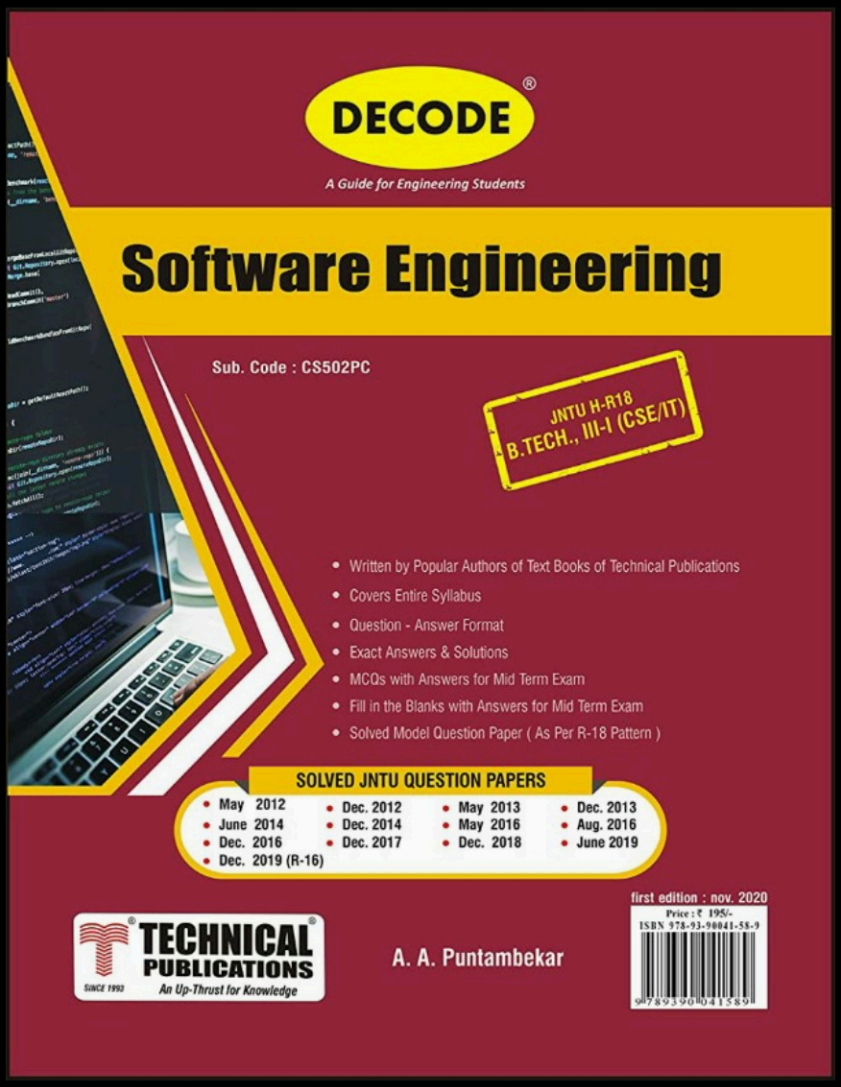

DECODE
A Guide for Engineering Students
Software Engineering
Sub. Code : CS502PC
JNTU H-R18
B.TECH., III-I (CSE/IT)
Written by Popular Authors of Text Books of Technical Publications
Covers Entire Syllabus
Question - Answer Format
Exact Answers & Solutions
MCQs with Answers for Mid Term Exam
Fill in the Blanks with Answers for Mid Term Exam
Solved Model Question Paper ( As Per R-18 Pattern )
SOLVED JNTU QUESTION PAPERS
May 2012
Dec. 2012
May 2013
Dec. 2013
June 2014
Dec. 2014
May 2016
Aug. 2016
Dec. 2016
Dec. 2017
Dec. 2018
June 2019
Dec. 2019 (R-16)
TECHNICAL PUBLICATIONS
SINCE 1993
An Up-Thrust for Knowledge
A. A. Puntambekar
first edition : nov. 2020
Price : ₹ 195/-
ISBN 978-93-90041-58-9

SUBJECT CODE : **CS502PC**

# JNTUH - R18

B.Tech., III-I (CSE / IT)

# SOFTWARE ENGINEERING


Mrs. Anuradha A. Puntambekar
M.E. (Computer)
Formerly Assistant Professor in
P.E.S Modern College of Engineering, Pune


## FEATURES

- ☑ Written by Popular Authors of Text Books of Technical Publications
- ☑ Covers Entire Syllabus ☑ Question - Answer Format ☑ Exact Answers & Solutions
- ☑ Fill in the Blanks with Answers for Mid Term Exam
- ☑ MCQs with Answers for Mid Term Exam ☑ Solved Model Question Paper As Per R-18 Pattern

**SOLVED JNTU QUESTION PAPERS**

| | | | |
|---|---|---|---|
| ❍ May - 2012 | ❍ Dec. - 2012 | ❍ May - 2013 | ❍ Dec. - 2013 |
| ❍ June - 2014 | ❍ Dec. - 2014 | ❍ May - 2016 | ❍ August - 2016 |
| ❍ Dec. - 2016 | ❍ Dec. - 2017 | ❍ Dec. - 2018 | ❍ June - 2019 |
| ❍ Dec. - 2019 [R - 16] | | | |


DECODE®

*A Guide For Engineering Students*

DECODE®

*A Guide For Engineering Students*

# SOFTWARE ENGINEERING

SUBJECT CODE : **CS502PC**

B.Tech., III-I [CSE / IT]



Published by :

TECHNICAL PUBLICATIONS® SINCE 1993 An Up-Thrust for Knowledge

Amit Residency, Office No.1, 412, Shaniwar Peth,
Pune - 411030, M.S. INDIA Ph.: +91-020-24495496/97
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.

ISBN 978-93-90041-58-9

9 789390 041589

JNTUH 18

9789390041589 [1]

# SYLLABUS

## Software Engineering (CS502PC)

### UNIT -I

**Introduction to Software Engineering** : The evolving role of software, changing nature of software, software myths.

**A Generic view of process** : Software engineering- a layered technology, a process framework, the capability maturity model integration (CMMI), process patterns, process assessment, personal and team process models.

**Process models** : The waterfall model, incremental process models, evolutionary process models, the unified process. **(Chapters - 1, 2, 3)**

### UNIT -II

**Software Requirements** : Functional and non-functional requirements, user requirements, system requirements, interface specification, the software requirements document.

**Requirements engineering process** : Feasibility studies, requirements elicitation and analysis, requirements validation, requirements management.

**System models** : Context models, behavioral models, data models, object models, structured methods. **(Chapters - 4, 5, 6)**

### UNIT -III

**Design Engineering** : Design process and design quality, design concepts, the design model.

**Creating an architectural design** : software architecture, data design, architectural styles and patterns, architectural design, conceptual model of UML, basic structural modeling, class diagrams, sequence diagrams, collaboration diagrams, use case diagrams, component diagrams. **(Chapters - 7, 8)**

### UNIT -IV

**Testing Strategies** : A strategic approach to software testing, test strategies for conventional software black-box and white-box testing, validation testing, system testing, the art of debugging.

**Product metrics** : Software quality, metrics for analysis model, metrics for design model, metrics for source code, metrics for testing, metrics for maintenance. **(Chapters - 9, 10)**

### UNIT-V

**Metrics for Process and Products** : Software measurement, metrics for software quality.

Risk management : Reactive Vs proactive risk strategies, software risks risk identification, risk projection, risk refinement, RMMM, RMMM plan.

**Quality Management** : Quality concepts, software quality assurance, software review, formal technical reviews, statistical software quality assurance, software rehability, the ISO 9000 quality standards. **(Chapters - 11, 12)**

# TABLE OF CONTENTS


## Unit - I

**Chapter - 1 Introduction to Software Engineering (1 - 1) to (1 - 5)**

**1.1** Evolving Role of Software . . . . . 1 - 1
**1.2** Changing Nature of Software . . . . . 1 - 2
**1.3** Legacy Software . . . . . 1 - 3
**1.4** Software Myths . . . . . 1 - 4

**Chapter - 2 Generic View of Process (2 - 1) to (2 - 6)**

**2.1** Layered Technology . . . . . 2 - 1
**2.2** Process Framework . . . . . 2 - 1
**2.3** The Capability Maturity Model Integration (CMMI) . . . . . 2 - 2
**2.4** Process Pattern . . . . . 2 - 4
**2.5** Personal and Team Process Models . . . . . 2 - 5

**Chapter - 3 Process Models (3 - 1) to (3 - 15)**

**3.1** Process Models . . . . . 3 - 1
**3.2** Waterfall Model . . . . . 3 - 1
**3.3** Incremental Process Models . . . . . 3 - 2
**3.4** Evolutional Process Models . . . . . 3 - 5
**3.5** Specialized Process Models . . . . . 3 - 10
**3.6** The Unified Process . . . . . 3 - 11

**Fill in the Blanks with Answers for Mid Term Exam . . . . . 3 - 13**

**Multiple Choice Questions with Answers for Mid Term Exam . . . . . 3 - 13**

## Unit - II

**Chapter - 4 Software Requirements (4 - 1) to (4 - 8)**

**4.1** Functional and Non Functional Requirements . 4 - 1
**4.2** User Requirements . . . . . 4 - 4
**4.3** System Requirements . . . . . 4 - 4
**4.4** Interface Specification. . . . . 4 - 6
**4.5** The Software Requirements Document . . . . . 4 - 6

**Chapter - 5 Requirements Engineering Process (5 - 1) to (5 - 11)**

**5.1** Requirement Engineering Process. . . . . 5 - 1
**5.2** Feasibility Studies . . . . . 5 - 1
**5.3** Requirements Elicitation and Analysis . . . . . 5 - 3
**5.4** Requirements Validation. . . . . 5 - 8
**5.5** Requirement Management . . . . . 5 - 9

**Chapter - 6 System Models (6 - 1) to (6 - 15)**

**6.1** Context Model . . . . . 6 - 1
**6.2** Behavioral Models . . . . . 6 - 2
**6.3** Data Models . . . . . 6 - 7
**6.4** Object Models . . . . . 6 - 9
**6.5** Structured Methods . . . . . 6 - 11

**Fill in the Blanks with Answers for Mid Term Exam . . . . . 6 - 12**

**Multiple Choice Questions with Answers for Mid Term Exam . . . . . 6 - 13**

## Unit - III

**Chapter - 7 Design Engineering (7 - 1) to (7 - 10)**

**7.1** Design Process and Design Quality . . . . . 7 - 1
**7.2** Design Concepts . . . . . 7 - 2
**7.3** The Design Model . . . . . 7 - 7
**7.4** Pattern Based Software Design . . . . . 7 - 9

**Chapter - 8 Creating an Architectural Design (8 - 1) to (8 - 37)**

**8.1** Software Architecture . . . . . 8 - 1
**8.2** Data Design . . . . . 8 - 3
**8.3** Architectural Styles and Patterns . . . . . 8 - 4
**8.4** Architectural Design . . . . . 8 - 7

**8.5** Assessing Alternative Architectural Designs . 8 - 8

**8.6** Mapping Data Flow into Software Architecture 8 - 9

**8.7** Conceptual Model of UML . . . . . 8 - 13

**8.8** Basic Structural Modeling. . . . . 8 - 14

**8.9** Class Diagrams . . . . . 8 - 15

**8.10** Sequence Diagram. . . . . 8 - 21

**8.11** Collaboration Diagrams . . . . . 8 - 23

**8.12** Use Case Diagram . . . . . 8 - 26

**8.13** Component Diagram . . . . . 8 - 33

**Fill in the Blanks with Answers for Mid Term Exam . . . . . 8 - 35**

**Multiple Choice Questions with Answers for Mid Term Exam . . . . . 8 - 35**

**Unit - IV**

**Chapter - 9 Testing Strategies (9 - 1) to (9 - 14)**

**9.1** A strategic Approach to Software Testing . . . . . 9 - 1

**9.2** Testing Strategies For Conventional Software. . . . . 9 - 2

**9.3** Black Box Testing. . . . . 9 - 8

**9.4** White Box Testing. . . . . 9 - 10

**9.5** Validation Testing. . . . . 9 - 13

**9.6** System Testing . . . . . 9 - 13

**9.7** Art of Debugging. . . . . 9 - 13

**Chapter - 10 Product Metrics (10 - 1) to (10 - 16)**

**10.1** Software Quality . . . . . 10 - 1

**10.2** Framework for Product Metrics . . . . . 10 - 2

**10.3** Metrics for Analysis Model. . . . . 10 - 3

**10.4** Metrics for Design Model . . . . . 10 - 3

**10.5** Metrics for Source Code . . . . . 10 - 6

**10.6** Metrics for Testing . . . . . 10 - 6

**10.7** Metrics for Maintenance . . . . . 10 - 6

**10.8** Software Measurement . . . . . 10 - 7

**10.9** Metrics for Software Quality. . . . . 10 - 13

**Fill in the Blanks with Answers for Mid Term Exam . . . . . 10 - 14**

**Multiple Choice Questions with Answers for Mid Term Exam . . . . . 10 - 14**

**Unit - V**

**Chapter - 11 Risk Management (11 - 1) to (11 - 7)**

**11.1** Reactive Vs. Proactive Risk Strategies . . . . . 11 - 1

**11.2** Software Risks . . . . . 11 - 1

**11.3** Risk Identification . . . . . 11 - 2

**11.4** Risk Projection . . . . . 11 - 3

**11.5** Risk Refinement . . . . . 11 - 4

**11.6** RMMM . . . . . 11 - 4

**11.7** RMMM Plan . . . . . 11 - 6

**Chapter - 12 Quality Management (12 - 1) to (12 - 10)**

**12.1** Quality Concepts . . . . . 12 - 1

**12.2** Software Quality Assurance . . . . . 12 - 1

**12.3** Software Reviews . . . . . 12 - 3

**12.4** Formal Technical Reviews . . . . . 12 - 4

**12.5** Statistical Software Quality Assurance . . . . . 12 - 5

**12.6** Software Reliability . . . . . 12 - 7

**12.7** The ISO 9000 Quality Standards . . . . . 12 - 8

**Fill in the Blanks with Answers for Mid Term Exam . . . . . 12 - 9**

**Multiple Choice Questions with Answers for Mid Term Exam . . . . . 12 - 9**

**Solved JNTU Question Papers (S - 1) to (S - 2)**

**Solved Model Question Paper (S - 3) to (S - 4)**

UNIT - I

# 1 Introduction to Software Engineering

## 1.1 : Evolving Role of Software

**Q.1 Define software and software engineering**

[JNTU : Part A, Dec.-12, Marks 3]

**Ans. : (i) Software :** Software is a collection of computer programs and related documents that are intended to provide desired features, functionalities and performance. A software can be of two types -

1. Generic software and 2. Custom software.

**(ii) Software engineering :** *Software engineering is a discipline in which theories, methods and tools are applied to develop professional software.*

**Q.2 Explain evolving role of software.**

[JNTU : Part B, March-06, Marks 8]

**Ans. :** (1) The evolving role of software means changing role of software.

(2) Basically any software appears in two roles :

1. Software as **product** and 2. Software as **vehicle.**

**(3) Software as product :**

(i) Being a product the role of software can be recognised by its computing potentials, hardware capabilities and accessibility of network computers by the local hardware.

(ii) Being a product it also acts as an information transformer i.e. producing, managing, modifying and conveying the source of information.

**(4) Software as vehicle :**

(i) Being a process the software acts as a vehicle for driving the product.

(ii) In this role the duty of software is to control the computer or to establish communications between the computers. Thus software plays dual role. It is both a product and a vehicle for delivering a product.

(5) There are various factors that affect the role of software and those are -

(i) Changes in computer architecture(Right from Pentium I to Supercomputers).

(ii) Improvements in hardware performance.

(iii) Vast increase in amount of memory.

(iv) Wide variety of inputs and outputs such as text, audio, video and so on.

**Q.3 Explain the role of software in democratization of knowledge.**

[JNTU : Part B, May-11, Marks 8]

**Ans. :** (1) **Democratization of knowledge** means use of computers at all the public places and development of useful software applications allowed more and more people to access it.

(2) **New technologies** and **improved user experience** increasing number of users of the computers.

(3) Due to increasing scale, consumers have greater access to use and purchase technologically sophisticated products and services.

(4) But this evolutionary role of software brings some **crucial problems.**

(5) Here are some sample **problems** encountered due to evolution on software

(i) Advances in hardware demand for more capable software.

(ii) Ability to build new programs can not meet the demand for new programs and such programs are not sufficient for business and market needs.

(iii) Vast use of computer based systems brings less use of manpower and ultimately society becomes more dependant on machine and not on man.

(iv) Constant struggle for high reliability and quality software.

**Q.4 Explain various characteristics of software.**
**[JNTU : Part B, Dec.-10, April-11, Marks 8 Dec.-19, Marks 3]**

**Ans. :** Some important characteristics of software are :

- **Software is engineered, not manufactured :** Quality problems that occur in hardware manufacturing phase can not be removed easily. On the other hand, during software development process such problems can be rectified.
- **Software does not ware out :** 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. The failure rate remains constant for some period of time and again it starts increasing because of environmental maladies (extreme temperature, dusts, and vibrations).
- 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 in Fig. Q.4.1.
- During the life of software if any change is made, some **defects** may get **introduced**. This causes failure rate to be high. Before the curve can return to original steady state another change is requested and again the failure rate becomes high. Thus the failure curve looks like a spike. Thus frequent changes in software cause it to deteriorate.
- **Most software is custom built rather than being assembled from components :** 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. Most of the software is custom built.

## 1.2 : Changing Nature of Software

**Q.5 Discuss about changing nature of software in detail.** **[JNTU : Part B, Dec.-11,13,16, Mark 5]**

**OR State and explain various areas of software applications.** **[JNTU : Part B, Dec.-12, Marks 5]**

**Ans. : Software can be classified into -**

- **System software** - It is collection of programs written to service other programs. Typical programs in this category are compiler, editors, and assemblers.
- **Application software** - It consists of standalone programs that are developed for specific business need. This software may be supported by database systems.
- **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.

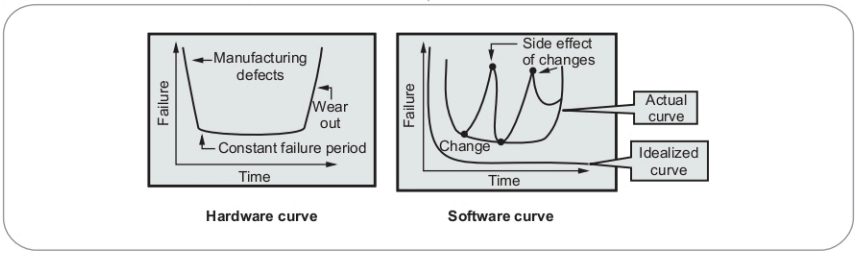



**Fig. Q.4.1 Bath tub curves**

- **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.
- **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.
- **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.

**Q.6 Difference between hardware engineering and software engineering.** [JNTU : Part B, Marks 5]

**Ans. :**

| Sr. No. | Hardware engineering | Software engineering |
|---|---|---|
| 1. | In hardware engineering process the abstract design is translated into hardware component. | In software engineering process the abstract design is translated into coding which is actually the software component. |
| 2. | The scope of problem domain is limited. | The scope of problem domain is very wide. |
| 3. | The hardware engineering is product intensive process. | The software engineering is design intensive process. |
| 4. | The overall cost of the project gets reduced with product standardization. | The overall cost of the project is increased with product standardization process. |
| 5. | This engineering is applicable for controlling, monitoring the electrical system. | This engineering is applicable for computing, updating or monitoring the software components. |

**Q.7 Differentiate between system and application Software.** [JNTU : Part B, Dec.-10, Marks 8]

**Ans. :**

| System software | Application software |
|---|---|
| It is collection of programs which are written to service other programs. | It may consist of a standalone program or a set of programs developed for specific business needs. |
| The communication with hardware must be established by system software. | The communication with user/customer must be established by application software. |
| This software may be supported by DLL or configuration files. | This software may be supported by databases. |
| Typical programs in this category are : Compiler, editors, assemblers. | Typical programs in this category can be MS WORD, Stylesheet programs, Payroll systems, Library management systems. |

## 1.3 : Legacy Software

**Q.8 What is legacy software ? Explain.** [JNTU : Part A, May-09, Marks 3, Dec.-16, Marks 2, Dec.-17, Marks 5, Dec.-19, Marks 2]

**Ans. :** • The Legacy systems are the **older, large complex computer based systems** which may be using the obsolete technology.

- These systems may include the older hardware, software, process and procedures.
- The legacy systems are business critical systems and are used for a long period because it is too risky to replace them. **For example :** Air Traffic Control (ATC) system which may be used from a long period.

**Q.9 What type of changes is made to legacy system if it exhibits poor quality ?** [JNTU : Part B, May-09, Marks 5]

**Ans. :** • Following are the **types of changes that a legacy systems must make -**

1. For meeting the need of new computing environment or technologies the existing legacy systems need to be changed.

2. Sometimes there is a requirement to the use of modern systems or databases. To accommodate this requirement the legacy system must be changed.
3. The software needs to be enhanced for implementing new business requirements.
4. The software architecture needs to be re-designed in order to make it work within networking environment.

- When such type of changes are required the legacy systems has to be reengineered.

**Q.10 Explain why legacy systems evolve as time passes ?** [JNTU : Part B, May-09, Marks 8]

**OR Explain about software evolution in detail.** [JNTU : Part B, Dec.-10, Marks 8]

**Ans. :** Following are the laws that represent the reasons why the system evolve as time passes -

1. **Law of continuing change** : The systems that work in real world computing context are called **E-type** systems and such systems must be changed continuously in order to make them more satisfactory.
2. **Law of self regulation** : The E-type systems are self regulatory by distributing product and process measure close to normal.
3. **Law of increasing complexity** : The E-type system are complex and must be evolved in order to maintain them properly.
4. **Law of organisational stability** : The global activity rate in evolving the E-type systems is invariant.
5. **Law of continuing growth** : The functions of E-type systems are continuously getting increased in order to maintain the user satisfaction.
6. **Law of conservation of familiarity** : When E-type systems get evolved they are associated with developers, sales personnel, users. Hence the system gets more familiar.
7. **Law of declining quality** : The E-type systems need to be rigorously maintained otherwise they get declining quality. Such systems need to get adapted in operational environment changes.
8. **Feedback system Law** : E-type systems posses multilevel, multi-loop, multi-agent feedback systems and must be considered while making the significant changes.

## 1.4 : Software Myths

**Q.11 Discuss in brief about different software Myths and their consequences.** [JNTU : Part B, Dec.-16,17, Marks 5, Marks 10]

**OR What are various software myths prevalent in industry ? Why do the stakeholders believe them ? Contradict the myths with reality.** [JNTU : Dec.-19, Marks 5]

**Ans. :** • **Myth :** Using a collection of standards and procedures one can build software. **(Management Myth).**

**Reality :** Eventhough we have all standards and procedures with us for helping the developer to build software, it is not possible for software professionals to build desired product. This is because - the collection which we have should be complete, it should reflect modern techniques and more importantly it should be adaptable. It should also help the software professional to bring quality in the product.

- **Myth :** Add more people to meet deadline of the project. **(Management Myth).**
  **Reality :** Adding more people in order to catch the schedule will cause the reverse effect on the software project i.e. software project will get delayed. Because, we have to spend more time on educating people or informing them about the project.
- **Myth :** If a project is outsourced to a third party then all the worries of software building are over. **(Management Myth).**
  **Reality :** When a company needs to outsource the project then it simply indicates that the company does not know how to manage the projects. Sometimes, the outsourced projects require proper support for development.
- **Myth :** Even if the software requirements are changing continuously it is possible to accommodate these changes in the software. **(Customer Myth).**
  **Reality :** It is true that software is a flexible entity

but if continuous changes in the requirements have to be incorporated then there are chances of introducing more and more errors in the software. Similarly, the additional resources and more design modifications may be demanded by the software.

- **Myth :** We can start writing the program by using general problem statements only. Later on using problem description we can add up the required functionalities in the program. **(Customer Myth)**
  **Reality :** It is not possible each time to have comprehensive problem statement. We have to start with general problem statements; however by proper communication with customer the software professionals can gather useful information. The most important thing is that the problem statement should be unambiguous to begin with.
- **Myth :** Once the program is running then its over! **(Practitioner's Myth)**
  **Reality :** Even though we obtain that the program is running major part of work is after delivering it to customer.
- **Myth :** Working program is the only work product for the software project. **(Practitioner's Myth)**
  **Reality :** The working program/software is the major component of any software project but along with it there are many other elements that should be present in the software project such as documentation of software, guideline for software support.
- **Myth :** There is no need of documenting the software project; it unnecessarily slows down the development process.
  **(Practitioner's Myth)**
  **Reality :** Documenting the software project helps in establishing ease in use of software. It helps in creating better quality. Hence documentation is not wastage of time but it is a must for any software project.

***END...***

UNIT - I

# 2 A Generic View of Process

## 2.1 : Layered Technology

**Q.1 Explain software engineering as layered technology.** [JNTU : Part B, Marks 5]

**Ans. :** • 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**.


**Fig. Q.1.1 Software engineering : A layered technology**

- 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.

## 2.2 : Process Framework

**Q.2 What is meaning of the term software process.** [JNTU : Part A, Marks 2]

**Ans. :** Software process can be defined as the structured set of activities that are required to develop the software system.

**Q.3 Explain the need for process framework.** [JNTU : Part A, Dec.-14, Marks 4]

**Ans. :** The need for process framework is specified by following checklist -

1. Classify the processes into appropriate groups.
2. Establish common language of communication among the system developers.
3. Establish the standard practices and methods for software development.
4. Provide visibility on how the system gets developed.

**Q.4 Explain the common process framework in detail.** [JNTU : Part B, Dec.-17, Marks 5]

**Ans. : Process framework activities**

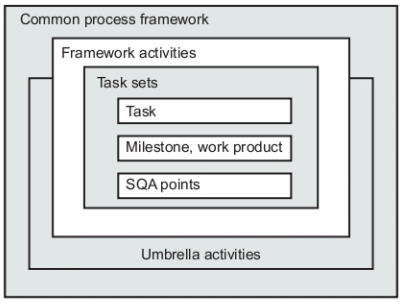



**Fig. Q.4.1 Common process framework**

- 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.

## 2.3 : The Capability Maturity Model Integration (CMMI)

**Q.5 Explain CMMI model with neat sketch.**
**[JNTU : Part B, May-13,Dec.-14,16,19, Marks 5]**

**OR Give an overview of Capability Maturity Model Integration. Which level of organizations as a customer you would prefer and why ?**

**[JNTU : Dec.-19, Marks 5]**

**Ans. :** • The Capability Maturity Model(CMM) is used in assessing how well an organization's processes allow to complete and manage new software projects. Refer Fig. Q.5.1.

- The Capability Maturity Model Integration(CMMI) a **process improvement** training and appraisal program used specially in software development. CMMI can be used to guide process improvement across a project, division, or an entire organization.
- The process area defines the goal and practices of a process.
- Following is a process **capability levels** for the processes

**Level 0 : Incomplete -**

- At this level,the **process areas** is not performed.
- This is a level at which the goals and objectives of CMMI level 1 are not defined completely.

**Level 1 : Performed -**

- The goals and objectives of process are satisfied.
- The **work tasks** that are required to perform are for development of the product are defined.

**Level 2 : Managed -**

- The criteria for level 1 are satisfied at this stage.
- The work associated with the process area is conducted as per the business or organization policy.
- All the required resources are easily available at this stage. Stakeholders take active participation in project development.

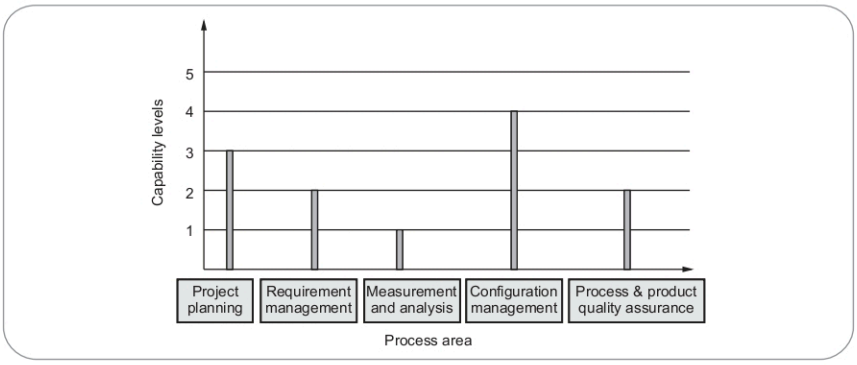



**Fig. Q.5.1 CMMI profile for capability and process area**

- The work tasks and work products are monitored, reviewed and evaluated.

**Level 3 : Defined -**

- The criteria for level 2 are satisfied at this stage.
- 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 : Quantitatively Managed -**

- The criteria for level 3 are satisfied at this stage.
- Both the software process and product are quantitatively understood and controlled using detailed measures.

**Level 5 : Optimizing -**

- The criteria for level 4 are satisfied at this stage.
- Establish mechanisms to plan and implement change.
- Innovative ideas and technologies can be tested.
- The modifications in process area can be made to meet changing customer needs and continually improve the process area under consideration

- Thus CMMI is used for improving the software project.

**Q.6 Discuss major problems with capability maturity Model.** [JNTU : Part A, June 14, Marks 3]

**Ans. :** 1. The CMM is a goal and not a method.

2. Some organizations make use of CMM simply as a stamp of approval without any commitment towards the process improvement.
3. CMM does not help the projects that are in crisis.
4. CMM is based on the method of reusing past results for future software projects. To create a brand new software of unknown size, unknown risks, unknown human creativity CMM is not a right model.

**Q.7 What is staged and continuous CMMI ?** [JNTU : Part A, Marks 2]

**Ans. : (i) Staged CMMI :** Staged representation is an approach which focuses on the **predefined set of process areas**. Organizations use predefined and proven **improvement path**. This improvement path is described by a set of components called **maturity levels**.

**(ii) Continuous CMMI :** The continuous representation is concerned with **selecting particular process area** to improve and desired capability level for that process area. Hence there is a name **incomplete** which is given to the starting point of continuous representation.

## 2.4 : Process Pattern

**Q.8 What is process pattern ?**
[JNTU : Part A, Marks 2]

**Ans. :** • The process pattern is a template which appears as a general solution to a common problem. Process pattern acts as a consistent method for describing an important characteristic of software process.

- Using process pattern we can easily build the process that satisfies all the requirements.

**Q.9 Explain three types of process patterns.**
[JNTU : Part A, Marks 3]

**Ans. :** Ambler has suggested three types of patterns

1) **Task pattern -** It represents the software engineering action or work task which is a part of process. For example, Formal Technical review is a task pattern.
2) **Stage pattern -** It defines the process framework activity. A framework activity has multiple work tasks; hence stage pattern consists of multiple task patterns. For example, Coding phase is a stage pattern.
3) **Phase patterns -** It defines the sequence of framework activities. For example the phase pattern can be spiral model or rapid prototype model.

**Q.10 Give and explain the template used for process pattern.** [JNTU : Part B,May-09, Marks 8]

**Ans. :** Scott Ambler - An object oriented consultant has proposed a template for process pattern as follows :

- Pattern name
- Intent
- Type
- Initial context
- Problem
- Solution
- Resulting context
- Known uses

The description of process pattern is as follows -

**Pattern name :** The pattern name should be a meaningful name given to the pattern. From pattern name one can guess its functionality.

**Intent :** The objective or the purpose of the pattern should be described here.

**Type :** The type of pattern should be specified here.

Ambler has suggested three types of patterns

1) ***Task Pattern*** **-** It represents the software engineering action or work task which is a part of process. For example, **Formal Technical review** is a task pattern.
2) ***Stage Pattern*** **-** It defines the process framework activity. A framework activity has multiple work tasks; hence stage pattern consists of multiple task patterns. For example, **Coding phase** is a stage pattern.
3) ***Phase Patterns*** **-** It defines the sequence of framework activities. For example the phase pattern can be **spiral model or rapid prototype model.**

**Initial context :** In this section the conditions under which the pattern applies are described.

Sometimes the **entry conditions** must be true before the process begins.

In this section following issues need to described :

1. The set of organisational or team related activities that have already occurred.
2. The list of entry state processed.
3. Already existing software engineering or project related information.

**Problem :** Under this section the problem is mentioned for which the pattern is to be described. For example : *Insufficient requirements* is a problem. That means customers are not sure about what they want exactly. They could not specify the requirements in proper manner.

**Solution :** Every problem for which pattern has to be described should be accompanied with some solution. For example: The problem of *insufficient requirements* has solution. That is - Establish effective

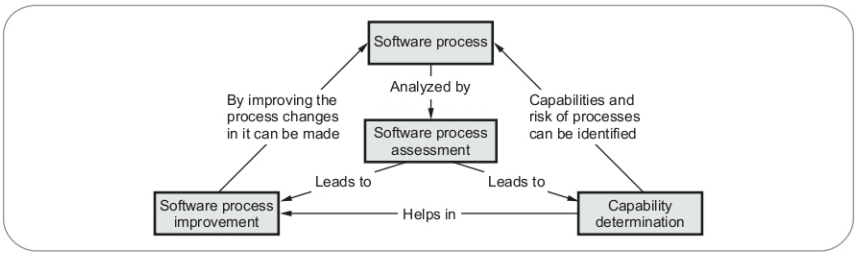



**Fig. Q.11.1 Software process assessment**

communication with the customer. Ask questions in order to obtain meaningful requirements.

**Resulting context :** It describes the results after successful implementation of pattern. The resulting context should have following type of information on successful completion of pattern -

1. The team-related or organizational activities that must have occurred.
2. Exit state for the process.
3. The software engineering information or project information that has been developed.

**Known uses/Examples :** The specific instances or applications in which the described pattern is useful should be mentioned under this section. In other words we describe applicability of the pattern. For example : Spiral model is useful for the large scale projects in which work products must be examined in several iterations.

**Q.11 What is software process assessment ? Discuss different approaches to it.**
**[JNTU : Part B, August- 07, Marks 7]**

**Ans. :** Refer Fig. Q.11.1.

*Process assessment is an activity in which it is ensured whether the software meets the basic criteria for successful software engineering project.*

Following approaches are used for software assessment -

**Standard CMMI assessment method for process improvement :** It is a five step process assessment model. These five steps are initiating, diagnosing, establishing, acting and learning. This model makes use of SEI CMM as the base model.

**CMM-based appraisal for internal process improvement :** This method provides the diagnostic technique for internal process assessment.

**SPICE :** Using this standard all the requirements are analyzed for software process assessment. This standard helps in doing an objective evaluation on efficiency of any process.

**ISO 9001:2000 :** This is a popularly used standard for improving the overall quality of the organization. The International Organization for Standardization i.e. ISO has developed this standard.

## 2.5 : Personal and Team Process Models

**Q.12 Explain the Personal Software Process Model(PSP) in detail.** **[JNTU : Part B, Marks 5]**

**Ans. :** The personal software process model is a SEI(Software Engineering Institute) technology that brings discipline to software development habits of individual software engineer.

Under PSP model five framework activities are suggested as follows - Refer Fig. Q.12.1.


**Fig. Q.12.1 Framework activities in PSP**

**Planning :** In this activity, requirements are identified based on these requirements size, resource estimates and defect estimates are made. All metrics are arranged in template form. Development task are identified and project schedule is created.

**High level design :** The specification is created first and then component level design is created. To resolve the uncertainty prototypes are created.

**Review :** Formal technical reviews are made for uncovering the errors. Metrics are formed for important tasks and recorded.

**Development :** Under this activity code is generated. The generated code is then reviewed, modified and tested. Metrics are formed for important tasks and recorded.

**Postmortem :** Using the collected measures and metrics the effectiveness of the process is analysed. These measures and metrics should direct certain changes in the process in order to improve its efficiency.

**Q.13 Explain the Team Software Process Model(TSP) in detail.** [JNTU : Part B, Marks 5]

**Ans. :** The term process model (TSP) is designed to produce strategy and set of operational procedures for using disciplined software methods at **team levels**. The goal of TSP is to have **self directed project team** for producing high quality software. Following are the objectives of Term Process Models.

1. Build the self directed team for planning the software project in systematic manner. The normal size of team should be 3 to 20 software engineers.
2. Indicate the project manager for the coaching needed by the team members for the performance improvement.
3. Using CMM level 5 (optimizing level), improve the software process.
4. Provide the improvement guidance for the software team.
5. Provide the university teaching.

There are five framework activities in TSP

- Launch
- High level design
- Implementation
- Integration and test
- Postmortem.

**Q.14 Compare the Personal and team process models.** [JNTU : Part B, Dec.-12, Marks 8]

**Ans. :**

| Personal Software Process (PSP) | Team Software Process (TSP) |
|---|---|
| It brings discipline to software development habits of individual software engineer. | It brings disciplined software methods at team level. |
| Limited scope. | Wide scope. |
| Simple to implement. | Complex to implement. |
| Framework activities : Planning design, review, development, postmortem. | Framework activities : Launch, design, implementation, integration and test postmortem. |

*END...*

UNIT - I

# 3 Process Models

## 3.1 : Process Models

**Q.1 What is software process ?**

[JNTU : Part A, Marks 2]

**Ans. :** Software process can be defined as the structured set of activities that are required to develop the software system. The fundamental activities are -

1. Specification 2. Design and implementation
3. Validation 4. Evolution

**Q.2 What is process model ?**

[JNTU : Part A, Marks 3]

**Ans. :** • Software process can be defined as the structured set of activities that are required to develop the software system.

- The fundamental activities are : (i) Specification (ii) Design and implementation (iii) Validation (iv) Evolution
- A software process model is an abstract representation of a process. It presents a description of a process from some particular perspective.

**Q.3 What do you understand by the term software development life cycle ?**

[JNTU : Part A, May-09, Marks 3]

**Ans. :** Software development life cycle model or process model is a diagrammatic representation of various activities required to make software product. Along with this diagrammatic representation, description of each phase of SDLC must be given. Each phase of SDLC is carried out by conducting various activities required to make the software product. The SDLC also specifies the **order** in which each phase must be executed. Various life cycle models are -

1. Waterfall model 2. Prototype model
3. Spiral model
4. Evolutionary development model
5. Iterative enhancement model.

## 3.2 : Waterfall Model

**Q.4 What is waterfall model ?**

[JNTU : Part B, Marks 5]

**OR What is linear sequential model ?**

[JNTU : Part B, Marks 5]

**Ans. :** • The waterfall model is also called as 'linear-sequential model' or 'classic life cycle model'. It is the oldest software paradigm. This model suggests a systematic, sequential approach to software development.

- The software development starts with requirements gathering phase. Then progresses through analysis, design, coding, testing and maintenance. Following Fig. Q.4.1 illustrates waterfall model. (See Fig. Q.4.1 on next page.)

1. **Requirements gathering and analysis :** In this phase, basic requirements of the system must be identified and analysed.
2. **Design :** In this phase, data structure, software architecture, interface definition and algorithmic details are designed.
3. **Coding :** It is an implementation phase in which the software programs are created.
4. **Testing :** It is a phase in which errors and bugs are corrected.
5. **Maintenance :** In this phase, the system is installed and put in practical use.

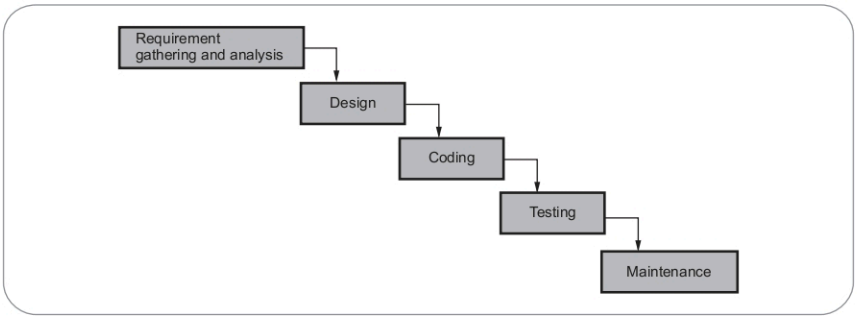



**Fig. Q.4.1 Waterfall model**

**Q.5 Discuss merits and demerits of waterfall model.** [JNTU : Part B, Marks 5]

**Ans. : Merits**

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

**Demerits**

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.
3. The customer can see the working model of the project only at the end.
4. Linear nature of waterfall model induces blocking states, because certain tasks may be dependent on some previous tasks.

**Q.6 Explain how water-fall model is applicable for the development of the following systems :**
**a) University accounting system**
**b) Interactive system that allows railway passengers to find time and other information from the terminals installed in the station.**
[JNTU : Part B, Dec.-10, CSE/IT/CS & SE, Marks 8]

**Ans. : a) University accounting system :** If the software developers who have the experience in developing the account systems then building university account system based on existing design could be easily managed with water-fall model.

**b) Interactive system that allows railway passengers to find time and other information from the terminals installed in the station.**

For developing such interactive system, all the requirements must be correctly identified and analyzed before the designing of the project. The requirements of end-users must be correctly and un-ambiguously understood by the developers prior to design phase. Once the requirements are well defined then using disciplined design, coding and testing phases the required system can be built using water-fall model.

**Q.7 A software project which is considered to be very simple and the customer is in position of giving all the requirements at the initial stage, which process model would you prefer for developing the project ?** [JNTU : Part B, Dec.-09]

**Ans. :** The linear sequential model or a waterfall model is appropriate when a simple project has to be developed with already known requirements.

**Waterfall model -** Refer Q.4.

## 3.3 : Incremental Process Models

**Q.8 Describe the incremental software development process model.**
[JNTU : Part B, May-03,04, Marks 8]

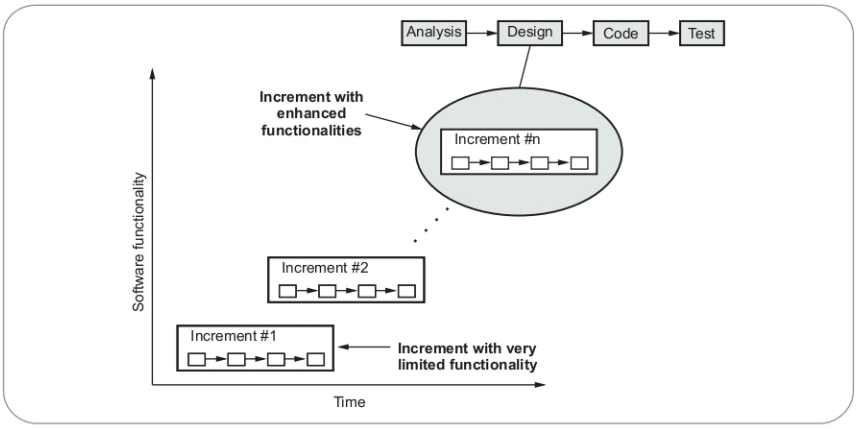



**Fig. Q.8.1 The incremental model**

**Ans. :** • The incremental model has same phases that are in waterfall model. But it is iterative in nature. The incremental model has following phases.

1. Analysis 2. Design
3. Code 4. Test

- The incremental model delivers series of releases to the customer. These releases are called **increments**. More and more functionality is associated with each increment.
- The first increment is called **core product**. In this release the basic requirements are implemented and then in subsequent increments new requirements are added.
- The word processing software package can be considered as an example of incremental model.
- **When to choose it ?**

1. When requirements are reasonably well-defined.
2. When overall scope of the development effort suggests a purely linear effort.
3. When limited set of software functionality needed quickly.

**Q.9 What are the phases encompassed in RAD model ?** **[JNTU : Part A, Marks 2]**

**Ans. :** Various phases in RAD model are - i) Communication ii) Planning iii) Modeling iv) Construction and v) Deployment

**Q.10 Illustrate RAD model with neat sketch.**
**[JNTU : Part B, June-14, Marks 7]**

**Ans. :** • 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 **communication** phase the developers communicate with the users of the system and understand the business process and requirements of the software system.
- During **planning** phase, the analysis on the gathered requirements is made and a planning for various software development activities is done.

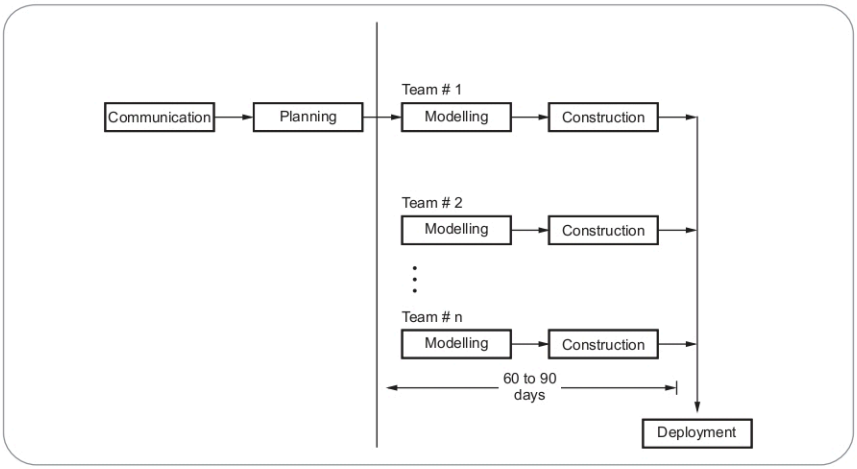



**Fig. Q.10.1 The RAD model**

- During the **modelling** phase various models are created. Those models are Business model, data model and process model.
- The **construction** is an activity in which using the existing software components and automatic code generation tool the implementation code is created for the software system. 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.

**Q.11 Which type of applications suit RAD model ? Justify your answer.** **[JNTU : Part A, Marks 3]**

**Ans. :** The RAD model is suitable for information system applications, business applications and the for systems that can be modularized because of following reasons -

1. This model is similar to waterfall model but it uses very short development cycle.
2. It uses component-based construction and emphasizes reuse and code generation.
3. This model uses multiple teams on scalable projects.
4. The RAD model is suitable for the projects where technical risks are not high.
5. The RAD model requires heavy resources.

**Q.12 Provide three examples of software projects that would be amenable to incremental model. Be specific.** **[JNTU : Part A, Marks 3]**

**Ans. :** There can various examples of software projects that would be amenable to incremental model. For instance -

1. Banking software service : This service can be personal service. That means for personal banking system the incremental model can be used. In later state of increments, this system can implement insurance service, home loans and some other features of banking services.
2. Web browser application : The base application can be developed and distributed. This is the basic increment of the application. In the later increments the plugins can be provided to enhance the experience of web browser applications.
3. Operating system software : The operating system software providing the basic system handling functionalities is the first increment. After the release of the basic versions then updates or security patches are provided to the

customer in the form of increments. Various distribution package in the form of versions such as basic home edition, premium, ultimate and so on can be the increments of operating system software.

**Q.13 Discuss the statement "RAD model is a high speed adaptation of the waterfall model".**
**[JNTU : Part B, Dec.-11, Marks 8]**

**Ans. :** Various phases in RAD model are requirements gathering, analysis, design, construction and finally deployment which are just similar to that of waterfall model.

Similar to waterfall model, the requirements need to be fully understood. But in RAD model the component based approach is adopted. Multiple teams work on various modules to obtain the result parallely. There is a need for appropriate modularization in RAD.

Thus RAD model is a model in which the rapid development is achieved by using a component based construction approach and using many steps from waterfall model. Hence it is said that RAD model is a high speed adaptation of waterfall model.

**Q.14 Explain the prerequisite for applying the RAD model.** **[JNTU : Part B, Dec.-11, Marks 8]**

**Ans. :** RAD model is a incremental software process model that focuses on short development cycle time. The prerequisite for applying RAD model are - **large number of resources, multiple teams**, and committed and **skilled developers**.

The projects of RAD model make use of heavy resources. It requires multiple teams or large number of people to work on scalable projects. The model requires heavily committed developers and customers for gathering the requirements. It is necessary to have sufficient **requirement gathering and analysis** performed before the development starts in RAD model.

## 3.4 : Evolutional Process Models

**Q.15 Explain about prototyping in detail. Explain its merits and demerits.**
**[JNTU : Part B, April-11, June-14, Marks 8]**

**Ans. :** • In prototyping model initially the **requirement gathering** is done.

- Developer and customer define overall objectives; identify areas needing more requirement gathering.
- Then a **quick design** is prepared. This design represents what will be visible to user in input and output format.

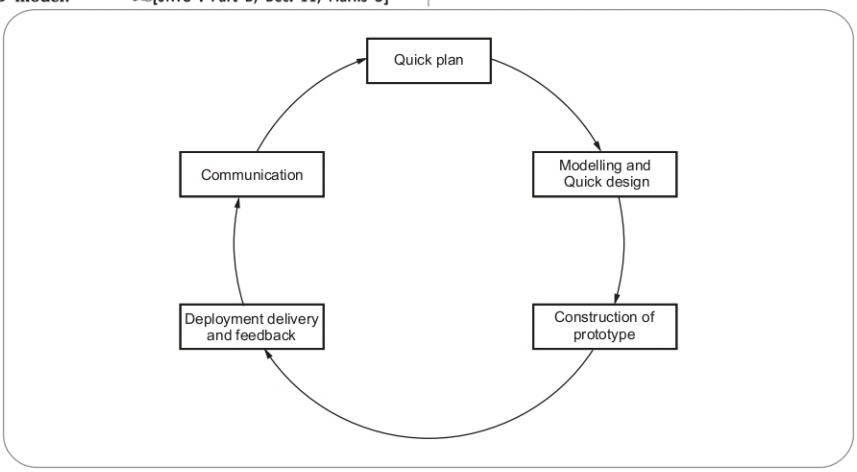



**Fig. Q.15.1 Th prototype model**

- From the quick design a **prototype** is prepared. Customer or user **evaluates the prototype** in order to refine the requirements. Iteratively prototype is tuned for satisfying customer requirements. Thus prototype is important to identify the software requirements.
- When working prototype is built, developer use existing program fragments or program generators to throw away the prototype and rebuild the system to high quality.

**Merits**

1. Users are actually involved in development.
2. Since during development process, the working model of the system is provided, the users get better understanding of the system being developed.
3. Errors can be detected and corrected at the early stage.

**Demerits**

1. In the first version itself, customer often wants "few fixes" rather than rebuilding of the system whereas rebuilding of new system maintains high level of quality.
2. The first version may have some compromises.
3. Sometimes developer may make implementation compromises to get prototype working quickly. Later on developer may become comfortable with compromises and forget why they are inappropriate.

**Q.16 Define spiral model. Explain the characteristics of it.**

**[JNTU : Part B, Dec.-10, Marks 8, May-13, Marks 5]**

**Ans. :** • This model possess the **iterative nature** of prototyping model and controlled and **systematic approaches** of the linear sequential model.

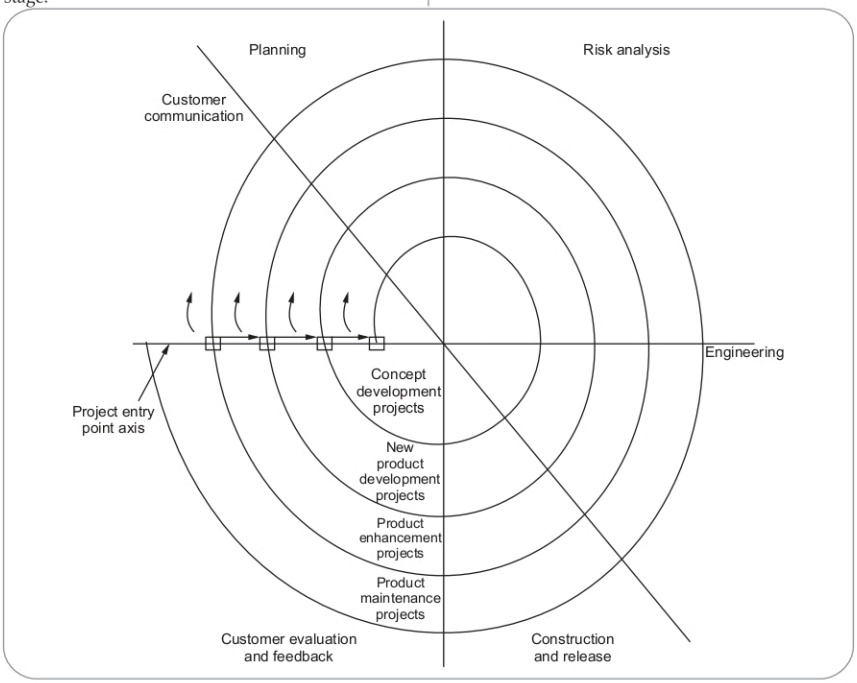



**Fig. Q.16.1 Spiral model**

- The spiral 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. Q.16.1. (See Fig. Q.16.1 on previous page.)
- 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.

**Q.17 What are merits and demerits of spiral model ?** **[JNTU : Part B, Feb.-10, Marks 6]**

**Ans. : Advantages of spiral model**

- Requirement changes can be made at every stage.
- Risks can be identified and rectified before they get problematic.

**Drawbacks of spiral model**

- It is based on **customer communication**. If the communication is not proper then the software product that gets developed will not be up to the mark.
- It demands considerable **risk assessment**. If the risk assessment is done properly then only the successful product can be obtained.

**Q.18 As you move outward along with process flow path of the spiral model, what can we say about the software that is being developed or maintained ?** **[JNTU : Part A, Marks 3]**

**Ans. :** When software engineering team moves around the spiral, the first circuit around the spiral results in development of product specification. The subsequent passes around the spiral might be used to develop prototype in more subsequent manner. In each pass, through planning region, some adjustments to project plan are made. Cost and schedule adjustments can also be made according to customer feedback.

**Q.19 How does "Project Risk" factor affect the spiral model of software development ?** **[JNTU : Part A, Marks 2]**

**Ans. :** The spiral model demands considerable risk assessment because if a major risk is not uncovered and managed, problems will occur in the project and then it will not be acceptable by end user.

**Q.20 How does a spiral model represent a process suitable to represent a real time problem ?** **[JNTU : Part A, Marks 3]**

**Ans. :** Spiral model represents a process suitable to represent a real time problem because of following reasons -

1. Software evolves as the project progresses. And at every evolutionary level the risks are identified and managed and risks are reduced at every stage.
2. It enables the developer to apply the prototype approach at any stage in the evolution of the product. It helps in adopting the approach systematic stepwise development of the product.

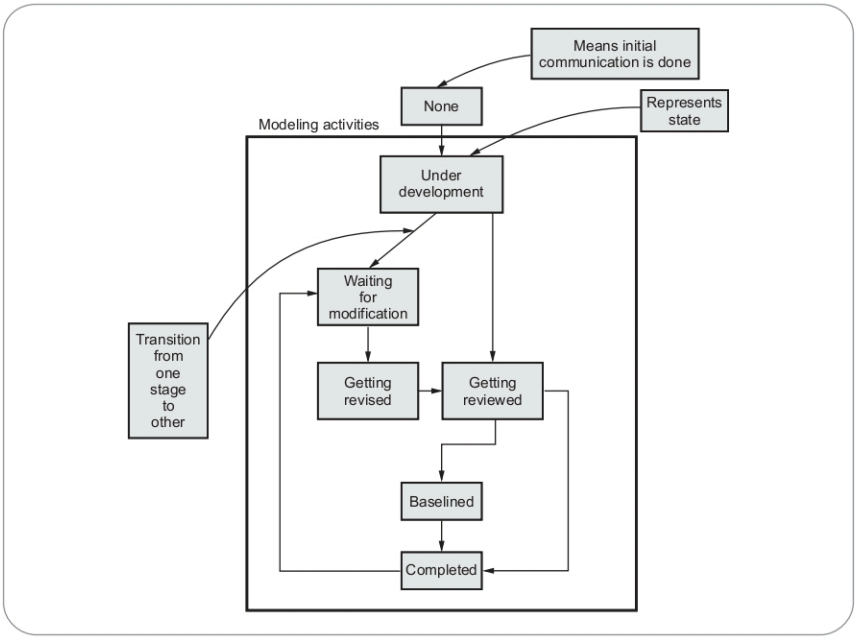



**Fig. Q.21.1 Concurrent development model**

3. The iterative frameworks help in analyzing the product at every evolutionary stage.
4. The spiral model demands a direct consideration of technical risks at all stages of project. The risks are reduced before they get problematic.

**Q.21 Explain the modeling activity and the development activity of the concurrent development process model with the state-transition diagram.**

**☞[JNTU : Part B, May-09, Marks 8]**

**Ans. :** • The concurrent development model is also called as **concurrent engineering.**

- In this model, the framework activities or software development tasks are represented as **states.**
- The **modeling** or **designing** phase of software development can be in one of the states like *under development*, *waiting for modification*, *under revision* or *under review* and so on.
- Fig. Q.21.1 represents these states.
- All the software development activities exist concurrently in this model but these activities can be in various **states.**
- These states make transitions. That is during **modeling,** the transition from **under development** state to **waiting for modification** state occurs.
- This model basically defines the **series of events** due to which the transition from one state to another state occurs. This is called **triggering.** These series of events occur for every software development activity, action or task.
- This model defines various activities that occur concurrently and a network of activities is defined.

**Q.22 Differentiate between spiral and prototyping model.** [JNTU : Part B, Dec.-10, Marks 8]

**Ans. :**

| Sr. No. | Spiral model | Prototyping model |
|---|---|---|
| 1. | The development team with less domain knowledge can be accommodated due to iterative nature of this model. The change in technology in the later phase can not be tolerated. | The development team has adequate domain knowledge. Similarly they can adopt the new technologies if product demands. |
| 2. | All the end-users need not be involved in all the phases of development. | All the end-users are involved in all phases of development. |
| 3. | Funding are not stable for the projects that can be developed using spiral model. | Funding are stable for these type of projects. |
| 4. | The requirements that are gathered and analyzed are high reliability requirements. | Some requirements are gathered initially, but there may be change in requirements when the working prototype is shown to the customer. |

**Q.23 Compare linear sequential model with the spiral model.** [JNTU : Part B,Dec.-12, Marks 7]

**Ans. :**

| 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. |

**Q.24 Distinguish between incremental process model and evolutionary process model.** [JNTU : Part B, Dec.-13, Marks 8]

**Ans. :**

| Sr. No. | Evolutionary process model | Incremental process model |
|---|---|---|
| 1. | Some requirements are gathered initially, but there may be change in requirements when the working prototype is shown to the customer. | The requirements are precisely defined and there is no confusion about the final product of the software. |
| 2. | The development team has adequate domain knowledge. Similarly they can adopt the new technologies if product demands. | The development team with less domain knowledge can be accommodated due to iterative nature of this model. The change in technology in the later phase cannot be tolerated. |
| 3. | All the end-users are involved in all phases of development. | All the end-users need not be involved in all the phases of development. |
| 4. | There can be use of some reusable software components in project development process. | There is no use of reusable components in development process. |

**Q.25 Differentiate between prototyping and incremental process model.**

☞[JNTU : Part B, May-09, Marks 8]

**Ans. :**

| Sr. No. | Prototyping | Incremental process model |
|---|---|---|
| 1. | Some requirements are gathered initially, but there may be change in requirements when the working prototype is shown to the customer. | The requirements are precisely defined and there is no confusion about the final product of the software. |
| 2. | The development team has adequate domain knowledge. Similarly they can adopt the new technologies if product demands. | The development team with less domain knowledge can be accommodated due to iterative nature of this model. The change in technology in the later phase can not be tolerated. |
| 3. | All the end-users are involved in all phases of development. | All the end-users need not be involved in all the phases of development. |
| 4. | There can be use of some reusable software components in project development process. | There is no use of reusable components in development process. |

**Q.26 Compare and contrast between waterfall model and spiral model with neat diagrams.**

☞[JNTU : Dec.-19, Marks 5]

**Ans. :** Refer Q.4, Q.16 and Q.23.

## 3.5 : Specialized Process Models

**Q.27 Enlist the names of three specialized models used.** ☞[JNTU : Part A, Marks 2]

**Ans. :** The three specialized models are - i) Component based development ii) Formal methods models iii) Aspect oriented models

**Q.28 Explain component based development model in detail.** ☞[JNTU : Part B, Marks 5]

**Ans. :** • 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.

**Q.29 What are the merits of component based development model ?** ☞[JNTU : Part A, Marks 2]

**Ans. :** • **Software reusability** is the major advantage of component based development.

- The **reusability** reduces the development cycle time and overall cost.

**Q.30 Explain Aspect Oriented Software Development(AOSD).** ☞[JNTU : Part B, Marks 5]

**Ans. :** • In traditional software development process, the system is decomposed into multiple units of **primary functionality**. But there are other issues of **concern** that do not fit into these primary functionalities. Later on this becomes programmers' duty to code modules corresponding to the primary functionality and to incorporate **all other concerned** issues wherever appropriate.

- Programmers need to keep in mind all the things that need to be done, how to deal with each issue, the problems associated with them and the correct execution.
- **Aspect-oriented software development** focuses on the identification, specification and representation of **cross-cutting concerns and their modularization** into separate functional units as well as their **automated** composition into a working system.
- **Aspectual requirements** define these cross-cutting concerns that have impact on the software architecture.
- **Aspect Oriented Software Development (AOSD)** is often referred as **aspect oriented programming.**
- It is a relatively new software engineering paradigm and is not matured enough. But is likely that it will adopt the characteristics of both the spiral and concurrent process models.

## 3.6 : The Unified Process

**Q.31 What is unified process ?**

☞[JNTU : Part A, Dec.-19, Marks 3; Part B, Dec.-19, Marks 5]

**Ans. :** • The unified process is a framework for object oriented models. This model is also called as **Rational Unified Process model (RUP)**.

- It is proposed by Ivar Jacobson, Grady Bootch and James Rumbaugh.
- This model is iterative and incremental by nature.

**Q.32 What are the advantages of unified process ?**

☞[JNTU : Part A, Dec.-16, Marks 3 Part B, Dec.-19, Marks 5]

**Ans. :** (1) This process gives constant user feedback. There is involvement of user in development process.

(2) Testing is conducted at the early stage.

(3) This processes apply use cases where needed.

(4) It allows to manage the change requests.

(5) Evolutionary approach can lead to clean implementation.

(6) It gives out incremental releases so user can understand the System development.

**Q.33 Explain the processes that are involved in a unified approach for software development.**

☞[JNTU : Part B, May-09, Marks 8]

**OR Explain various work produced in elaboration phase of unified process.**

☞[JNTU : Part B, Dec.-11, Marks 7]

**Ans. :** The process involved in unified process model are -

- **Inception**

(i) In this phase there are two major activities that are conducted : ***Communication*** and ***Planning***.

(ii) By having customer communication business requirements can be identified. Then a rough architecture of the system is proposed. Using this rough architecture it then becomes easy to make a plan for the project.

(iii) **Use cases** are created which elaborates the user scenario. Using use cases the sequence of actions can be identified.

(iv) Thus use cases help to identify the scope of the project which ultimately proves to be the basis for the plan.

- **Elaboration**

(i) Elaboration can be done using two activities : ***Planning*** and ***Modeling***.

(ii) In this phase the use cases are redefined. And an architectural representation is created using five models such as **use-case model, analysis model, design model, implementation model and deployment model.**

(iii) Thus **executable baseline** gets created.

(iv) Then a **plan** is created carefully to check whether scope, risks and delivery dates are reasonable.

- **Construction**

(i) The main activity in this phase is to make the use cases **operational**.

(ii) The analysis and design activities that are started in elaboration phase are completed in this phase and a **source code** is developed which implements all desired functionalities.

(iii) Then **unit testing** is conducted and **acceptance testing** is carried out on the use cases.

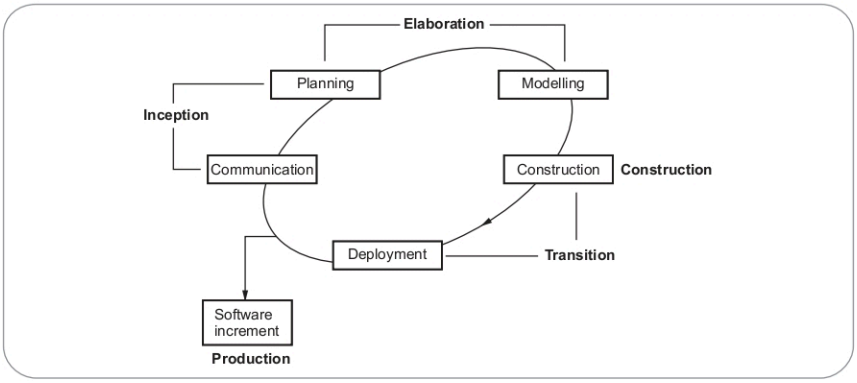



**Fig. Q.33.1 Unified process**

- **Transition**

(i) In the transition phase all the activities that are required at the time of deployment of the software product are carried out.

(ii) **Beta testing** is conducted when software is delivered to the end user.

(iii) User **feedback** report is used to remove defects from the created system.

(iv) Finally software team prepares user manuals, installation guides and trouble shooting procedures. This makes the software **more usable** at the **time of release.**

- **Production**

This is the **final phase** of this model. In this phase mainly the **maintenance activities** are conducted in order to support the user in operational environment.

# For Mid Term Exam



## Fill in the Blanks for Mid Term Exam

1. The two types of software product are generic and ____.
2. Software is both ____ and a vehicle that delivers a product.
3. Software is a process and --------.
4. One of the characteristic of software is - Software doesn't ____.
5. Quality focus layer, process layer, methods layer and____ layer is a known as software engineering as layered technology.
6. SDLC stands for _____.
7. Application of artificial intelligence software is _____. **[JNTU : August-16]**
8. The process pattern can be of three types and these types are task pattern, stage pattern and ___ pattern.
9. The PSP stands for _____.
10. The goal of TSP is to build a ___ team that organizes itself to produce high quality software.
11. The CMMI stands for ____.
12. _____ is characterized by longevity and business criticality. **[JNTU : August-16]**
13. _____ activity combines code generation and the testing. **[JNTU : August-16]**
14. Waterfall model is also called as _____ **[JNTU : August-16]**
15. The RAD stands for ____
16. Risk Analysis is conducted in ____ model.
17. ___ is a collection of software engineering work task, milestones and deliverable that must be accomplished to complete particular project.

## Multiple Choice Questions for Mid Term Exam

**Q.1** _____ software is a collection of programs written to service other. **[JNTU : August-16]**

[a] Application [b] System
[c] Engineering [d] Embedded

**Q.2** A growing trend that results in distribution of _____ for systems applications so that customer can make local modifications. **[JNTU : August-16]**

[a] netsource [b] source code
[c] object code [d] outsource

**Q.3** CMM level 2 is named as _____. **[JNTU : August-16]**

[a] incomplete [b] performed
[c] defined [d] managed

**Q.4** Software engineering is a layered technology. It consists of ___ layers.

[a] two [b] three
[c] four [d] five

**Q.5** _____ process framework activity is responsible for feedback.

[a] Communication [b] Modeling
[c] Construction [d] Deployment

**Q.6** The _____ model is a realistic approach to the development of large-scale systems and software. **[JNTU : August-16]**

[a] incremental [b] RAD
[c] prototype [d] spiral

**Q.7** Design phase is followed by _____.

[a] coding [b] testing
[c] maintenance

**Q.8** Which of the following is evolutionary process model ?

a Incremental model
b Spiral model
c Concurrent development model
d All of the above

**Q.9** Which two models doesn't allow defining requirements early in the life cycle ?

a Waterfall and RAD
b Prototyping and Spiral
c Prototyping and RAD
d Waterfall and Spiral

**Q.10** If XYZ company wants to enhance the current software product, then which process model will you prefer to perform this job ?

a Spiral
b Iterative enhancement model
c RAD model
d Both (b) and (c)

**Q.11** If the project is to be completed within the tight schedule then we choose waterfall model (True/False)

**Q.12** Which of the following is not a software process model ?

a Waterfall model
b Incremental model
c Capability maturity model
d Spiral model

**Q.13** Choose the type of project where the prototyping model of software development is well suited.

a For the projects with large development teams
b When the customer cannot define the requirements clearly
c When the requirements are well defined.

**Q.14** Which model is popular for students small projects ?

a Waterfall model
b RAD model
c Spiral model
d WIN WIN model

**Q.15** If the requirements are easily understandable and defined then which model is best suited ?

a Spiral model
b Prototyping model
c Waterfall model
d Incremental model

**Q.16** Spiral Model has high reliability requirement (True/False)

**Q.17** The model in which the requirements are implemented by its category is _____.

a evolutionary development model
b waterfall model
c prototyping model
d iterative enhancement model

**Q.18** RAD model has reliability requirement (True/False)

**Q.19** The most important feature of spiral model is _____.

a requirement analysis
b risk management
c quality management
d configuration management

**Q.20** What is the main difference between the spiral model and other models ?

a Each loop is considered as a phase
b Describe the process as a spiral
c Does not include planning activities
d Explicit recognition of risk

**Answer Keys for Fill in the Blanks :**

| | | | |
|---|---|---|---|
| **1.** | custom | **2.** | product |
| **3.** | product | **4.** | wear out |
| **5.** | tools | **6.** | Software Development Life Cycle |
| **7.** | robotics, expert system and pattern recognition | **8.** | phase pattern |
| **9.** | Personal Software Process | **10.** | self-directed |
| **11.** | Capability Maturity Model Integration | **12.** | Legacy software |
| **13.** | Construction | **14.** | linear sequential model or classic life cycle model |
| **15.** | Rapid Application Development | **16.** | spiral |
| **17.** | Task set | | |

**Answer Keys for Multiple Choice Questions**

| | | | |
|---|---|---|---|
| **1.** | b | **2.** | b |
| **3.** | d | **4.** | c |
| **5.** | d | **6.** | d |
| **7.** | a | **8.** | d |
| **9.** | b | **10.** | d |
| **11.** | False | **12.** | c |
| **13.** | b | **14.** | a |
| **15.** | c | **16.** | True |
| **17.** | a | **18.** | False |
| **19.** | b | **20.** | d |

***END...***

UNIT - II

# 4 Software Requirements

## 4.1 : Functional and Non Functional Requirements

**Q.1 Explain the terms requirements and requirement engineering.** [JNTU : Part A, Marks 2]

**Ans. : Requirement :** A requirement can range from a high-level abstract statement of a service or of a system constraint to a detailed mathematical functional specification.

**Requirement Engineering :** *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.

**Q.2 What are the types of requirements ?** [JNTU : Part A, Marks 2]

**Ans. :** • **There are two types of requirements -** Functional and Non functional requirements.

- **Functional requirements -** The functional requirements describe the required functionality of the system as well as system services.
- **Non functional requirements -** The non functional requirements define the **system properties and constraints.** Various properties of requirements are - Reliability, response time ,storage requirements. The constraints are - input and output device capability, system representation etc.

**Q.3 Why it is so difficult to gain a clear understanding of what customer wants ?** [JNTU : Part B, Marks 5]

**Ans. :** Following are the reasons for : why it is difficult to understand customer wants -

1. Customer sometimes is unable to specify the **scope of the project**. Sometimes customers specify too many technical details and this may increase the confusion.
2. There is difficulty in **understanding the problem**. Sometimes customer could not decide what are their *needs and wants*. Sometimes they have got poor understanding of capabilities and limitations the existing computing environment.

   Sometimes customers find it difficult to communicate with the system engineer about their needs. Sometimes customers may have got some conflicting requirements. This ultimately results in specifying ambiguous requirements.
3. As project progresses the needs or requirements of the customers changes. This creates a problem of volatility.

**Q.4 Enlist various functional and Non functional requirements for the bank ATM system.** [JNTU Part B, Marks 5]

**Ans. : 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.

**Q.5 Describe functional and non functional requirements of software.**

☞[JNTU : Part A, Dec.-19, Marks 5 Part B, May-13, Marks 3]

**Ans. : Functional requirements :**

- 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.

**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.

**Non functional requirements :**

- 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.

**Q.6 Enlist the metrics used for specifying the non functional requirements.** ☞[JNTU : Part A, Marks 2]

**Ans. :**

| Property | Metric |
|---|---|
| Speed | Events per response time processed transactions per second. |
| Size | Kilo bytes. |
| 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. |

**Q.7 Distinguish between functional an non functional requirements.**

☞[JNTU : Part B, Dec.-10, Marks 6, Dec.-16, Marks 5]

**Ans. :**

| 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. |

**Q.8 What are domain requirements ? Explain domain requirements for the library system.**
**[JNTU : Part B, Dec.-10, Marks 16]**

**Ans. :** • 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.
- 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.

**Q.9 What is non functional requirement ? Give any four example of it.**
**[JNTU : Part B, April-11, Marks 8]**

**Or Discuss the classification of non functional requirements.** **[JNTU : Part B, Dec.-12, Marks 7]**

**Ans. : Classification :** The classification of non functional requirements is as given below. Refer Q.9.1.

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

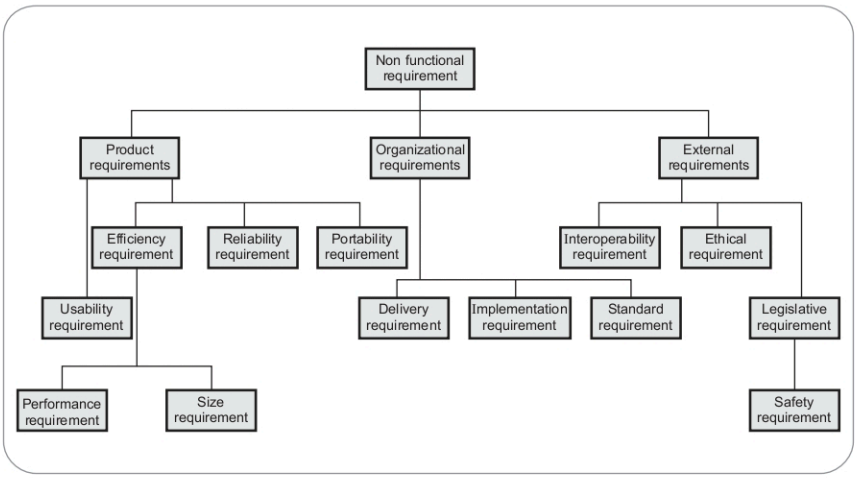



**Fig. Q.9.1 Types of non functional requirement**

**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.

**Examples of non functional requirement :** Refer Q.4.

**Q.10 Explain the problems with non functional requirements.** [JNTU : Part B, May-07, Marks 7]

**Ans. :** Following are the problems associated with non functional requirements -

(1) As the non functional requirements represent the properties and constraints of the system, these are not directly related to specific functionality of the system.

(2) The non functional requirements can not be identified easily as these requirements are dependent upon more than one services of the system.

(3) Even-though the functional requirements are fulfilled by the system it is essential to achieve some essential non functional requirements of the system, otherwise system becomes useless. For example - if the online banking system is not a secured system then it is of no use.

(4) Sometimes non functional requirements are costly to implement.

(5) Non functional requirements need to be expressed as quantitative requirements in order to test the system goals but translating the non functional requirements in quantitative requirements is difficult. Hence it is not possible to test the non functional requirement.

## 4.2 : User Requirements

**Q.11 What do you mean by user requirements ?** [JNTU : Part A, Marks 2]

**Ans. :** • The user requirements should describe functional and non functional requirements in such a way that they are understandable by system users who don't have detailed technical knowledge.

• User requirements are defined using natural language, tables and diagrams because these are the representations that can be understood by all users.

**Q.12 Discuss the problems of using natural language for defined user and system requirements.** [JNTU : Part B, May-09, Marks 8]

**Ans. :** Various problems that can arise in the requirement specifications when requirements are given in natural language -

**Lack of clarity :** Sometimes requirements are given in ambiguous manner. It is expected that text should help in clear and precise understanding of the requirements.

**Requirements confusion :** There may be confusion in functional requirements and non functional requirements, system goals and design information.

**Requirements mixture :** There may be a chance of specifying several requirements together as a single requirement.

## 4.3 : System Requirements

**Q.13 What are system requirements ?** [JNTU : Part A, Marks 2]

**Ans. :** • System requirements are more detailed specifications of system functions, services and constraints than user requirements.

• The system requirements can be expressed using system models.

• The requirements specify what the system does and design specifies how it does.

• System requirement should simply describe the external behavior of the system and its operational constraints. They should not be concerned with how the system should be designed or implemented.

**Q.14 Explain in detail the structured language specification.** [JNTU : Part B, April-11, Marks 8]

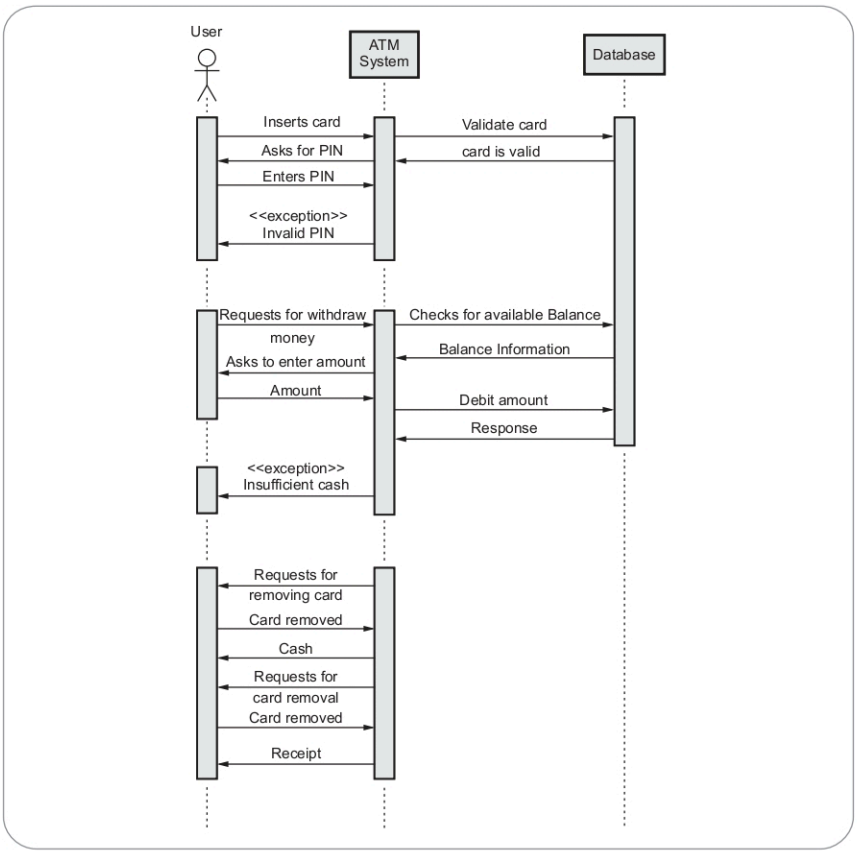



**Fig. Q.14.1 Sequence diagram of ATM withdrawal**

**Ans. :** • All the requirements should be written in a **standard way** while using structured language specification.

- The **advantage** of specifying requirements using this method is that requirement become **understandable** and **expressive.**
- The only necessary thing while writing requirements using natural language is that some degree of **uniformity must be maintained**.
- Extra information can be added when the requirements are written using natural language. This information can be represented using tables or graphical models.
- One way of using graphical model is use of **sequence diagram**.
- The sequence diagram represents the sequence of actions that user performs while interacting the system.

- Example : Following is a sequence diagram for withdrawal of cash from ATM. (See Fig. Q.14.1 on previous page)

**Q.15 Why requirement and design are inseparable ?** [JNTU : Part A, Marks 3]

**Ans. :** • A system architecture may be designed to structure the requirements.

- The system may inter-operate with other systems and that may generate design requirements.
- The use of a specific design may be a domain requirement.

## 4.4 : Interface Specification

**Q.16 Explain the term - system interface.** [JNTU : Part A, Marks 2]

**Ans. :** Sometimes there is already existing system which can be used with the newly created software system. This conjunction of old system with new system is called system interface.

**Q.17 Enlist the three types of interfaces that can be defined.** [JNTU : Part A, Marks 3]

**Ans. : 1. Procedural interfaces :** These are popularly known as **A**pplication **P**rogramming **I**nterfaces (API). Such procedures are intended to offer services that may be used by calling procedures.

2. **Data structures :** Data structures are the descriptors of data. They play an important role in organization of data for given algorithm. The data structures can be passed from one sub-system to another.
3. **Representation of data :** This level of specification is used in certain programming languages like ADA. For real time applications these kind of interfaces are often useful. Sometime to describe this interface diagrams can be used.

## 4.5 : The Software Requirements Document

**Q.18 What is software requirements document or Software Requirement Specification (SRS) ?** [JNTU : Part A, Marks 2]

**Ans. :** The software requirements document is the specification of the system. It should include both a definition and a specification of requirements.

**Q.19 Give the reasons why quantitative requirements specification is difficult in practice ?** [JNTU : JNTU : Part A, Dec.-10, CSE/IT/CS & SE, Marks 4]

**Ans. :** Following are some reasons why quantitative requirements specification is difficult in practice -

1. Customer may find it practically difficult to translate the system goals into quantitative requirements. For example if the goal is **maintainability** then there is no metric to translate it into quantitative measure.
2. Even though some goals are represented in quantitative metric for a customer it becomes difficult to understand that number. For instance: if reliability is 4 then customer may find it difficult to understand what the number 4 stands for in regards to reliability.

**Q.19 List the characteristics of good SRS.** [JNTU : Part A, Marks 2]

**Ans. :** Following are important characteristics of good SRS -

(1) SRS must be correct.

(2) SRS must be unambiguous.

(3) SRS must be complete.

(4) SRS must be consistent.

(5) SRS must be traceable.

**Q.20 Write significance and structure of software requirements document.** [JNTU : Part B, Dec.-13, Marks 8, Dec.-17, Marks 5]

**Ans. :**

**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.

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.

**4.1 User Interfaces**

Describes how this product interfaces with the user.

**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**

**... 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.

**Q.21 Describe desirable characteristics of a good software requirement specification document. What is the role of SRS in Software Engineering ?**

☞ **[JNTU : Dec.-19, Marks 5]**

**Ans. :** Refer Q.18 and Q.19.

***END...***

UNIT - II

# 5 Requirements Engineering Process

## 5.1 : Requirement Engineering Process

**Q.1 What are the goals of Requirement Engineering process ?** ☞[JNTU : Part A, Marks 2]

**Ans. :** The goal of requirement engineering process is to create and maintain system requirement specification document.

**Q.2 What is requirement engineering process ?** ☞[JNTU : Part A, Marks 2]

**Ans. :** Requirement engineering process is a process in which various activities such as discovery, analysis and validation of system requirements are done.

**Q.3 With the help of neat diagram depict the requirement engineering process activities.** ☞[JNTU : Part B, May-09, Feb.-10, Marks 6]

**OR Discuss various steps in requirements engineering. What are the work products of engineering the requirements ?** ☞[JNTU : Dec.-19, Marks 5]

**Ans. :** • It begins with feasibility study of the system and ends up with requirement validation. Finally the requirement document has to be prepared.

- This process is a three stage activity where the activities are arranged in the iterative manner. In the early stage of this process most of the time is spent on understanding the system by understanding the high-level non functional requirements and user requirements. The spiral model of requirement engineering process is as shown in Fig. Q.3.1. (See Fig. Q.3.1 on next page)
- The generic activities that are common to all processes are
  1. Requirements elicitation
  2. Requirements analysis
  3. Requirements validation
  4. Requirements management
- Requirements engineering process can also be viewed as **structured analysis method** in which the system is analysed fully some **system models** are prepared. Particularly use cases are developed which help in exposing the functionalities of the systems.
- Along with creation of system models some **additional information** is also provided in the requirement engineering process.

## 5.2 : Feasibility Studies

**Q.4 Define - Feasibility study.** ☞ [JNTU : Part A, Marks 2]

**Ans. :** A feasibility study is a study made to decide whether or not the proposed system is worthwhile.

**Q.5 Discuss software scope and feasibility study.** ☞ [JNTU : Dec.-19, Marks 2]

**Ans. : Software scope :** The software scope clearly defines all functionalities and artifacts to be delivered as a part of the software. The scope identifies what the product will do and what it will not do, what the end product will contain and what it will not contain.

**Feasibility study :** Refer Q.4.

**Q.6 What is the significance of feasibility study ?** ☞[JNTU : Dec.-16, Marks 3]

**Ans. :** The feasibility study is an important activity is requirement engineering process because – (1) it checks if the proposed system contributes to organizational objectives (2) it checks if the system is within the given budget. (3) if the system can be implemented with other useful systems.

Thus feasibility study is made to decide whether or not the proposed system is worthwhile.

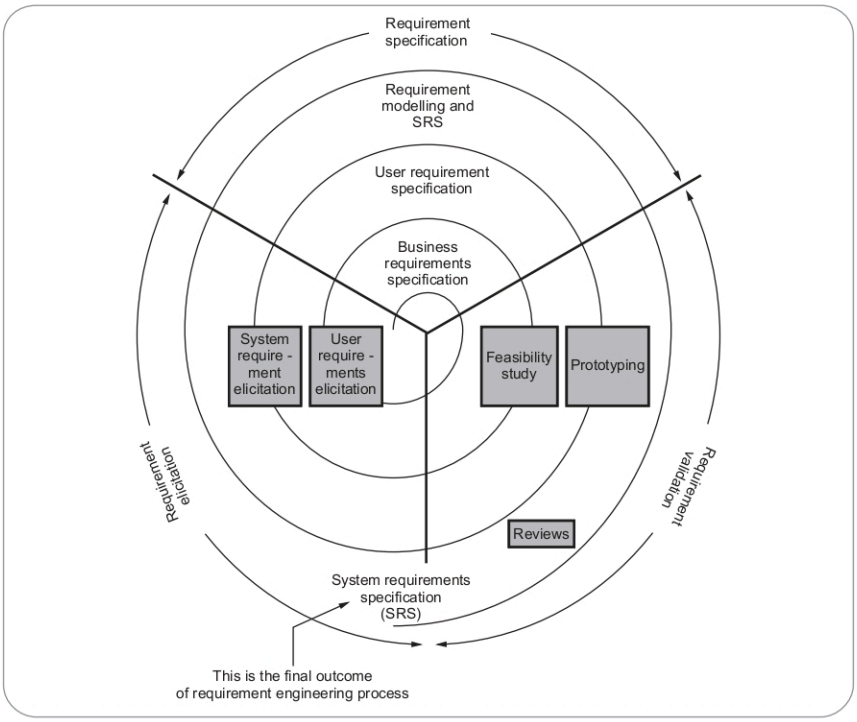



**Fig. Q.3.1 Spiral model of requirements engineering process**

**Q.7 Explain about feasibility studies.**
☞ [JNTU : Part B, Dec.-10, Marks 6, Dec.-17, Marks 5]

**OR What is meant by feasibility study ? Discuss its importance and discuss various types of feasibility studies that may performed on software project.** ☞ [JNTU : Part B, Dec.-13, Marks 8]

**Ans. : Definition -** Refer Q.4.

**Importance -** Refer Q.6.

**Types of Feasibility Study -**

**1. Technical Feasibility :** The technical feasibility is the study of configuration of the system.

Using this study a tender document can be prepared. Then the manufacturer or dealer can provide the equipments based on desired logical needs.

The technical feasibility study is supposed to be the most important and difficult study.

**2. Operational Feasibility :** The operational feasibility is based on the human factors and political aspects. The operational feasibility can be performed by answering following questions,

1. What change will be brought with the system ?
2. What are the factors that are disturbing organizational structure ?
3. Which are the new skills that are required for improvement in operations ?

**3. Economic Feasibility :** This kind of feasibility study is done for cost or benefit analysis. In this study the benefits of the proposed system are identified and the corresponding costs are determined.

**4. Management Feasibility :** The management feasibility means checking whether the management will accept the proposed project or not. If the top level management does not agree upon the project idea then it is considered as a non feasible project.

**5. Legal Feasibility :** The legal feasibility means finding out whether the proposed project is legally acceptable or not.

**6. Time Feasibility :** The time feasibility means identifying whether the proposed project will be completed within the stipulated time or not. If the projects runs for a long time then it is considered to be infeasible project.

**7. Social Feasibility :** Whether the project will be accepted by the people or not is called a social feasibility.

## 5.3 : Requirements Elicitation and Analysis

**Q.8 What are the activities of requirement elicitation and analysis ? Explain.**

☞[JNTU : Part B, Dec.-16, Marks 5]

**Ans. :** The spiral model as given below depicts the requirement elicitation and analysis process.

The process activities are -

**1. Requirement discovery :** By having effective communication with the customers the requirements can be identified.

**2. Requirements classification and discovery :** All the unstructured requirements can be categorized systematically depending upon their nature. And they are arranged in groups.

**3. Requirement prioritization :** There are some conflicting requirements. Hence the requirements are prioritized first. If there are some unrealistic requirements then negotiations are made and only

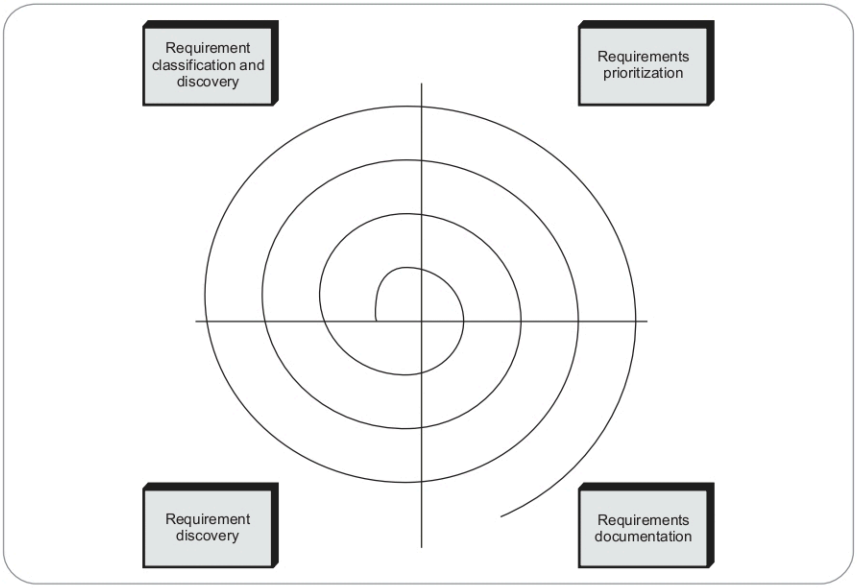



**Fig. Q.8.1 Requirements elicitation and analysis process**

realistic prioritized requirements are collected. If any conflict occurs then it its resolved by requirement engineers.

**4. Requirement documentation :** This is the specification of all the requirements. And an important requirement document is created.

**Q.9 What is the meaning of stakeholders in software engineering ?** [JNTU : Part A, Marks 2]

**Ans. :** Stakeholders are the human entities that take active participation during project development stages along with the software developers. It includes end users, mangers, engineers involved in maintenance, domain experts, trade unions etc.

**Q.10 Discuss the statement, "requirements elicitation is a human-centered activity".** [JNTU : Part B, Dec.-10, Marks 10]

**Ans. :** After performing feasibility study the requirements elicitation and analysis can be done. Requirement **elicitation means discovery** of all possible requirements. After identifying all possible requirements the analysis on these requirements can be done. Software engineers communicate the end-users or customers in order to find out certain information such as : application domain, expected services from the system, the expected performance level of the system. From this information even constraints of the system can be decided.

This requirement elicitation is a human centered activity.

**Q.11 Discuss the need for requirements prioritization and negotiation.** [JNTU : Part B, April-11, Marks 8]

**Ans. :** Refer Q.8.

**Q.12 Define a scenario. Write a sample use case scenario for an article downloading in the library system.** [JNTU : Part B, May-09, Marks 8]

**OR Discuss various steps in requirements engineering. What are the work products of engineering the requirements ?** [JNTU : Part B, May-09, Marks 8]

**Ans. :** Scenario is the sequence of interactions made by the user with the software systems. Requirements engineers can use the information gained from this scenarios to formulate actual system requirements. The scenarios describe the interaction sessions.

**Q.13 How scenarios are useful for describing requirements ?** [JNTU : Part B, Dec.-10, Marks 8]

**Ans. :** • Each scenario covers one or more interaction sessions.

• Each scenario includes -

- A description of what the system and the users want at the beginning of the scenario
- A description of normal flow of events
- A description of something that went wrong and handling of it
- A description of other activities occurring in parallel
- A description of finish state

**Q.14 Write short note on interviewing technique.** [JNTU : Part B, May-09,12, Marks 8 ]

**Ans. :** This is an effective method of requirement gathering. The requirement engineering team communicates the stakeholders by asking them various questions about the system and its use. From the answers the requirements can be identified. There are two types of interviews.

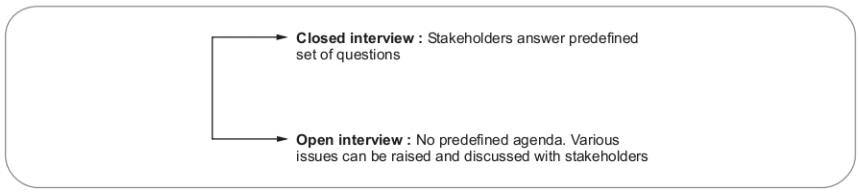



**Fig. Q.14.1 Types of interview**

Interviews are useful for understanding stakeholders but they are not much useful for understanding the application domain.

**Characteristics of effective interviews**

1. The interviews should be conducted in a free environment and they should be conducted with open minded approach. The requirement engineers should listen to stakeholders with patience. Similarly if stakeholders are expecting some unrealistic things about the system they should be ready to change their mind and ideas about the system.
2. Interviewee should start the discussion by asking questions and the requirements should be gathered together.

**Q.15 What is meant by viewpoint ?**

☞[JNTU : Part A, Marks 2]

**Ans. :** Viewpoint provides perspective to the system and using these perspective the requirements can be discovered.

**Q.16 Explain the viewpoint oriented techniques to requirement discovery for a library system along with various types of viewpoint.**

☞[JNTU : Part B, Dec.-10, Marks 16]

**Ans. :** The viewpoint is also useful in classifying the stakeholders. Following are the types of viewpoints -

- **Interactor viewpoint :** This viewpoint is useful for finding the interaction one system with other system. For example in ATM system the user of ATM is the interactor for the bank.
- **Indirect viewpoints :** The user who is not using the system directly but its existence reflects on the requirements of the system. Such stakeholders form indirect viewpoint.
- **Domain viewpoints :** The domain characteristics and constraints that affect on requirements of the system sets the domain viewpoints. For example in Library system the rules that are to be followed for reserving the book(the fine or dues should be paid already or book must be returned within 1 week etc.) form the domain view point.
- From these viewpoints the requirements can be discovered. From interactor viewpoint the system requirements can be obtained. The indirect viewpoints help in getting the organizational requirements and constraints. The domain requirements provide domain constructs that need to be applied for the system.
- View points in library management system is given by following Fig. Q.16.1.

**Q.17 What is meant by method based analysis ? Explain the VORD process model.**

☞[JNTU : Part B, Dec.-14, Marks 8]

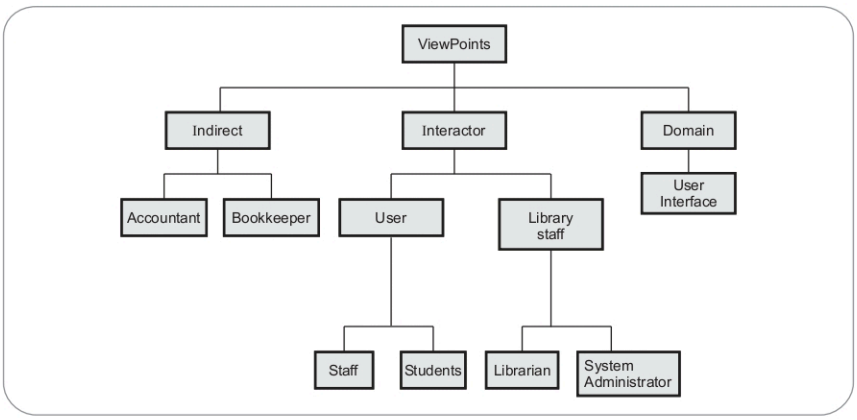



**Fig. Q.16.1 Viewpoints for library system**

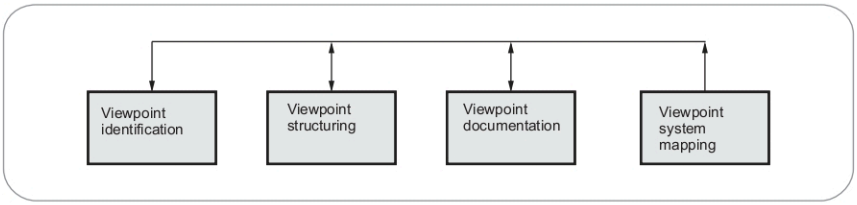



**Fig. Q.17.1 Stages of VORD**

**Ans. : Method Based Analysis :** The requirements analysis depends upon the application of structured methods in order to understand the system. Some methods are designed for requirement elicitation and others are for design methods. The VORD is an example of method based analysis used for requirement analysis and elicitation.

**VORD -** The Viewpoint Oriented Requirement Definition (VORD) is a service oriented definition for supporting requirement analysis and elicitation. The principle stages of VORD are - Refer Fig. Q.17.1.

**Viewpoint identification :** Viewpoint identification is a stage in which viewpoints to which the system services are received are identified. Similarly the services received to these viewpoints must be identified.

**Viewpoint structuring :** All the related viewpoints are grouped into the hierarchy. Common services are denoted by the top level of this hierarchy.

**Viewpoint documentation :** The identified viewpoints and services are described in detail.

**Viewpoint system mapping :** The transformation of system analysis is done into the object oriented design.

For example : Viewpoint template for ATM

| Viewpoint template | Service template |
|---|---|
| **Reference :** Account holder | **Reference :** Balance enquiry |
| **Attributes :** Account number PIN | **Rationale :** To allow the user to check the amount of money present in his account. |
| **Events :** Enters the account No. and PIN Starts transaction<br>Select service Cancel service<br>Performs the transaction<br>Close transaction | **Specification :** The user selects this service by pressing the balance enquiry button. After knowing the amount present in his account, he is then asked to get printout of mini statement. Then user select the service accordingly. |
| **Services :** Customer withdraw<br>Balance enquiry | **Viewpoints :** Account holder |
| **Sub-VPs :** Joint account holder | **Sub-VPs :** |

**Q.18 Define use cases. Explain their purpose.**
**[JNTU : Part B, May-07, Marks 7]**

**Ans. :** • Use cases are the fundamental units of modeling language, in which functionalities are distinctly presented.

- The use case is a scenario based technique.
- Use cases help to identify individual interactions with the system.
- Use-cases are extensively used for requirements elicitation.
- By designing the proper use cases for different scenarios major requirements of the system can be identified.

- The typical notations used in the use cases are

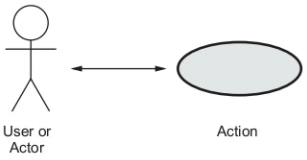



Fig. Q.18.1 Use-case notation

**Q.19 Using your own knowledge of how an ATM is used, develop a set of use cases that could be used to derive the requirements for an ATM system.** [JNTU : Part B, May-08, Marks 7]

**Ans. :** The use case for ATM system is as shown below.

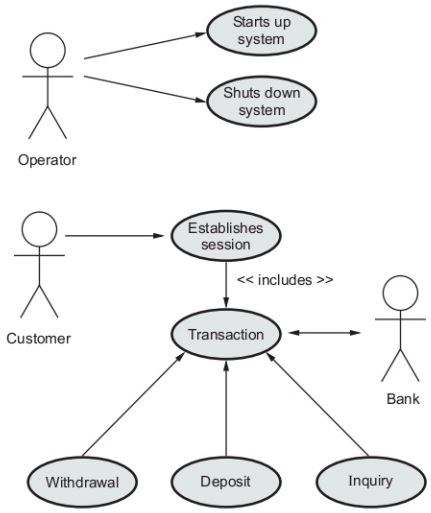



Fig. Q.19.1 Use cases for ATM system

Flow of events can be described as

**System start up :** The system is started when the operator turns on the switch. Initially the operator has to enter some amount of money in the cash dispenser. The connection with the bank gets established. Then only the servicing customer can begin.

**System shutdown :** The operator can shutdown the system only after confirming that there is no customer currently operating the ATM system. Then the connection to the bank gets disconnected.

**Session :** This use case starts when the user inserts the card. ATM pulls the card inside and reads it. Then further inquiry information is displayed on the display panel .

**Transaction :** The transaction use case is comprised of amount withdrawal, deposit and inquiry.

**Q.20 Define Ethnography. What are the requirements that are associated with ethnography.** [JNTU : Part B, May-09, Dec.-10, Marks 8]

**Ans. :** Definition: Ethnography is a technique of observation which is used to understand social and organizational requirements.

Two types of requirements can be discovered by ethnography.

**i. Requirements obtained from working style of people :** These are the requirements that can be identified from the sequence of actions that a user is performing. For example in ATM system when the user enters the PIN, the card validity action takes place. Unless and until the card gets validated there should not be any transaction processing. That means, it is required that a valid card, valid PIN must be entered for getting the money from ATM.

**ii. Requirements obtained from inter-activities performed by the people :** Sometimes for finding the social requirements the other people's activities should be known. For example in ATM system the operator can not shutdown the system if some transaction is in processing.

**Q.21 Is ethnography combined with prototyping? Discuss the statement.** [JNTU : Part B, May-09, Dec.-10, Marks 8]

**Ans. :** Yes, ethnography can be combined with prototyping. Following model represents the ethnographic process.

- The ethnography informs the development of prototype using this information prototype refinement cycles can be used.

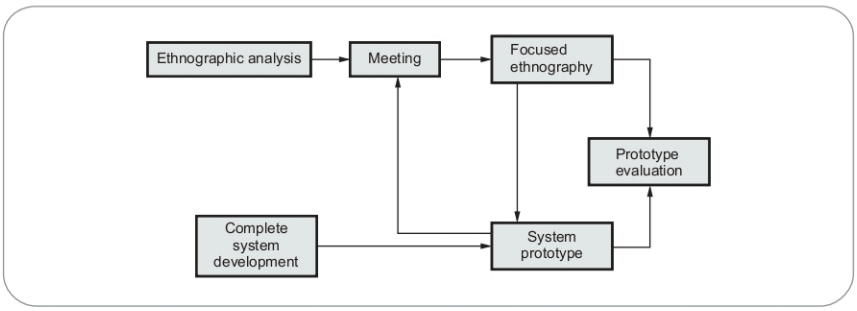



**Fig. Q.21.1 Ethnographic process**

- Prototype focuses on ethnography by identifying problems and questions discussed by ethnographer. (Refer Fig. Q.21.1.)

## 5.4 : Requirements Validation

**Q.22 What is requirements validation ?**

**[JNTU : Part A, Marks 2]**

**Ans. :** Requirement validation is a process in which it is checked whether the gathered requirements represent the same system that customer really wants.

**Q.23 State and explain various requirements validation techniques. Also discuss requirements reviews.** **[JNTU : Part B, Dec.-12, 14, Marks 8]**

**Ans. :** Refer Fig. Q.23.1.

**1. Requirements reviews :** Requirement review is a systematic manual analysis of the requirements.

- The requirement review should be taken only after formulation of requirement definition. And both the customer and contractor staff should be involved in reviews.
- Reviews may be formal (with completed documents) or informal.
- Good communications should take place between developers, customers and users. Such a healthy communication helps to resolve problems at an early stage.

**2. Prototyping :** The requirements can be checked using executable model of system.

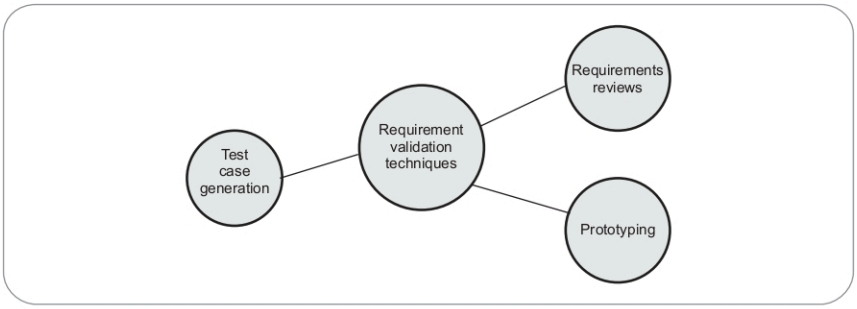



**Fig. Q.23.1 Requirement validation technique**

**3. Test-case generation :** In this technique, the various tests are developed for requirements. The requirement check can be carried out with

- **Verifiability :** Is the requirement realistically testable ?
- **Comprehensibility :** Is the requirement properly understood ?
- **Traceability :** Is the origin of the requirement clearly stated ?
- **Adaptability :** Can the requirement be changed without a large impact on other requirements ?

## 5.5 : Requirement Management

**Q.24 What is requirement management ?** [JNTU : Part A, Marks 2]

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

**Q.25 Why requirements get changed ?** [JNTU : Part A, Marks 3]

**Ans. :** • 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.

**Q.26 How do you validate and manage requirements ?** [JNTU : Part B, May-13, Marks 8]

**OR Elaborate on requirements management planning in detail.** [JNTU : Part B, May-12, Marks 8]

**Ans. : Requirement Validation -** Refer Q.22, Q.23.

**Requirement Management -** Refer Q.24, Q.25.

**Requirement Management process -** This process is carried out in following steps -

1. **Requirement Identification -** Each requirement is individually identified.
2. **Change Management Process -** While analyzing the requirement change the process plan is followed.
3. **Apply traceability support -** During the traceability of requirements, the amount of information about requirement relationship is maintained.
4. **Case Tool Support -** The automated case tools are used to manage requirement changes.

**Q.27 What is traceability ? Explain different types of traceability.** [JNTU : Part B, Sept.-07, Marks 8]

**Ans. :** Traceability is concerned with relationship between requirements their sources and the system design. Using traceability the requirement finding becomes easy.

Various types of traceability are

1. **Source traceability :** These are basically the links from requirement to stakeholders who propose these requirements.
2. **Requirements traceability :** These are the links between dependent requirements.
3. **Design traceability :** These are the links from requirements to design.

- Traceability information is typically represented by a data structure **Traceability matrix.** If one requirement is dependant upon the other requirement then in that row-column cell 'D' is mentioned and if there is a weak relationship between the requirements then corresponding entry can be denoted by 'R'.

For example

| Requirement ID | A | B | C | D | E | F |
|---|---|---|---|---|---|---|
| A | | D | | | R | |
| B | | | D | | | |
| C | | | | R | | |
| D | | | D | | | R |
| E | | | | | | |
| F | R | | | D | | |

For mentioning the traceability of small systems usually the traceability matrix is maintained.

**Q.28 Differentiate between enduring and volatile requirements. Give examples for each.**
[JNTU : Part B, Sept.-07, Marks 07]

**Ans. : Eduring requirements :** These are the stable requirements that are derived from the core activity of the organization. These requirements are dependent upon the application domain of the software. For example: For banking system, transfer of money from one account to another is the enduring requirement.

**Volatile requirements :** For certain requirements if there is a possibility that those requirements may get changed during the development stage or after the system becomes operational, then such requirements are called volatile requirements.

For example - For hospital management system, health care policies may get changed and hence associated requirements will change.

**Q.29 What are mutable requirements ? Differentiate between mutable and consequential requirements.** [JNTU : Part B, Dec.-10, Marks 10]

**Ans. : 1. Mutable Requirement -** If due to change in the environment, if the requirements get changed then such requirements are called as mutable requirements. For example : In the library management system, if the traditional library is turned into a digital library (containing e-books, on-line articles, e-learning or video conferencing facilities) then requirements for the library automation software will be changed.

**2. Consequential requirements -** Requirements that gets changed due to introduction of computer based systems, such requirements are called consequential requirements.

For example : In a tours and travel agencies, due to on-line ticket booking facilities requirements get changed. Such changed requirements are consequential requirements.

**Q.30 Discuss about emergent requirements.**
[JNTU : Part A, Marks 3]

**Ans. :** Due to customer's understanding of the system during the development stage certain requirements may get changed. Such types of requirements are called emergent requirements.

**Q.31 Explain about requirement change management in detail.**
[JNTU : Part B, Dec.-11, Marks 8]

**Ans. :** • The requirement change management is a technique that can be applied to the processes in which requirements may get changed.

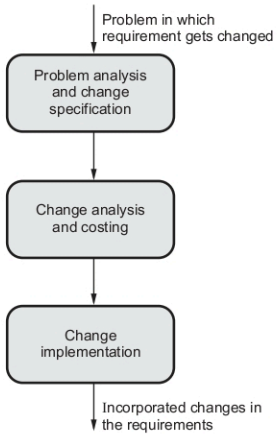



**Fig. Q.31.1 Requirement change management process**

• The need for requirement change management is that even though the changes are made consistently in the requirements it is possible to incorporate those changes in a controlled manner. The requirement change management process can be applied in three stages.

**1. Problem analysis and change specification :** When requirement change request is made for some particular problem then the problem with older requirement is mentioned or sometimes simply change specification is given. Then first of all, problem analysis or change specification is analysed in order to **validate** the required change. If necessary the feedback of this analysis is given to the person who is demanding such change.

**2. Change analysis and costing :** Following actions are carried out in this stage :

i) The effect of change is assessed using traceability information.

ii) The cost of such change is estimated.

iii) After getting the cost of changes the decision is made on whether to go for implementation of these changes or not.

**3. Change implementation :** Once it is decided to implement the proposed changes in the requirement, the requirement document has to be modified. The requirement document has to either re-written or re-organised. This can be achieved by making the modularity in the requirement specifications, so that it becomes easy to change individual section without affecting other part of requirement document.

***END...***

UNIT - II

# 6 System Models

## 6.1 : Context Model

**Q.1 What is system Model ?**

[JNTU : Part A, Marks 2]

**Ans. :** • The system model is a **graphical representation** that is used to describe various processes of the system, the type of input and output of the system.

• These system models not only specify the user requirements but they also serve as an important element in **analysis and design phase.**

**Q.2 What are the three different perspectives of developing the system models ?**

**Ans. :** We can develop the system model using different perspectives and use of these perspectives gives the categorization of system models into different models -

(1) Using External Perspective context model of the system can be developed.

(2) Using Behavioral perspective behavioral model of the system can be developed.

(3) Using Structural perspective data model of the system can be developed.

**Q.3 What is context model ?**

**Ans. :** Context model is a graphical representation of the system in which the system boundaries are specified. This is the model which represents the system environment in which system is working.

**Q.4 Write the purpose of context model ?**

[JNTU : Part A, Dec.-16, Marks 2]

**Ans. :** (1) The context models are the architectural models that show the working environment of the system. The context models also shows the relationship that exists with other systems.

(2) All these defined relationships help in finding the requirements of the system.

**Q.5 Explain the context model with the context of an ATM system.**

[JNTU : Part B, Feb.-10, Marks 16, Dec.-12, Marks 8]

**OR What is context model ? Describe the importance of context model.**

[JNTU : Dec.-19, Marks 5]

**Ans. : Definition of context model -** Refer Q.3.

**Context Model of ATM System -**

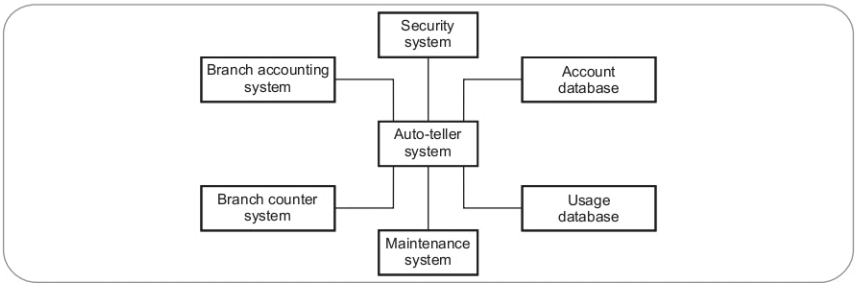



**Fig. Q.5.1 Contex model of ATM system**

## 6.2 : Behavioral Models

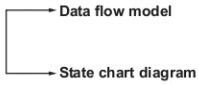
**Q.6 What is behavioral model ? Discuss various types of behavioral models with examples for each.** [JNTU : Part B, May-07, Marks 8]

**Or Explain the need of behavioral models for software development.** [JNTU : Part B, April-11, Dec.-11, Marks 8]

**Or Discuss the role of behavioral models in Software Engineering. Give suitable example.** [JNTU : Part B, Dec.-13, Marks 7]

**Ans. :** • 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.



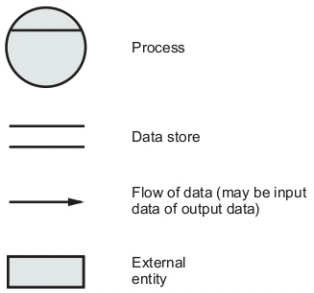

- These models can be used separately on together depending upon nature of application.

**Data Flow Model -** Refer Q.7.

**State Chart Diagram -** Refer Q.10.

**Q.7 Write short notes on - Data flow model.** [JNTU : Part B, Feb.-10, Marks 8]

**Ans. :** • 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.

For example - Refer Fig. Q.9.1. (on next page.)

**Q.8 Discuss the merits and demerits of data flow diagram.** [JNTU : Part B, May-12, Marks 7]

**Ans. : Merits -**

(1) The DFDs define the boundaries of the system.

(2) The DFD provide the detailed representation of system components.

(3) It can be used as the part of system documentation.

(4) DFDs are easy to understand by technical and non-technical users.

(5) It represents the logic and functionalities behind the flow of data within the system.

**Demerits -**

(1) It takes very long time to create.

(2) Physical considerations of the system are not taken into account in case of DFDs.

**Q.9 What is data flow diagram ? Draw data flow diagram for order processing system.** [JNTU : Part B, Dec.-11, Marks 5]

**Ans. : Data Flow Diagram -** Refer Q.7.

**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.

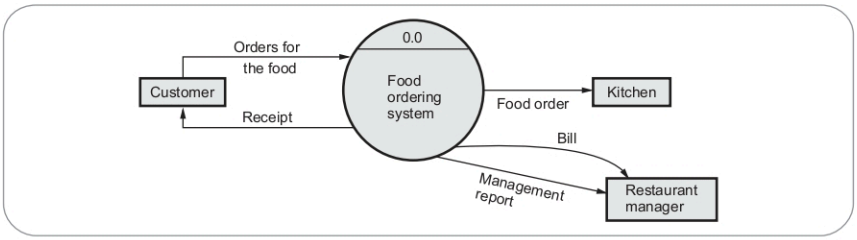



**Fig. Q.9.1 Level 0 DFD (Context level DFD)**

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.

**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.

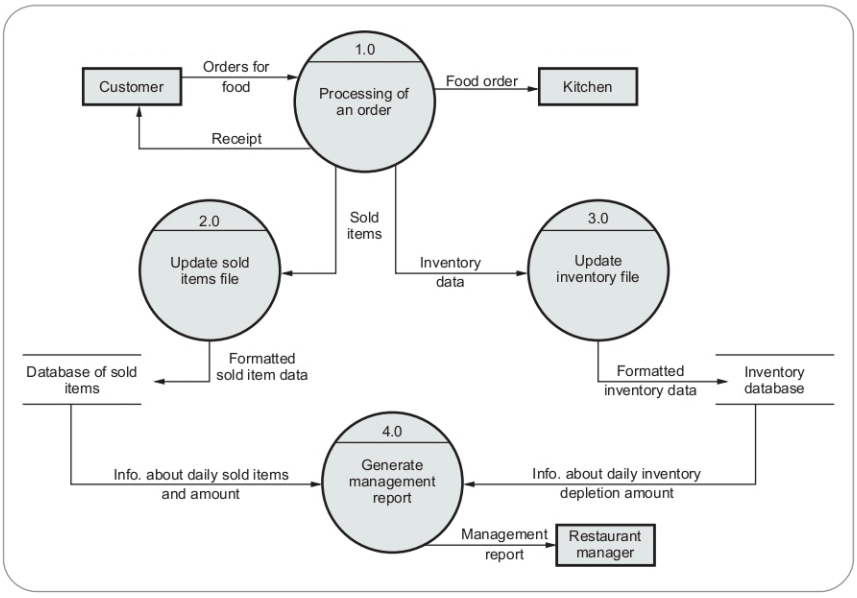



**Fig. Q.9.2 Level 1 DFD**

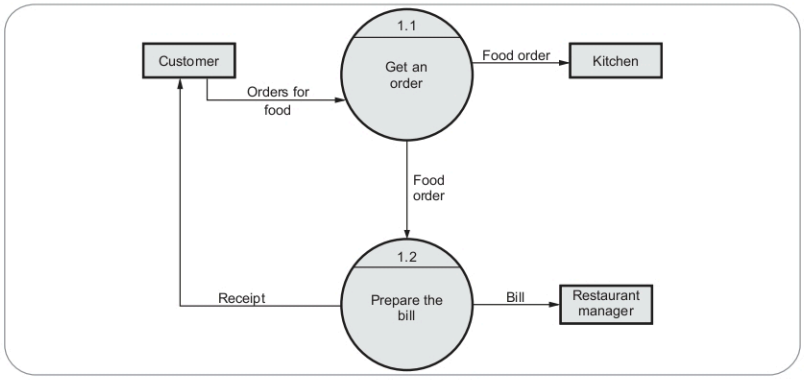



**Fig. Q.9.3 Level 2 DFD**

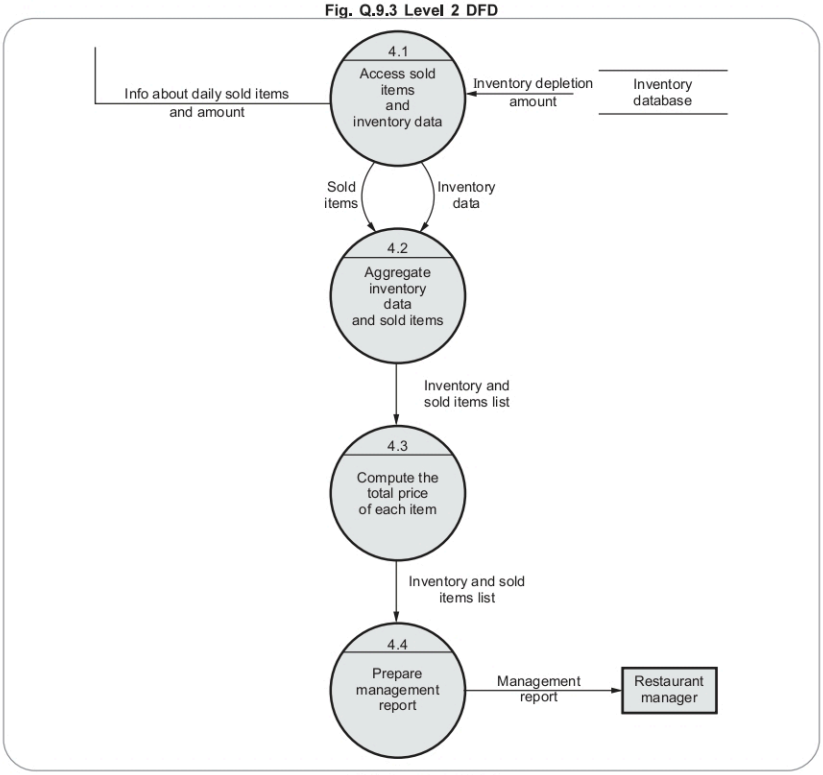



**Fig. Q.9.4 Level 3 DFD**

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

**Level 2 DFD :** "Processing of an order" is shown in detail. (Refer Fig. Q.9.3 on previous page.)

**Level 3 DFD :** In this DFD we will elaborate "Generate management report" activity in more detail. For generating management report we have to access sold items data and inventory data. Then aggregate both solid items data and inventory data. Total price of each item has to be computed. Then from these calculation a management report has to be prepared and given to the restaurant manager. These details can be shown in this DFD - (Refer Fig. Q.9.4 on previous page.)

**Q.10 Write short notes on - State machine model.**
**[JNTU : Part B, Feb.-10, Marks 8]**

**OR Write briefly about the utility of state transition diagram in analysis modelling activity.**
**[JNTU : Part B, Dec.-10, Marks 8]**

**OR What is state machine? Explain the state machine model of simple microwave oven.**
**[JNTU : Part B, April-11, Marks 8]**

**Ans. : Definition :** State machine diagram is a behavior diagram which shows the discrete behavior of the part of designed system. It consists of finite number of states. The machine changes from one state to another state in response to some external inputs; the change from one state to another is called a transition. An state machine is defined by a list of its states, its initial state and the conditions for each transition.

For example : State machine model for simple microwave oven -

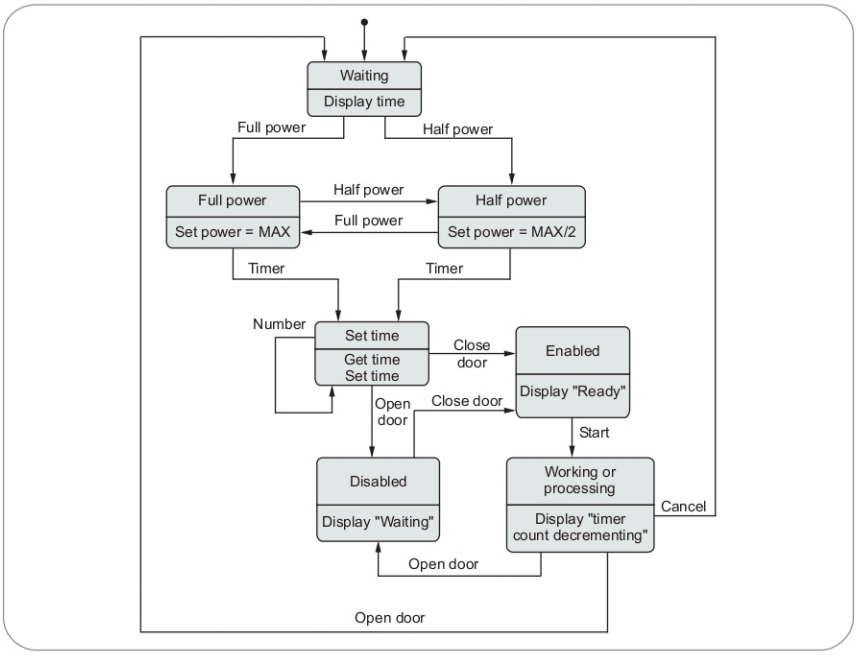



**Fig. Q.10.1 State machine for Microwave oven**

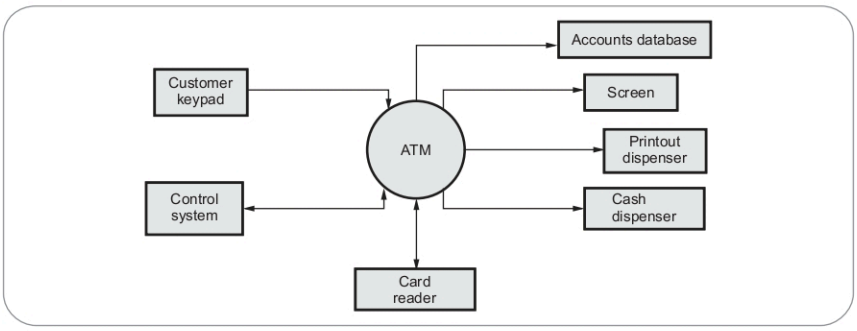



Fig. Q.11.1 Level 0 DFD

**Q.11 Based on your experience with a bank ATM draw a DFD modelling the processing involved when customer withdraws cash from the machine.**
[JNTU : Part B, Marks 5]

**Ans. :** The DFD for ATM system can be drawn with level 0 DFD and level 1 DFD. The level 0 DFD is the context level DFD in which only inputs and outputs that are interacting with the system are given. (Refer Fig. Q.11.1.)

More detailing is done in level 1.

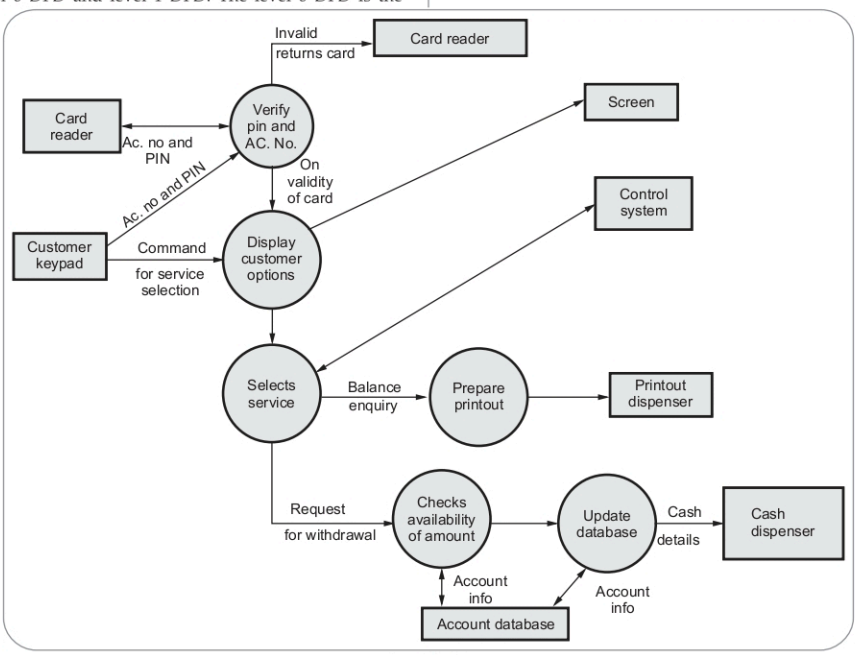



Fig. Q.11.2

**Q.12 Draw the data flow diagram upto level 1, for a 'temperature monitoring system' in an intensive care unit of a hospital.** [JNTU : Part B, Marks 5]

**Ans. :** DFD Level 0

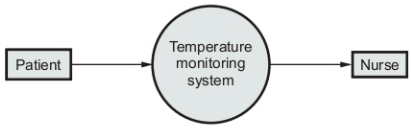



**Fig. Q.12.1 DFD Level 0**

**DFD Level 1 :** Refer Fig. Q.12.2.

## 6.3 : Data Models

**Q.13 What is Data object ?** [JNTU : Part A, Marks 2]

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

**Q.14 Give examples of various data objects.** [JNTU : Part A, Marks 3]

**Ans. : 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

**Q.15 What are attributes ? Also explain various types of attributes.** [JNTU : Part A, Marks 3]

**Ans. :** 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.

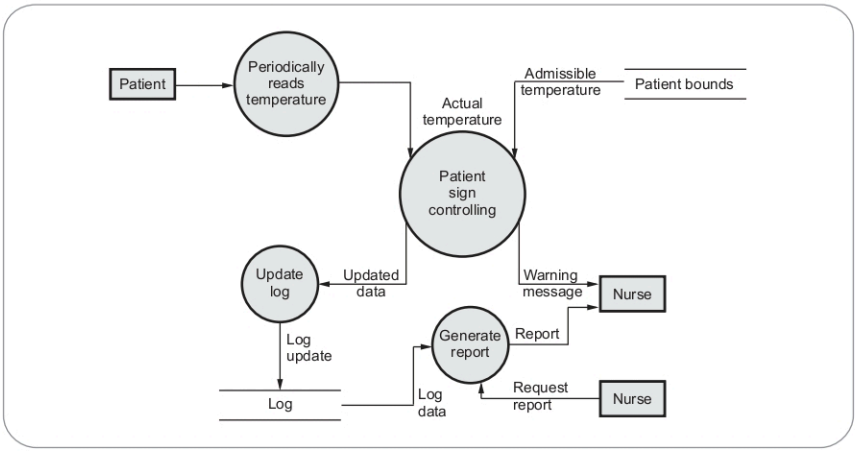



**Fig. Q.12.2 DFD Level 1**

**Q.16 Explain the term - 'Relationship' used in Data modeling.** ☞[JNTU : Part A, Marks 3]

**Ans. :** Relationship represents the connection between the data objects. For example

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


**Fig. Q.16.1 Relationship**

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

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

**Q.17 Define and explain Cardinality and Modality.**

**Ans. : Cardinality :** Cardinality in data modelling, cardinality specifies how the number of occurrences of one object is related to the number of occurrences of another object.

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

**Example**

**Q.18 Discuss in detail the data modelling activity.** ☞[JNTU : Part B, Dec.-10, Marks 8]

**Ans. :** • Data Modeling is a basic step in analysis modeling.

- In data modeling the data objects are examined independently of processing.
- The data model represents how data objects are related with one another.
- During Data modeling ER diagram or Entity Relationship Model is created.
- ER Diagram: Refer Q.19.1.

**Q.19 Explain Entity Relationship Model in detail.** ☞[JNTU : Part B, Marks 5]

**Ans. :** • The object relationship pair can be graphically represented by a diagram called Entity Relationship Diagram (ERD).

- The primary purpose of ERD is to represent the relationship between data objects.
- Various components of ERD are –

**Entity**

- Drawn as a rectangle.
- An entity is an object that exists and is distinguishable.
- Similar to a record in a programming language with attributes.

**Relationship**

- Drawn as a diamond.

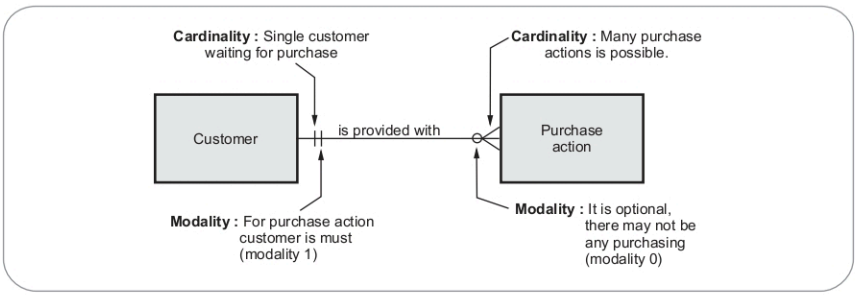



**Fig. Q.17.1 Cardinality and modality**

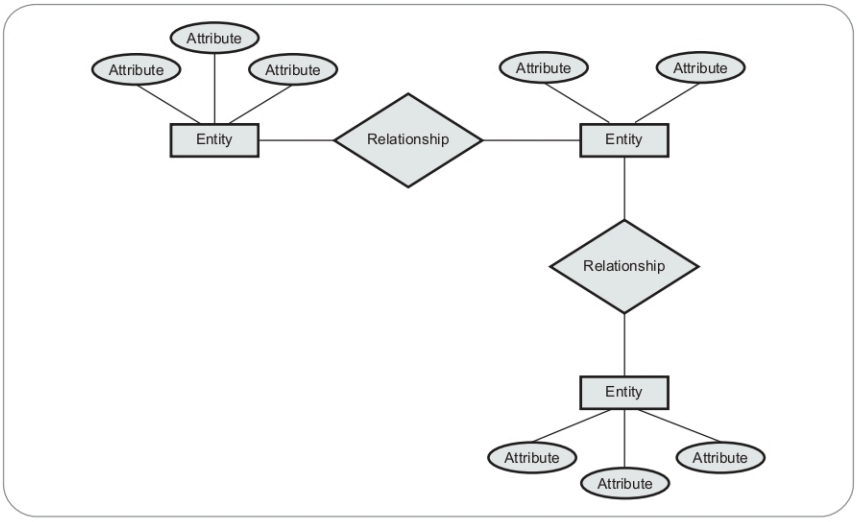



**Fig. Q.19.1 ER diagram**

- An association among several entities.
- Relationships may have attributes.
- Relationships have cardinality (e.g., one-to-many)

**Attribute**

- Drawn as ellipses.
- Similar to record fields in a programming language.
- Each attribute has a set of permitted values, called the domain.
- Primary key attributes may be underlined.

**Example :** Refer Fig. Q.19.1.

**Q.20 Differentiate between data flow model and state machine models.**

☞ **[JNTU : Part B, Dec.-11, Marks 8]**

**Ans. :**

| Sr.No. | Data Flow Model | State Machine Model |
|---|---|---|
| 1. | Data Flow model is the model used for representing the data processing in a system. | State Machine model is the model used for representing system's response to internal or external events. |
| 2. | Data flow modelling uses ER model or DFD. | State machine model uses state-chart diagram. |
| 3. | It is a stateless flowchart. | It is a state-transition diagram. |
| 4. | It represents flow of data in a sequence of processing steps. | It represents the states that are acquired by the object in response to events. |
| 5. | It represents flow of data in the system. | It represent the behavior of object. |
| 6. | The components of data flow model are - processes, data stores. | The components of state machine are states and transitions. |
| 7. | The data flow modeling is applicable for numeric intensive computation of applications. | The state machine model is applicable for modeling the real-time application. |

## 6.4 : Object Models

**Q.21 Write short note on - Object model.**

☞ **[JNTU : Part B, May-13, Dec.-14, 16, 17, Marks 5]**

**Ans. :** • The object model is created for the systems possessing **object oriented programming** approach.

- Generally object models are created for interactive systems. Such systems can then be implemented using object oriented programming languages such as **C++ or JAVA**.
- For the creation of object model during requirement analysis the information about **data** and **its processing** is collected.
- The object model is a natural representation of real world entities such which are required in information processing systems. Such entities can be student, book, customer, Account, Ticket, Car and so on. These entities have their own attributes(such as name, Roll number, ID, Colour, Department and so on). These entities are typically called as **object classes**.
- The system can manipulate these entities by performing some set of operations. Moreover these entities can **interact** with other entities. These all things can be very well represented using object model.
- There are three types of models used to represent object model -
  - Inheritance Model
  - Object Aggregation
  - Object Behavior Model

**1. Inheritance Model :**

- Inheritance is a property by which the child class can inherit some properties(either some attributes or some operations) from the parent class.
- In object oriented modelling some classes are identified and if there are some common attributes and services then inheritance is applied.
- For example : Following Fig. Q.21.1 represents that the person is a base class from which two child classes are inherited Student and Employee.

**2. Object Aggregation :**

- The aggregate relationship is a has a relationship. For example in the following figure Computer is a class which has screen, Memory and Keyboard. That means in aggregate relationship one object can be formed by grouping one or more objects.

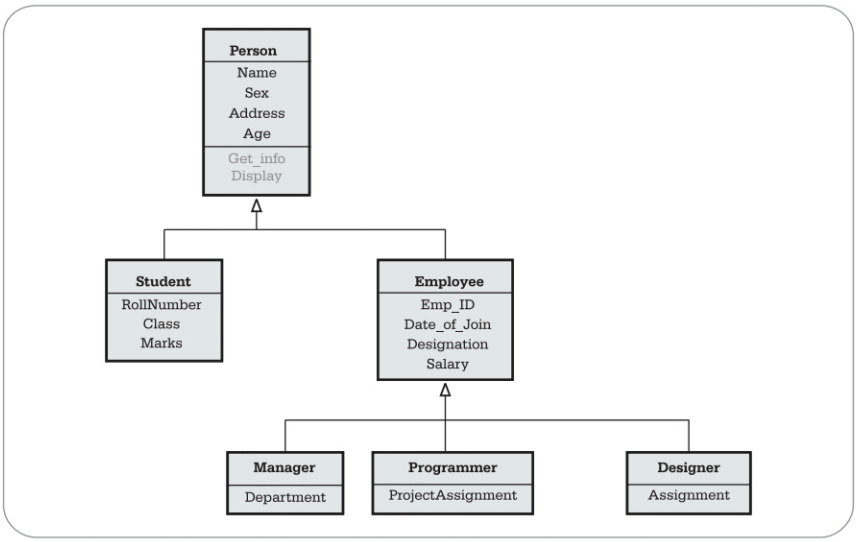



**Fig. Q.21.1 Inheritance hierarchy**

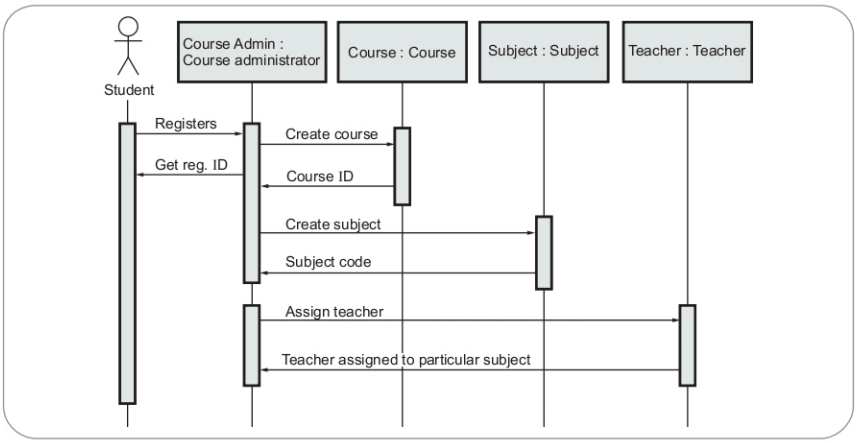



**Fig. Q.21.2 Sequence diagram for student registration**

- The arrow with *diamond* is used to represent the aggregate relationship.

**3. Object Behavior Model :**

- The object behavior model is for representing the sequence of events occurring in the system. Typically it is represented by a Sequence diagram in UML. For example in the above figure the student registration for some course is given. The objects and actors that interact with the system and the events that are taking place are shown by sequence diagram. (Refer Fig. Q.21.2.)

## 6.5 : Structured Methods

**Q.22 Discuss about structured methods.**
☞ **[JNTU : Part B, June-14, Marks 7, Dec.-16,17, Marks 5]**

**Ans. :** • Applying the structured methods means designing different models to represent that particular system.

- Various structured models can be Context models, Process models, Data flow diagrams, State chart diagrams, Entity relationship diagrams, Object models.
- The structured modelling provides the systematic framework for **system modelling** as a part of requirement elicitation and analysis.
- Along with these models **documentation** is essential in order to provide line certain guideline in system development.
- Use of **CASE tool** support also helps to generate *reports* or *code generation* from the system model.

**Drawbacks of structured methods -**

1. **Non functional** requirements can not be represented effectively using structured methods.
2. It is hard to **predict** whether the structured methods that are applied to particular **problem** are suitable for solving that problem or not. Even it is not possible to decide whether the structured method is suitable for particular environment or not.
3. Sometimes **too much documentation** hides the basic requirements of the system. Unnecessary detailing is done in structured methods.
4. The system models are represented in **more detail** so casual user can **not understand** the system because he gets lost in the unnecessary details.

# For Mid Term Exam



**Fill in the Blanks for Mid Term Exam**

1. The software requirements deal with the _____ of the proposed system.
2. The document produced during requirements engineering is popularly known as ___
3. SRS stands for _____ [JNTU : August-16]
4. IEEE 830-1993 is a IEEE recommended standard for ____ document of requirement analysis
5. The term _____ is used to refer to any person or group who will be affected by the system, directly or indirectly. [JNTU : August-16]
6. _____ methods provide a framework for detailed system modeling as part of requirements elicitation and analysis. [JNTU : August-16]
7. _____ diagrams are simple and intuitive. [JNTU : August-16]
8. DFD stands for ____
9. In the context of requirements analysis, partitioning results in the elaboration of data, function, or behavior.(True/False)
10. An external entity is represented using _____ in a DFD.
11. _____ is a software engineering task that bridges the gap between system level requirements engineering and software design
12. _____ requirements describe the functionality or system services.
13. ____ requirements describe the system properties and constraints.
14. ____ problem arises when requirements are not specified in a precise and unambiguous way
15. ___ is a study made to decide whether or not the proposed system is worthwhile.
16. During requirement engineering the next phase after feasibility study is ____.
17. ___ is a requirement discovery technique of observation which is used to understand social and organizational requirements.
18. ___, _____ and _____ are the three types of view points
19. VORD stands for____.
20. ER diagram stands for ____
21. ___ and ___ are the two types of interviews.
22. Traceability matrix generation technique is used during ____.
23. ____ model is a graphical representation of the system in which boundaries are specified.
24. ___ is a collection of attributes that act as an aspect, characteristic, quality or descriptor of the object.
25. ___ model is created for the system processing object oriented programming approach.
26. ____ diagrams is a common practice for object oriented modeling
27. Three types of object models are - inheritance model, ____ and object behavior model.
28. Requirement analysis is critical to the success of the development of the project (True/False).
29. Requirement engineering is an iterative process (True/False)

**30.** It is not possible for the customer to state all the requirements (True/False).

**31.** SRS is known as ___ testing.

## Multiple Choice Questions for Mid Term Exam

**Q.1** Various tasks that are carried out during requirement engineering process are ___.

a feasibility study
b requirements gathering
c software requirements specification
d all of these

**Q.2** The goal of requirements analysis and specification is _____.

a to analyze the cost of the project
b to analyze the schedule of the project
c to understand the customer requirements and document them
d to determine the scope of the project

**Q.3** The process to gather the software requirements from client, analyse and document them is known as _____ .

a requirement engineering
b requirement elicitation
c user interface requirements
d software system analyst

**Q.4** Due to ethnography the number of prototype refinement cycles can be ___

a increased b reduced
c changed d unchanged

**Q.5** _____ and _____ are the two issues of requirement analysis.

a Performance, Design
b Stakeholder, Developer
c Functional, Non-Functional

**Q.6** Which one of the following is a requirement that fits in a developer's module ?

a Availability b Testability
c Usability d Flexibility

**Q.7** In the requirement analysis which model depicts the information domain for the problem ?

a Data models
b Class-oriented models
c Scenario-based models
d Flow-oriented models

**Q.8** SRS is called black box specification of the system because -

a it does not contain the contradictory material
b it does not contain the user documentation
c SRS document should specify only the external behavior of the system
d none of these

**Q.9** ___ makes use of SRS to estimate the cost of the project.

a Project developer
b Project manager
c Tester
d Programmer

**Q.10** "Consider a system where a heat sensor detects an intrusion and alerts the security company ". What kind of requirement the system is providing ?

a Functional
b Non functional
c None of the above

**Q.11** What DFD notation is represented by the rectangle ?

a Data flow
b Data store

c Process
d None of the mentioned

**Q.12** In DFDs, user interactions with the system is denoted by _____.

a circle b arrow
c rectangle d triangle

**Q.13** Behavioral model provides ____ view of the system.

a dynamic b static
c cost effective d none of these

**Q.14** While dealing with the system requirements ___ and ___ factors are considered.

a coding and design
b coding and maintenance
c behavioral and operational
d none of these

**Q.15** Inheritance is a ___ relationship

a has a b type of
c uses d none of these

**Q.16** ____ is more specifically called as generalization

a Inheritance
b Aggregation
c Composition
d Association

**Q.17** Sequence diagram are generally drawn for representing

a inheritance model
b object aggregation
c object behavioral model
d none of these

**Q.18** _____ requirements constraint the system being developed and the development process that should be used. **[JNTU : August-16]**

a Functional b Nonfunctional
c User d System

**Q.19** _____ viewpoints represent people or other systems that interact directly with the system. **[JNTU : August-16]**

a Domain b Indirect
c Interactor d Direct

**Q.20** _____ model shows the principal sub-systems that make up a system. **[JNTU : August-16]**

a Architectural
b Classification
c Composition
d Data-flow

**Q.21** Which of the following is not a check for requirements validation process ? **[JNTU : August-16]**

a Validity b Consistency
c Realism d Parallel

**Answer Keys for Fill in the Blanks :**

| | | | |
|---|---|---|---|
| 1. | requirements | 2. | SRS |
| 3. | Software Requirement Specification | 4. | SRS |
| 5. | stakeholder | 6. | stuctured |
| 7. | Data flow | 8. | Data Flow Diagram |
| 9. | True | 10. | rectangle |
| 11. | Requirement analysis | 12. | Functional |
| 13. | Non functional | 14. | Lack of clarity |
| 15. | Feasibility study | 16. | requirement elicitation and analysis |
| 17. | Ethnography | 18. | Interactor, indirect, domain |

| | | | |
|---|---|---|---|
| **19.** | Viewpoint Oriented Requirement Definition. | **20.** | Entity Relationship diagram |
| **21.** | Open interview and Closed interview | **22.** | requirement management |
| **23.** | Context model | **24.** | Data object |
| **25.** | Object | **26.** | UML or Unified Modeling Language |
| **27.** | object aggregation | **28.** | True |
| **29.** | True | **30.** | True |
| **31.** | black box testing | | |

**Answer Keys for Multiple Choice Questions**

| | | | |
|---|---|---|---|
| **1.** | d | **2.** | c |
| **3.** | a | **4.** | b |
| **5.** | b | **6.** | b |
| **7.** | a | **8.** | c |
| **9.** | b | **10.** | a |
| **11.** | b | **12.** | a |
| **13.** | a | **14.** | c |
| **15.** | b | **16.** | a |
| **17.** | c | **18.** | b |
| **19.** | c | **20.** | d |
| **21.** | d | | |

*END...*

UNIT - III

# 7 Design Engineering

## 7.1 : Design Process and Design Quality

**Q.1 Explain the term - Software design**
**[JNTU : Part A, Marks 2]**

**Ans. :** 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.

**Q.2 What do you mean by design quality ? Explain.** **[JNTU : Part A, Dec.-16, Marks 3]**

**Ans. :** • Design quality is the level of effectiveness of the design function in determining a product's operational requirements that can be converted into finished product.

- The goal of any software design is to produce high quality software.
- In order to evaluate quality of software there should be some predefined rules or criteria that need to be used to assess the software product. Such criteria serve as characteristics for good design.

**Q.3 Elaborate on software design engineering in detail.** **[JNTU : Part B, Dec.-10, ECE, Marks 16]**

**Ans. :** • During software engineering process, the first task is to identify and analyze the requirements. Based on this analysis the software design is developed. This design serves as a basis for code generation and testing. Hence software design is an intermediate process which translates the analysis model into design model.

- The four types of elements of **analysis model** are scenario based elements, class based elements, behavioral elements, and flow oriented elements.
- During the scenario based analysis various models such as use case diagrams, activity diagrams or swimlane diagrams are created. During the class based analysis the class diagrams are created. During flow oriented analysis the Data flow

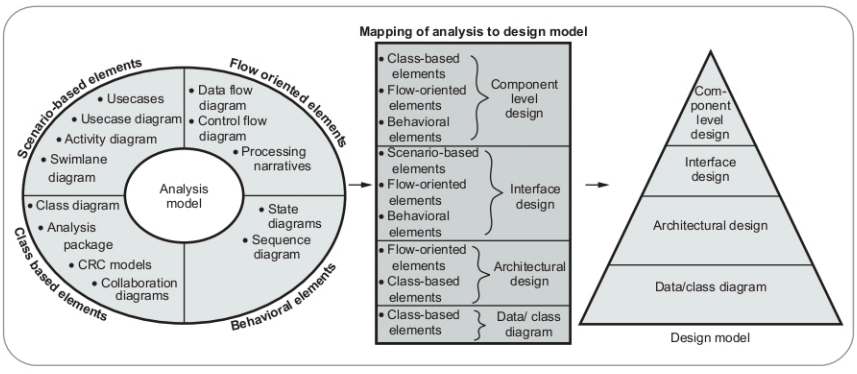



**Fig. Q.3.1 Translating analysis model into the design model**

diagrams are created. Behavioral analysis can be done using state diagrams and sequence diagrams.

- The **class based elements** of analysis model are used to create **class diagrams**.
- The **flow oriented elements** and **class based elements** are used to create **architectural design**.
- The **scenario based elements, flow oriented elements and behavioral elements** are used to create **Interface Design**. The interface design describes how software communicates with systems.
- The **class based elements, flow oriented elements and behavioral elements** are used to create **component level design**. This design transforms the structural elements of software architecture into procedural description of software module. (Refer Fig. Q.3.1).

**Q.4 State and explain various activities in software design. And also discuss the importance of design phase.** ☞[JNTU : Part B, Dec.14, Marks 7]

**Ans. : Activities in Software Design -** Refer Q.3.

**Importance of software design phase :**

- Software design is very important for assessing the quality of software. Because design is the only way that we can accurately translate the user requirements into the finished software product.
- Without software design, it is difficult to test the software product. Not only this, but for making a small change in the software - it is the software design which helps us to make the necessary changes without disturbing other part of the software.

**Q.5 Describe the design process in software development.** ☞[JNTU : Part A, Feb.-10,Marks 3]

**Ans. :** • Software design is an iterative process using which the user requirements can be translated into the software product.

- At the initial stage of the software is represented as an **abstract view**.
- During the sub-sequent iterations data, functional and behavioral requirements are traced in detail. That means at each iteration the refinement is made to obtain lower level details of the software product.

**Q.6 What are the characteristics and criteria for design ?** ☞[JNTU : Part A, Feb.-10, Marks 3]

**Ans. :** 1.The good design should **implement all the requirements** specified by the customer. Even if there are some implicit requirements of the software product then those requirements should also be implemented by the software design.

2. The design should be **simple** enough so that the software can be understood easily by the developers, testers and customers.
3. The design should be comprehensive. That means it should provide complete picture of the software.

## 7.2 : Design Concepts

**Q.7 Explain about various design concepts in detail.**
☞[JNTU : Part B, Dec.-12, Marks 7,May-13, Marks 5]

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

1. **Abstraction :** 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.
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.
4. **Refinement :** Refinement is actually a process of elaboration.The process of program refinement is analogous to the process of refinement and partitioning that is used during requirements analysis.

5. **Pattern :** The design pattern can be defined as - It is a named sugget(something valuable) of insight which conveys the essence of proven solution to a recurring problem within a certain context.
6. **Information Hiding :** 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.
7. **Functional Independence :** By using functional independence functions may be compartmentalized and interfaces are simplified. Independent modules are easier to maintain with reduced error propagation.
8. **Refactoring :** Refactoring is necessary for simplifying the design without changing the function or behavior. Fowler has defined refactoring as "the process of changing a software system in such a way that the external behavior of the design do not get changed, however the internal structure gets improved".

**Q.8 Justify how modular design is an effective design method ?**

**[JNTU : Part B,Feb.-10, Dec.-10,Marks 8]**

**Ans. : Meyer** defines five criteria that enable us to evaluate a design method with respect to its ability to define an effective modular system :

1. **Modular decomposability :** A design method provides a systematic mechanism for decomposing the problem into sub-problems. This reduces the complexity of the problem and the modularity can be achieved.
2. **Modular composability :** A design method enables existing design components to be assembled into a new system.
3. **Modular understandability :** A module can be understood as a standalone unit. It will be easier to build and easier to change.
4. **Modular continuity :** Small changes to the system requirements result in changes to individual modules, rather than system-wide changes.
5. **Modular protection :** An aberrant condition occurs within a module and its effects are constrained within the module.

**Q.9 Define coupling and cohesion.**

**[JNTU : Part B, Dec.-10, Marks 8, Dec.-17, Marks 2]**

**Ans. : Coupling :** Coupling represents how the module can be **connected with other module or with the outside world.**

**Cohesion :** Cohesion is a measure that represents that a **module performs only one task** in software procedure with little interaction with other modules.

**Q.10 What is data abstraction ? Give an example.**

**[JNTU : Part A, Marks 3]**

**Ans. :** In **data abstraction** the collection of data objects is represented. For example for the procedure search the data abstraction will be record. The record consists of various attributes such as record ID, name, address and designation.

**Q.11 Differentiate between procedural and data abstractions.**

**[JNTU : Part B, April-11, Marks 8]**

**Ans. :** The procedural abstraction gives the named sequence of instructions in the specific function. That means the functionality of procedure is mentioned by its implementation details are hidden. For example : Search the record is a procedural abstraction in which implementation details are hidden (i.e. Enter the name, compare each name of the record against the entered one, if a match is found then declare success. Otherwise declare 'name not found')

In data abstraction the collection of data objects is represented. For example for the procedure search the data abstraction will be record. The record consists of various attributes such as record ID, name, address and designation.

**Q.12 Differentiate between content and common coupling.** **[JNTU : Part B, Dec.-11, Marks 7]**

**Ans. :**

| Sr. No. | Content Coupling | Common Coupling |
|---|---|---|
| 1. | Content coupling occurs between two modules when one module refers to the **internals** of the other module. | Common coupling occurs when modules communicate using **global data areas**. |

| | | |
|---|---|---|
| 2. | Content coupling is also known as **pathological coupling**. | Common coupling is also known as **global coupling**. |
| 3. | This type of coupling **does not require any global data or global variable** for communication. | This type of coupling is based on the communication using **global data areas**. |
| 4. | The **major drawback** of content coupling is that - if we want to reuse one component we need to import all the components that are coupled with the component being reused. | The **major drawback** of common coupling is that - modules in the common coupling are tightly coupled. A fault in one module using global data may show up in another module because global data may be updated by any module at any time. This greatly affects the reusability of modules. |
| 5. | The access to internal structure is of main concern in this kind of communication. | The access to global data is of main concern in this kind of communication. |
| 6. | In this coupling, component directly modifies another's data. | In common coupling, it is difficult to determine all the components that affect a data element. |
| 7. | **Example -** Part of Program that searches for the entry of particular employee. when the module does not find the entry for the employee, it directly adds employee modifying the contents of data structure containing employee data. | **Example -** A process control component that maintains the status of operations in a global data store. This control block gets data from multiple sources or supplies data to multiple sinks. Each source process writes the status of its operation directly to the global data store. |

**Q.13 Define and explain about coupling and cohesion. Also differentiate between them.**
☞[JNTU : Part B, June-14, Marks 7]

**Ans. :**

| Sr. No. | Coupling | Cohesion |
|---|---|---|
| 1. | Coupling represents how the **modules are connected** with other modules or with the outside world. | In cohesion, the cohesive module performs **only one thing.** |
| 2. | With coupling **interface complexity** is decided. | With cohesion, **data hiding** can be done. |
| 3. | The goal of coupling is to achieve **lowest coupling.** | The goal of cohesion is to achieve **high cohesion.** |
| 4. | Various **types** of couplings are - Data coupling, Control coupling, Common coupling and Content coupling. | Various **types** of cohesion are - Coincidental cohesion, Logical cohesion, Temporal cohesion, Procedural cohesion and Communicational cohesion. |

**Q.14 Define and explain about different types of cohesion** ☞[JNTU : Part B, Dec.-16, Marks 5]

**Ans. :** • With the help of cohesion the **information hiding** can be done.

- 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.
- Different types of cohesion are :

1. **Coincidentally cohesive** - The modules in which the set of tasks are related with each other loosely then such modules are called coincidentally cohesive.
2. **Logically cohesive** - A module that performs the tasks that are logically related with each other is called logically cohesive.
3. **Temporal cohesion** - The module in which the tasks need to be executed in some specific time span is called temporal cohesive.
4. **Procedural cohesion** - When processing elements of a module are related with one another and must be executed in some specific order then such module is called procedural cohesive.

5. **Communicational cohesion** - When the processing elements of a module share the data then such module is communicational cohesive.

- The goal is **to achieve high cohesion** for modules in the system.

**Q.15 Define and explain about different types of coupling** [JNTU : Part B, Marks 5]

**Ans. :** Various types of coupling are :

i) **Data coupling** - The data coupling is possible by parameter passing or data interaction.

ii) **Control coupling** - The modules share related control data in control coupling.

iii) **Common coupling** - In common coupling common data or a global data is shared among the modules.

iv) **Content coupling** - Content coupling occurs when one module makes use of data or control information maintained in another module.

**Q.16 Explain various types of cohesion and coupling. List in the order of level of cohesion and coupling preferred for component level design.**

[JNTU : Dec. 19, Marks 5]

**Ans. :** Refer Q.14 and Q.15.

**Cohesion and coupling in component level design :**

**The levels of cohesion are -**

From worst to best level

1) Coincidental cohesion 2) Logical cohesion

3) Temporal cohesion 4) Procedural cohesion

5) Communicational cohesion

6) Functional cohesion

**The levels of coupling are -**

From worst to best level

1) Content coupling 2) Common coupling

3) Control coupling 4) Data coupling

**Q.17 Discuss the advantages and disadvantages of modularization** [JNTU : Part B, May-08, Marks 7]

**Ans. : Advantages :**

(1) Due to modularization algorithm can be **understood** easily.

(2) Many programmers can **work on different modules** at the same time which **saves the time.**

(3) **Testing** of **each module** can be done thoroughly.

(4) Modules can be **reused** for similar kind of projects.

(5) **Library programs** can be inserted in separate module.

**Disadvantages :**

(1) **Documentation** of each module need to be maintained.

(2) It can create problems when modules are **linked together** because each link need to be tested properly.

**Q.18 Discuss about refactoring in detail**

[JNTU : Part B, Dec.-11, Marks 7]

**OR What is refactoring? Why is it done?**

[JNTU : Part B,May 08, Marks 7]

**Ans. :** • Refactoring is necessary for simplifying the design without changing the function or behaviour.

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

**Benefits** of refactoring are -

i) The **redundancy** can be achieved.

ii) **Inefficient algorithms** can be eliminated or can be replaced by efficient one.

iii) Poorly constructed or **inaccurate data structures** can be removed or replaced.

iv) Other **design failures** can be rectified.

The decision of refactoring particular component is taken by the designer of the software system.

**Q.19 Discuss the statement, "Abstraction and refinement are complementary concepts.**

[JNTU : Section B, June-14, Marks 8]

**Ans. :** 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.

**Q.20 "Modularity is the single attribute of the software that allows a program to be intellectually manageable" - How this is true ?**
**☞[JNTU : Section A, Marks 2]**

**Ans. :** This is quoted by Glenford Mayers. Consider that there is a large program composed of single module. Such a program cannot be grasped by reader. The control paths, span of reference, number of variables and overall complexity of such a program is beyond understanding. On the other hand if the program is divided into certain number of modules then overall complexity of the program gets reduced. The error detection and handling can be done effectively. Also changes made during testing and maintenance become manageable. Hence it is true that "Modularity is the single attribute of the software that allows a program to be intellectually manageable".

**Q.21 Differentiate between : abstraction and modularization.** **☞[JNTU : Section A, Marks 2]**

**Ans. :** • Abstraction is a software design concept which is applied to refine software solution. In the process of abstraction only necessary information is abstracted form requirement analysis to create the software solution in broad terms. While moving through different levels of abstraction the procedural and data abstraction are prepared. During abstraction the procedural and data objects are prepared.

- Modularity is a design concept in which requirements are divided into separately named and addresable components called as modules. Modularity in design reduces complexity and helps in easier implementation.

**Q.22 Discuss the relationship between the concept of information hiding as an attribute of effective modularity and the concept of module independence ?**
**☞[JNTU : Section B, Feb.-10, CSE/IT/CS & SE, Marks 8]**

**Ans. :** Information hiding can be related to both coupling and cohesion concepts.

For reducing the coupling only those modules are linked together that are absolutely needed. This reduces the coupling between modules. Similarly information is isolated so that the functionalities can be isolated. This improves the cohesion of individual modules.

**Information hiding implies that effective modularity** can be achieved by defining a set of independent modules that communicate with one another only that information necessary to achieve software function. Using abstractions the procedural entities are defined. This is helpful for testing and maintenance.

The concept of module independence helps in achieving modularity, abstraction and information hiding. Independent modules are easier to maintain and test. Reusability of such modules is also possible.

**Q.23 What do you understand by the term Design classes ?** **☞[JNTU : Part A, Marks 2]**

**Ans. :** Design classes are defined as the classes that describe some elements of the problem domain, in which the problem is viewed from user's point of view.

**Q.24 What are the goals of design classes ?**
**☞[JNTU : Part A, Marks 2]**

**Ans. :** The **goal** of design classes is :

1. To refine the analysis classes by providing the **detail design**. Using detailed design further implementation is carried out.
2. To create new set of classes for implementing the core requirements of the software.

**Q.25 What type of design classes does the designer create ? Explain about the "well formed" design class.** **☞[JNTU : Part B, May-09, Marks 8]**

**Ans. : 1. User Interface Class :** The user interface class defines all the interactions with the system. The user interface classes is basically a **visual representation**. It is also called Human Computer Interface (HCI)

**2. Business Domain Class :** Business domain classes are the refinement of analysis classes. These classes

identify the **attributes and services** that are needed to implement some elements of **business domain.**

**3. Process Class :** Process class is used business domain. It define processes required by business domain.

**4. Persistent Class :** These classes represent the data store such as databases which will be retained as it is even after the execution of the software.

**5. System Class :** These classes are responsible for software management and control functions that are used for system operation.

Each class must be **well formed design class**.

**Q.26 Write any three desirable characteristics of design classes** ☞[JNTU : Part A, Marks 3]

**Ans. : • Complete and Efficient**

A design class must be properly **encapsulated** with corresponding attributes and methods. Design class must contain all those methods that are sufficient to achieve the intent of the class.

**• Primitiveness**

Methods associated with one particular class must perform unique service and the class should not provide another way to implement the same service. There should not be conflicting methods.

**• High Cohesion**

A cohesive designed class must posses small, focused set of responsibilities and these responsibilities must be associated with all the attributes of that class.

**• Low Coupling**

Design classes must be collaborated with manageable number of classes. If one design class is collaborated with all other design classes then the implementation, testing and maintenance of the system becomes complicated. The **law of Demeter** suggests that the one particular design class should send messages to the methods of neighbouring design classes only.

## 7.3 : The Design Model

**Q.27 What are the major elements of the design model ? Explain the abstraction dimensions and process dimensions of the analysis and design model.**

☞[JNTU : Part B, May-09, Feb.-10, Marks 16, ☞May-13, Marks 5]

**Ans. :** The major element of design model are -

(1) Architecture Elements

(2) Interface Elements

(3) Data Design Elements

(4) Component Level design Elements

**Design Model :**

- The **process dimension** denotes the progress of design model that occurs due to various software tasks that get executed as the part of software process.
- The **abstract dimension** represents in how much detail the analysis model is transformed into design model.
- In Fig. Q.27.1 the **dashed line** shows the boundary between analysis and design model.
(See Fig. Q.27.1 on next page)
- In both the analysis and design models the same UML diagrams are used but in analysis model the UML diagrams are abstract and in design model these diagrams are refined and elaborated. Moreover in design model the implementation specific details are provided.
- Along the horizontal axis various elements such as architecture element, interface element, component level elements and deployment level elements are given. It is not necessary that these elements have to be developed in sequential manner. First of all the preliminary architecture design occurs then interface design and component level design occur in parallel. The deployment level design ends up after the completions of complete design model.

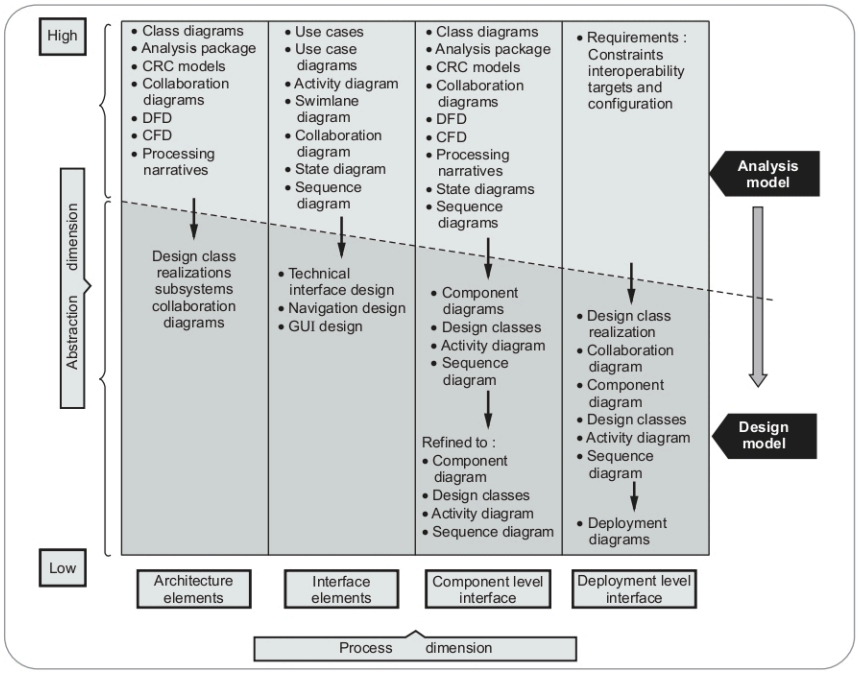



**Fig. Q.27.1 Dimension of design model**

**Q.28 Explain the concept of data design element.**
**[JNTU : Part B, Marks 5]**

**Ans. :** • The data design represents the **high level of abstraction**.

- This data represented at data design level is **refined gradually** for implementing the computer based system.
- The data has great impact on the architecture of software systems. Hence structure of data is very important factor in software design.
- Data appears in the form of **data structures and algorithms** at the program **component level**.
- At the **application level** it appears as the **database** and at the **business level** it appears as **data warehouse and data mining**.
- Thus data plays an important role in software design.

**Q.29 Explain the architectural Design Element**
**[JNTU : Part B, Marks 5]**

**Ans. :** • The architectural design gives the layout for overall view of the software.

- Architectural model can be built using following sources -
- **Data flow models** or **class diagrams.**
- Information obtained from application domain.
- Architectural **patterns and styles.**
- The **architectural style** is a pattern for creating system architecture for given problem. Commonly used architectural styles are -

(1) Data centered architecture

(2) Data flow architecture

(3) Call and return architecture

(4) Object oriented architecture

(5) Layered architecture.

- The **architectural pattern** is basically an approach for handling behavioral characteristics of software systems.

**Q.30 Explain the interface level design elements**
[JNTU : Part B, Marks 5]

**Ans. :** Interface Design represents the **detailed design** of the software system. In interface design how **information flows** from one component to other component of the system is depicted. Typically there are **three types of interfaces -**

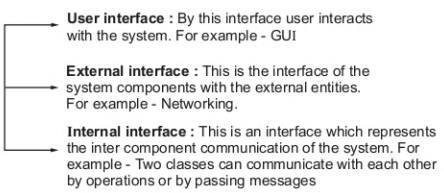



**Fig. Q.30.1 Interface design elements**

**Q.31 Explain component level design elements.**
[JNTU : Part B, Marks 5]

**Ans. :** The component level design is more detailed design of the software system along with the specifications. The component level design elements describe the internal details of the component. In component level design all the local data objects, required data structures and algorithmic details and procedural details are exposed.

Fig. Q.31.1 represents that component **order** makes use of another component **form**.

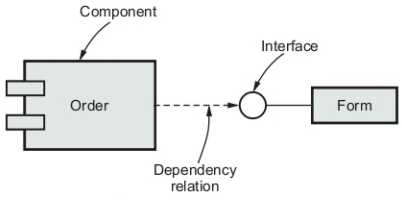



**Fig. Q.31.1 Components**

The **order** is dependent upon the component form. These two objects can be interfaced with each other.

**Q.32 Explain deployment level design elements**
[JNTU : Part B, Marks 5]

**Ans. :** The deployment level design elements indicate how software functions and software subsystems are assigned to the physical computing environment of the software product. For example web browsers may work in mobile phoned or they may run on client PC or can execute on server machines. Refer Fig. Q.32.1.

## 7.4 : Pattern Based Software Design

**Q.33 Explain about pattern based software design in detail** [JNTU : Part B, Dec.-16, Marks 5]

**Ans. :** • In software designing process, certain design patterns can be created and used to solve the problems.

- Sometimes instead of creating new design patterns the existing pattern that satisfies the design needs can be reused.
- Mature engineering branches like mechanical and electrical engineering make use of design patterns. For example - to solve some particular problem in electrical engineering the Integrated circuit design the created. This IC design may vary based on nature of the problem. Thus design can be viewed as an activity in which the same design pattern can be reused with little or no change in the configuration.
- The attributes and operations of the pattern can be **inherited.**
- The design pattern can be used using following design pattern template.

| Template for Design Pattern | |
|---|---|
| **Name :** | The name of the design pattern is given initially. The name must be short and expressive |
| **Purpose :** | The purpose of the design pattern is given in this section. |
| **Known-as :** | If any synonym for the pattern is existing then it should be mentioned. |

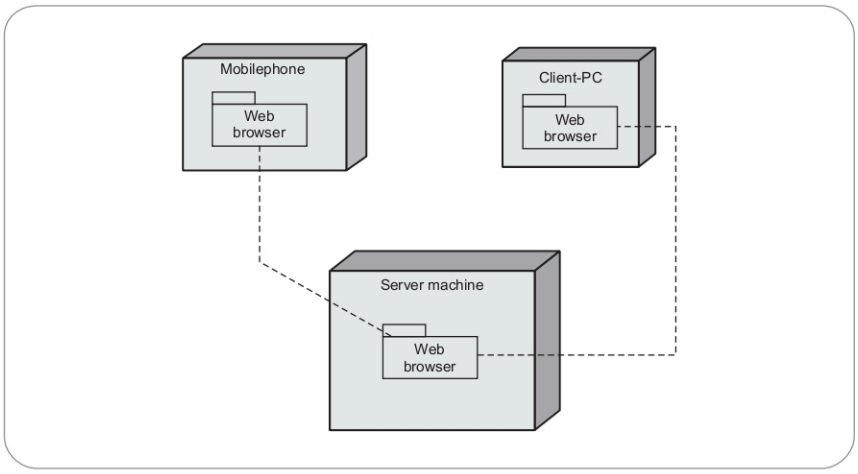



**Fig. Q.32.1 Deployment diagram**

| | |
|---|---|
| **Motivation :** | It denotes nature of the problem along with supporting example. |
| **Applicability :** | It describes the design situation in which the pattern is applicable. |
| **Structure :** | It describes the classes that are required to implement the problem |
| **Participants :** | It includes the responsibilities of the classes that are used in the pattern. |
| **Collaborations :** | It describes all the collaborating classes that share some responsibilities. |
| **Consequences :** | It describe various design forces of the pattern. |
| **Cross References :** | If any related design patterns exist then those are described under this section. |

**Q.33 Explain various types of patterns in Design.**

**Ans. :** Various types of design patterns are -

- **Architectural Pattern**

Architectural design pattern are those design patterns that depict the overall structure of software. These patterns also indicate the relationship among the components and define the rules for inter-relationship between the elements of the software architecture.

- **Design Pattern**

The design patterns are used to represent the design of specific elements, relationships among the components and component to component communication.

- **Idioms**

These are also called as **coding patterns**. These patterns are language specific and are used to implement an algorithmic elements, interfaces, protocols or methods used for component-to component communication.

***END...***

UNIT - III

# 8 Creating an Architectural Design

## 8.1 : Software Architecture

**Q.1 What is software architecture ? Why is it important ?** [JNTU : Part A, Marks 3]

**Ans. :** Software Architecture is a structure of systems which consists of various **components**, externally visible **properties** of these components and the inter-**relationship** among these components.

**Importance of software architecture**

There are three reasons why the software architecture is so important ?

1. Software architecture gives the representation of the computer based system that is to be built. Using this system model even the **stakeholders** can take **active part** in the software development process. This helps in clear specification/understanding of requirements.
2. Some **early design decisions** can be taken using software architecture and hence system performance and operations remain under control.
3. The software architecture gives a clear cut **idea** about the computer based system which is to be built.

**Q.2 Differentiate between Software architecture and software design process.**

**Ans. :**

| Sr. No. | Software architecture | Software design process |
|---|---|---|
| 1. | **Software architecture** deals with high level concepts that do not include any details as how they will be implemented. | **Software design process** deals with high level concepts and is applied to the concrete details so that software can be implemented. |
| 2. | It **deals with the issues** such as what kind of data storage is present, how do modules interact with each other, what recovery system is in place and so on. | It basically **deals** with the individual components, what are the functionalities or modules, what kind of tasks these modules should perform and so on. |
| 3. | Software architecture represents the **bigger picture** which deals with the choice of software framework, languages, scope, goals and high level methodologies. | Software design represents the **smaller picture** which deals with what are different modules of each subsystems and how will they interact with each other. |
| 4. | For building the software architecture, **architect ural patterns** are applied. | For building the software design, the design patterns are applied. |

**Q.3 What is software architecture ? Explain about structure chart with an example.** [JNTU : Part B, Feb.-10, Dec.-10, Marks 8]

**Ans. : Software Architecture -** Refer Q.1.

**Structure 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.

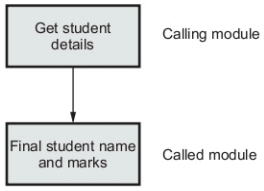



**Fig. Q.3.1 Connector**

- 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 to modules one for getting the employee detail(calling) and another module is for finding the employee name(called). 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. (Refer Fig. Q.3.2.)

- The iterations and decisions on a structured chart is as shown below. (Refer Fig. Q.3.3.)

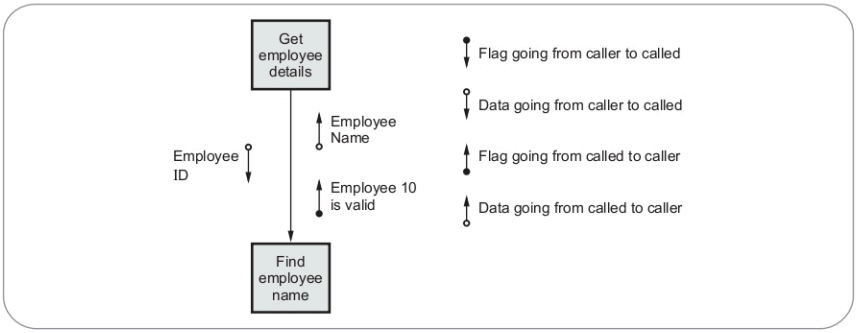



**Fig. Q.3.2 Use of flag**

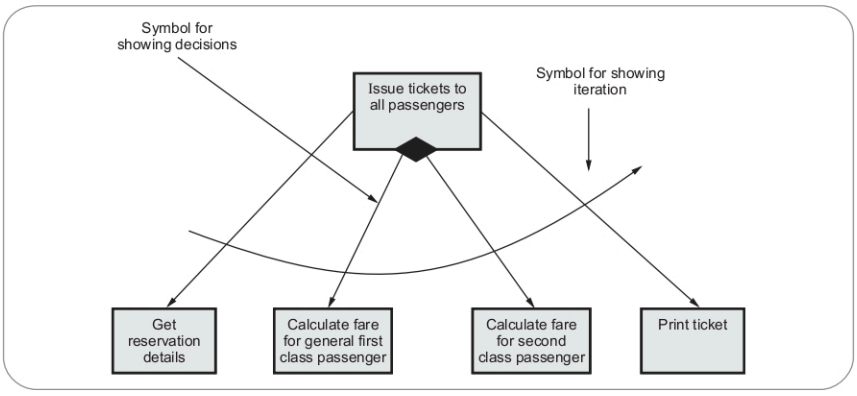



**Fig. Q.3.3 Interaction**

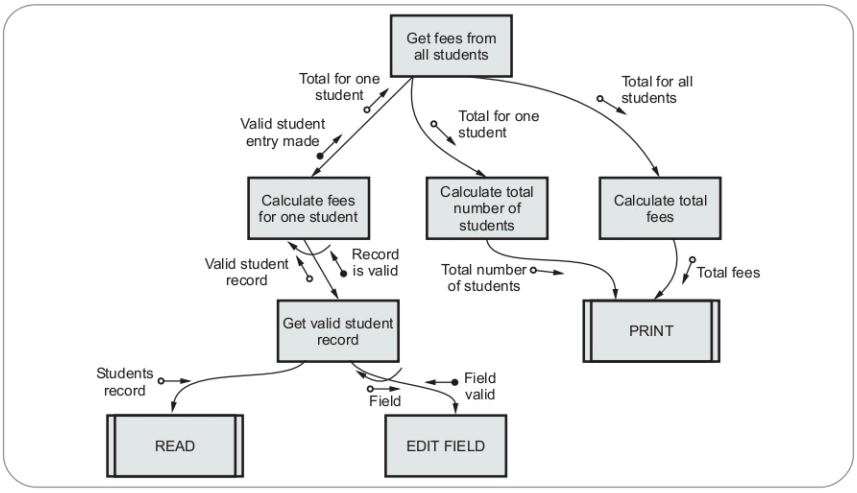



**Fig. Q.3.4 Structure chart**

- The example of a structure chart is as given below. (Refer Fig. Q.3.4)

**Q.4 Discuss why software architecture play important role during development and discuss various architectural terminology**

**[JNTU : Part B, Dec.-10, 12, Mark 7]**

**Ans. : Software Architecture Importance : Refer Q.1.**

A computer-based system (software is part of this system) exhibits one of the many available architectural styles. Every architectural style describes a system category that includes the following -

**(1) Components :** Computational components such as clients, server, filter, and database to execute the desired system function.

**(2) Connectors :** A set of connectors such as procedure call, events broadcast, database protocols, and pipes to provide communication among the computational components.

**(3) Constraints :** Constraints to define integration of components to form a system.

**(4) Semantic model :** A semantic model, which enable the software designer to identify the characteristics of the system as a whole by studying the characteristics of its components.

## 8.2 : Data Design

**Q.5 What is data design ? Explain with an example.** **[JNTU : Part B, May-09, Dec.-14, Marks 7]**

**Ans. :** • Data design is basically the model of data that is represented at the high level of abstraction.

- The data design is then progressively refined to create implementation specific representations.
- Various elements of data design are,
  - **Data object -** The data objects are identified and relationship among various data objects can be represented using entity relationship diagrams or data dictionaries.
  - **Databases -** Using software design model, the data models are translated into data structures and databases at the application level.
  - **Data warehouses -** At the business level useful information is identified from various databases and the data warehouses are created. For extracting or navigating the useful business information stored in the huge data warehouse then data mining techniques are applied.

**Q.6 Write and explain the steps used in data design.** ☞[JNTU : Part B, Marks 5]

**Ans. :** *1.Apply systematic analysis on data*

Represent data objects, relationships among them and data flow along with the contents.

2. *Identify data structures and related operations*

   For the design of efficient data structures all the operations that will be performed on it should be considered.

3. *Establish data dictionary*

   The data dictionary explicitly represents various data objects, relationships among them and the constraints on the elements of data structures.

4. *Defer the low-level design decisions until late in the design process*

   Major structural attributes are designed first to establish an architecture of data. And then low-level design attributes are established.

5. *Use information hiding in the design of data structures*

   The use of information hiding helps in improving quality of software design. It also helps in separating the logical and physical views.

6. *Apply a library of useful data structures and operations*

   The data structures can be designed for reusability. A use of library of data structure templates (called as abstract data types) reduces the specification and design efforts for data.

7. *Use a software design and programming language to support data specification and abstraction*

The implementation of data structures can be done by effective software design and by choosing suitable programming language.

## 8.3 : Architectural Styles and Patterns

**Q.7 What is architectural style ?** ☞[JNTU : Part A, May-07, Marks 02]

**Ans. :** The architectural model or style is a pattern for creating the system architecture for given problem. However, most of the large systems are heterogeneous and do not follow single architectural style.

**Q.8 List out some commonly used architectural styles.** ☞[JNTU : Part A, Marks 3]

**OR Explain in brief the taxonomy of various architectural styles.** ☞[JNTU : Part A, Dec.-19, Marks 2]

**Ans. : The commonly used architectural styles are,**

1. Data centered architectures.
2. Data flow architectures.
3. Call and return architectures.
4. Object oriented architectures.
5. Layered architectures.

**Q.9 Explain about architectural pattern in detail.** ☞[JNTU : Part B, April-11, Marks 8]

**Ans. :** The architectural pattern is basically an approach for handling **behavioural characteristics** of software systems. Following are the architectural pattern domains.

**1. Concurrency :** Concurrency means handling multiple tasks in parallel. For example in operating system, **multiple tasks** are executed in parallel. Hence concurrency is a pattern which represents that the system components can interact with each other in parallel. The benefit of this pattern is that system **efficiency** can be achieved.

**2. Persistence :** Continuity in the data can be maintained by the persistence pattern. In other words the data used in earlier execution can be made available further by storing it in files or in **databases**. These files/databases can be modified in the software system as per the need. In **object oriented** system the values of all attributes various operations that are to be executed are persistent for further use. Thus broadly there are two patterns. **i) Database management pattern ii) Application level pattern.**

**3. Distribution :** Distribution pattern refers to the way in which the system components communicate with each other in distributed systems. There are two major problems that occur in distribution pattern

- The nature of interconnection of the components.
- The nature of communication.

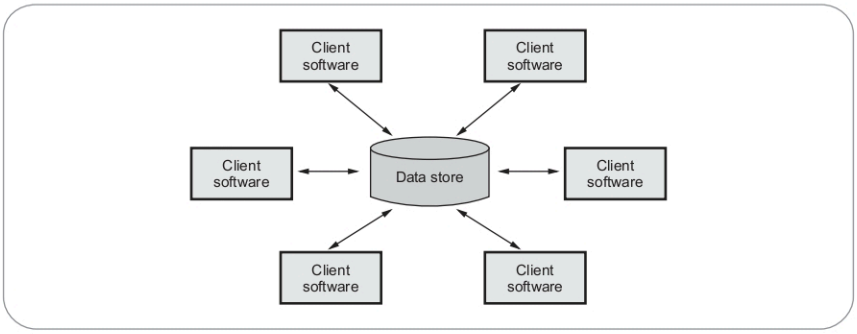



**Fig. Q.10.1 Data centered architecture**

These problems can be solved by other pattern called **broker pattern.** The broker pattern lies between server and client components so that the client server communication can be established properly. When client want some service from server, it first sends message to broker. The broker then conveys this message to server and completes the connection. Typical example is CORBA. The CORBA is a distributed architecture in which broker pattern is used.

**Q.10 Explain data centered architectural style.**
**[JNTU : Part B, Marks 5]**

**Ans. :** In this architecture the data store lies at the center of the architecture and other components frequently access it by performing add, delete and modify operations. The client software requests for the data to central repository. Sometime the client software accesses the data from the central repository without any change in data or without any change in actions of software actions.

Data centered architecture posses the property of interchangeability. Interchangeability means any component from the architecture can be replaced by a new component without affecting the working of other components.

In data centered architecture the data can be passed among the components.

In data centered architecture,

**Components are :** Database elements such as tables, queries.

**Communication are :** By relationships.

**Constraints are :** Client software has to request central data store for information. Refer Fig. Q.10.1.

**Q.11 Explain Call and Return Architecture style in detail.** **[JNTU : Part B, Marks 5]**

**Ans. :** The program structure can be easily modified or scaled. The program structure is organized into modules within the program. In this architecture how modules call each other. The program structure decomposes the function into control hierarchy where a main program invokes number of program components.

In this architecture the hierarchical control for call and return is represented. (Refer Fig. Q.11.1 on next page.)

**Q.12 What data Flow architecture style ? Explain.**
**[JNTU : Part B, Marks 5]**

**Ans. :** In this architecture series of transformations are applied to produce the output data. The set of components called **filters** are connected by **pipes** to transform the data from one component to another. These filters work independently without a bothering about the working of neighbouring filter.

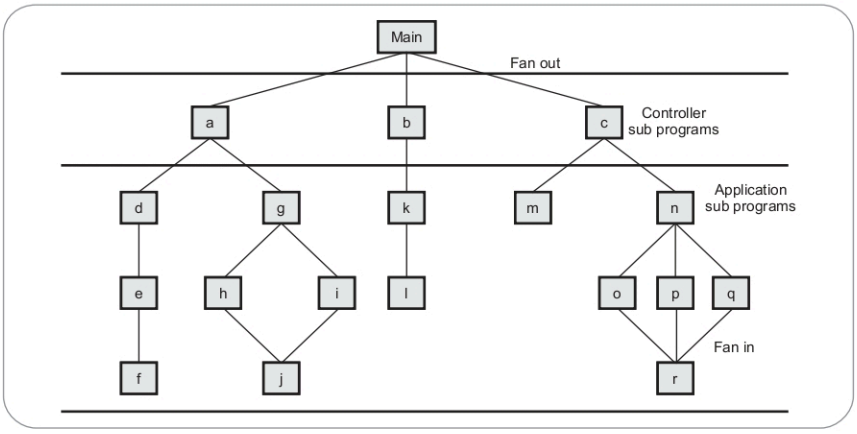



**Fig. Q.11.1 Call and return architecture**

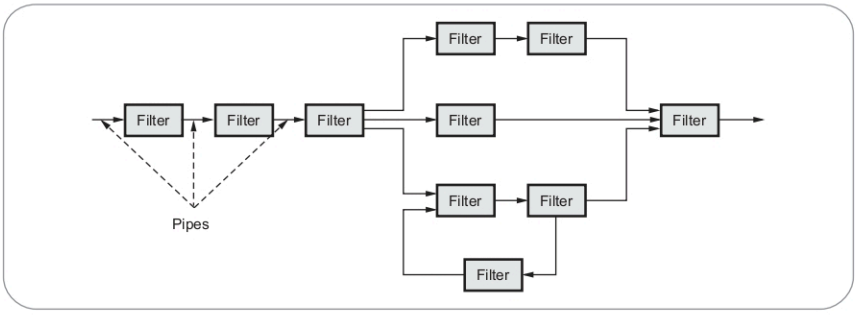



**Fig. Q.12.1 Pipes and filters**

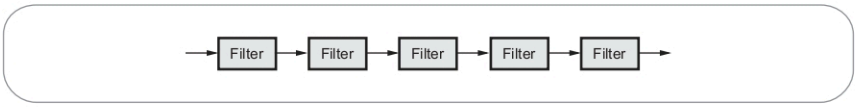



**Fig. Q.12.2 Batch sequential**

If the data flow degenerates into a single line of transforms, it is termed as batch sequential.

In this pattern the transformation is applied on the batch of data. (Refer Fig. Q.12.1.)

**Q.13 Explain layered architecture in detail.**

**[JNTU : Part B, Marks 5]**

**Ans. :** The layered architecture is composed of different layers. Each layer is intended to perform specific operations so machine instruction set can be generated. Various components in each layer perform specific operations.

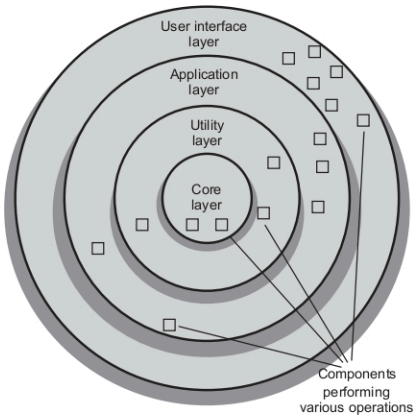



**Fig. Q.13.1 Layer architecture of components**

- The outer layer is responsible for performing the user interface operations while the components in the inner layer perform operating system interfaces.
- The components in intermediate layer perform utility services and application software functions.

**Q.14 Discuss various categories of architectural style.** **[JNTU : Part B, May-07, Marks 05]**

**Ans. :** Refer Q.7, Q.8, Q.9, Q.10, Q.11, Q.12 and Q.13.

**Q.15 Give a brief taxonomy of architectural styles and patterns.** **[JNTU : Part B, May-13, Marks 5]**

**Ans. :** Refer Q.8 and Q.9, Q.10 and Q.11.

## 8.4 : Architectural Design

**Q.16 What is an architectural context diagram ? Explain with an example.**
**[JNTU : Part B, Dec.-10, Marks 8]**

**Ans. :** The context model is a graphical model in which the environment of the system is defined by showing the external entities that interact with the software system.

**Q.17 Give the architecture context diagram for room temperature monitor system.**
**[JNTU : Part B, May-13]**

**Ans. :**

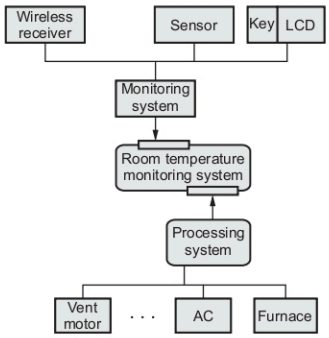



**Fig. Q.17.1 Context diagram for room temperature monitoring system**

**Q.18 What is an archetype ? How would you define it ?** **[JNTU : Part B, Dec.-10, Marks 8]**

**Ans. :** • Defining archetype is a **basic step** in architectural design, more precisely in functionality based design.

- Archetype is a **core abstraction** using which the system can be structured.
- Using archetypes, a **small set** of entities that describe the major part of system behaviour can be described.
- Typically archetypes are the stable elements and they do not change even though system undergo through various changes.
- Identifying archetypes is a critical task and it requires well experienced architect.
- Following figure represents various archetypes.

Various types of archetypes are

1) **Point or node :** It refers to the highest level of abstraction in which there is a cohesive collection of input and output functionalities.
2) **Detector :** Detectors capture the core functionalities of the system. For example, in temperature control system sensors for temperature is a sensor.

3) **Control unit or controller :** Controllers are the entities that are useful for controlling behaviour of the system. For example, in temperature control system. When the temperature exceeds beyond some threshold value alarming and dis-alarming system is required. Such a system acts as a controller or control unit.

4) **Indicator or output :** It represents the generic output functionalities. For example, monitoring system of any computer based system acts as an indicator.

- In software engineering archetype is a number of **major components** that are used to describe the system which we want to build.

## 8.5 : Assessing Alternative Architectural Designs

**Q.19 List out two alternative approaches used for architecture design.** ☞[JNTU : Part A, Marks 2]

**Ans. :** There are two alternative approaches used for architecture designs -

1. The first approach is to apply iterative method to access design trade-offs.
2. The pseudo quantitative technique for accessing design quality.

**Q.20 Discuss various criteria for accessing architectural design.**
☞[JNTU : Part B, April-11, Marks 8]

**Ans. : Two criteria for accessing architectural Design are -**

1. **Architecture Trade-off Analysis Method :** This method has an iterative evaluation process.

Various design analysis activities that are conducted are -

1. ***Collect Scenarios :*** Use cases are developed for representing the system from user's point of view. Hence a set of use cases help in collecting the scenarios.
2. ***Elicit requirements, constraints and environment description :*** This information is obtained during requirement analysis. Customer, user and stakeholders must specify the requirements correctly.
3. ***Describe architectural style chosen for representing scenarios and requirements :*** The architectural style should describe various views about the system. Those views are -
   i) **Model view :** Using model view the working of components and the information hiding among them can be described.
   ii) **Process view :** This view helps to analyse the system performance
   iii)**Data flow view :** Using this analysis, the functional requirements analysis can be made.
4. ***Evaluate quality attributes individually :*** Various quality attributes that can be evaluated are - reliability, performance, security, flexibility, maintainability, testability, portability, interoperatibility and reusability.
5. ***Identify the sensitivity of quality attribute :*** For checking the sensitivity of these attributes, small changes are made in the architecture and observe the effects of these changes on the quality attributes. If any attribute gets affected significantly by these changes then that attribute is considered as a sensitive point.
6. ***Criticize quality attribute :*** By making small changes in the architecture, sensitive points are obtained. Now the architecture trade-off points can be obtained by criticizing these quality factors.

**2. Pseudo Quantitative Technique :**

- In this approach, particular architecture should meet a predefined goodness criteria. This criteria is called **design dimension**. This criteria consists of various quality attributes such as reliability, security, performance, maintainability, flexibility, testability, reusability and interoperability.
- First model is **spectrum analysis**. This model assesses various architectural designs using the **goodness** spectrum. Then particular software architecture is proposed. This architecture is assessed by assigning some **score** to each of its design dimension. All these scores are summed up to obtain the total score of the entire design. The best case, worst cases are considered to obtain the best case and worst case score of the system. From

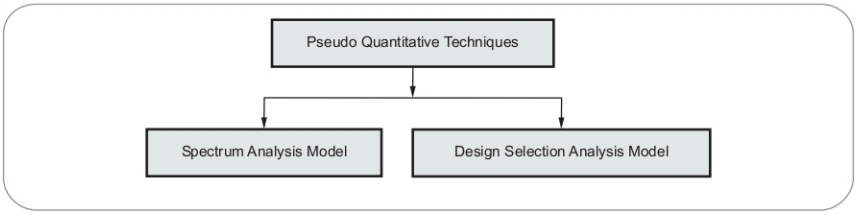



**Fig. Q.20.1 Pseudo quantitative techniques**

these values spectrum index is calculated using following equation

$$I_S = [(S - S_W)/(S_b - S_W)] \times 100$$

where

S is the total score

$S_W$ is the worst case score

$S_b$ is the best case score

This spectrum index indicates the spectrum within which the proposed system architecture can be built.

- **Design selection analysis** is another model in which a set of **design dimensions** are defined. The proposed architecture is assessed to determine the number of design dimensions that achieve when compared with best case system. The design selection index d can be calculated as -

$$d = (N_s/N_a) \times 100$$

where

$N_s$ is number of design dimensions achieved by the proposed architecture

$N_a$ sis the total number of design dimensions.

If the index value is higher then the proposed architecture is more close to the ideal system.

## 8.6 : Mapping Data Flow into Software Architecture

**Q.21 What is transform flow and transaction flow ?**

**Ans. : Transform flow :** A transform flow is a sequence of paths which forms transition in which input data are transformed into output data. (Refer Fig. Q.21.1.)

**Transaction flow :** A transaction flow represents the information flow in which single data item triggers the overall information flow along the multiple paths. This triggering data item is called **transaction**. (Refer Fig. Q.21.2 on next page.)

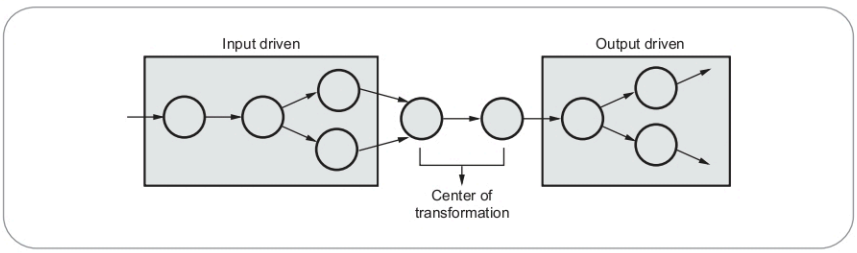



**Fig. Q.21.1 Transform flow**

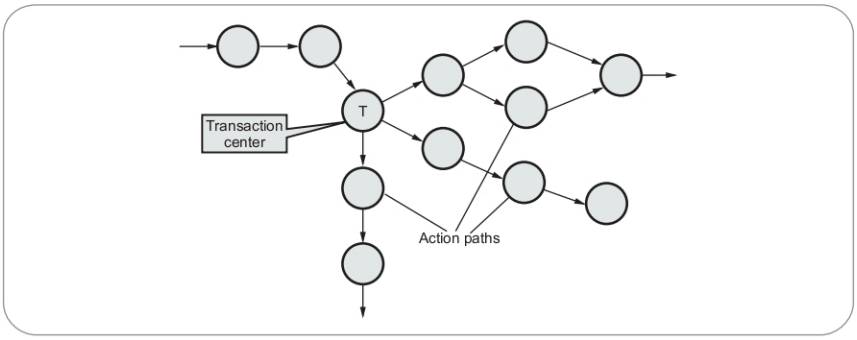



**Fig. Q.21.2 Transaction flow**

**Q.22 Explain the process of mapping dataflow into software architecture.**

**[JNTU : Part B, Dec.-16, Marks 5]**

**OR Discuss with an example, how to map data flow into a software architecture.**

**[JNTU : Part B, Dec.-13, Marks 7]**

**OR Illustrate with neat diagrams the process of mapping data - flow into a software architecture.**

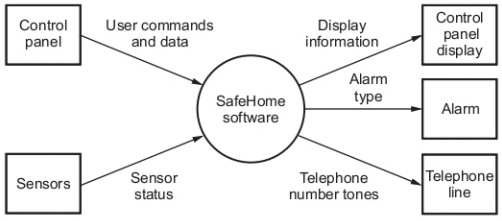
**[JNTU : Part B, Dec.-19, Marks 5]**

**Ans. :** The steps for transform mapping are as follows -

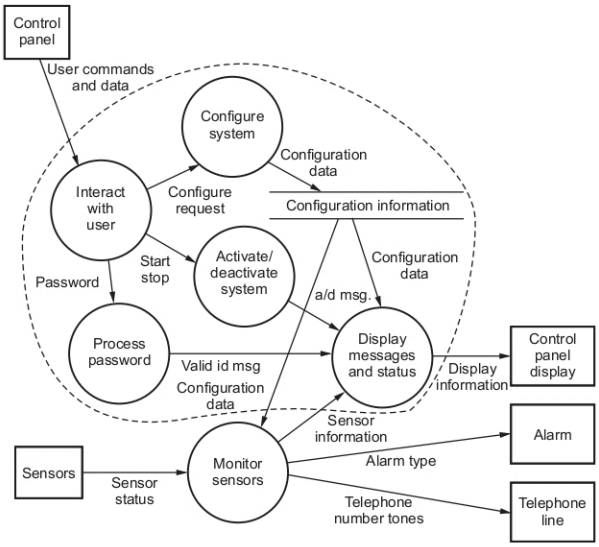
1. Review the fundamental system model.
2. Review and refine DFD for the software.
3. Determine whether the DFD has transform or transaction flow characteristics
4. Isolate the transform center by specifying incoming and outgoing flow boundaries.
5. Perform "First Level Factoring".
6. Perform "Second Level Factoring".
7. Refine the "First-cut" program structure using design heuristics for improved software quality.

Example - Mapping dataflow into software architecture for SafeHome Software

**Step 1 :** Review the fundamental system model.

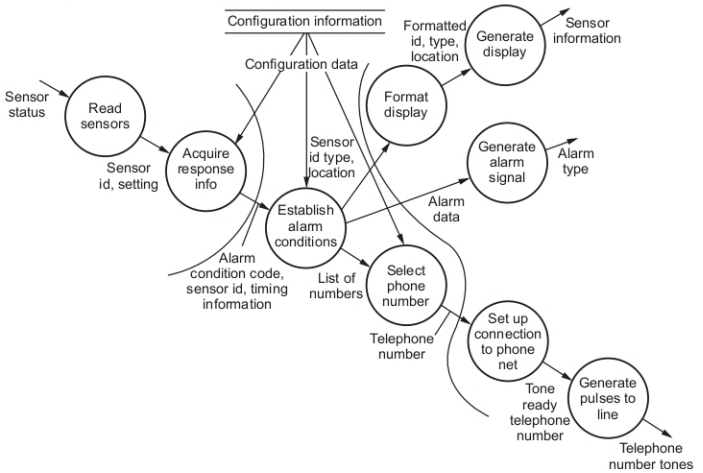


**Step 2 :** Review and refine DFD for the software.

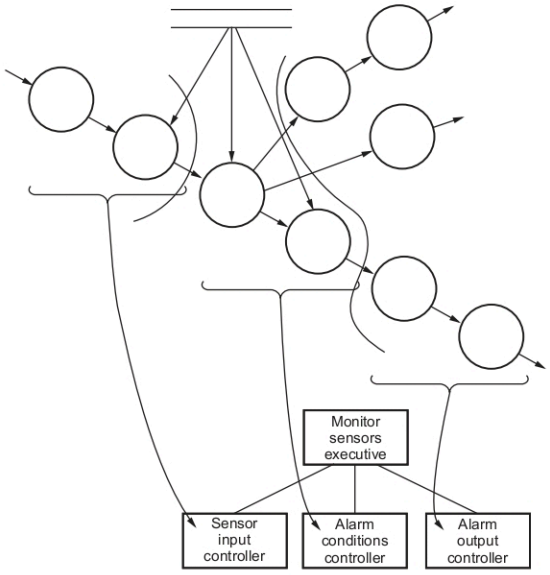



**Step 3 :** Determine whether the DFD has transform or transaction flow characteristics

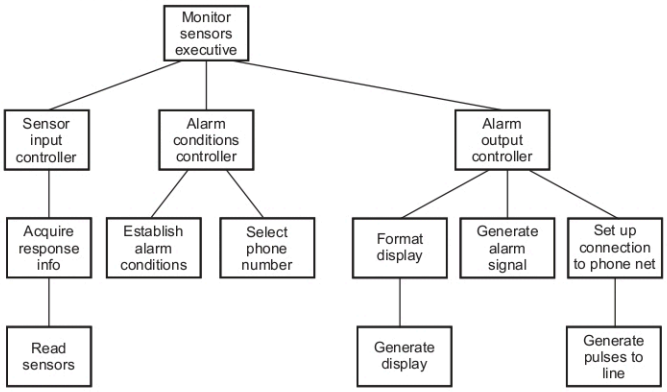
**Step 4 :** Isolate the transform center by specifying incoming and outgoing flow boundaries. Incoming and outgoing flow boundaries are open to interpretation and different designers may select slightly different points in the flow as boundary locations.



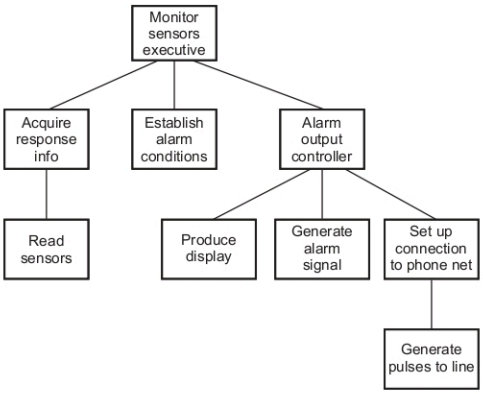
**Step 5 :** Perform "First Level Factoring" - (i) The factoring results in a program structure in which top level modules perform decision making and low level modules perform most input, computational and output work.(ii) Middle level modules perform some control and do moderate amounts of work.




**Step 6 :** Perform "Second Level Factoring" - (i) Individual transforms(bubbles) of DFD are now mapped into appropriate modules within the program structure. (ii) In the next step, modules can be combined or split apart to achieve a good level of cohesion and coupling within the program structure.



**Step 7 :** Refine the "First-cut" program structure using design heuristics for improved software quality - Modules can be exploded or imploded to produce sensible factoring, good cohesion, minimal coupling and most importantly, a structure that can be implemented.




## 8.7 : Conceptual Model of UML

**Q.23 What is UML?**

**Ans. : Definition :** The Unified Modeling Language (UML) is a standard diagramming notation for specifying, visualizing, constructing and documenting the artifacts of software systems, as well as for business modeling and other non-software systems.

**Q.24 Explain the goals and importance of UML**

**Ans. :**

1. UML is intended to provide ready- to use and expressive visual modeling language in order to develop the effective system model.
2. The modeling must be independent of the programming language or platform but at the same time the automatic code that can be generated by this tool must support the modern programming languages.
3. This tool supports the high level concepts such as collaborations, framework, patterns and components.
4. It helps to integrate the organizational standards in the software systems.

**Q.25 Explain in detail the three different perspectives of UML**

**Ans. :**

1. **Conceptual perspective :** Using this perspective, the things in a real world situation are described by the UML diagram.
2. **Specification perspective :** Using this perspective, the UML diagrams describe the software abstractions or components with specification or interface.
3. **Implementation perspective :** Using this perspective, the UML diagrams describe the particulars of implementation.

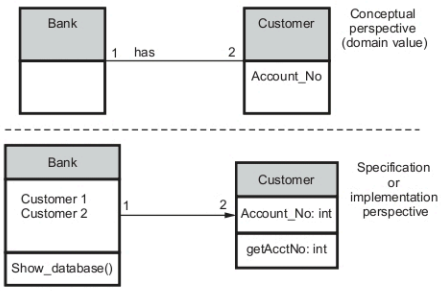



**Fig. Q.25.1 Three perspectives of UML**

The most commonly used perspective is specification perspective. However, some of the software systems make use of implementation perspective.

**• Meaning of class in different perspective**

The classes used in the above described perspectives have three different names. Hence we define -

1. **Conceptual class :** This type of class represents the real world entities.
2. **Software class :** This type of class represents the specification or implementation perspective of a software component.
3. **Implementation class :** This type of class is implemented using some suitable object oriented programming language. The examples of object oriented programming languages are C++, Java.

## 8.8 : Basic Structural Modeling

**Q.26 Explain different types of diagrams that can be drawn using UML.** ☞[JNTU : Part B, Marks 10]

**Ans. :** Diagrams are the graphical representation of the set of the elements. The diagrams can be drawn to represent the system from different perspectives. The same elements may appear in different diagrams. The diagram contains the collection of elements and relationship among these elements. Various types of diagrams that can be drawn in UML are -

1. Class diagram
2. Object diagram
3. Component diagram
4. Use case diagram
5. Sequence diagram
6. State diagram
7. Activity diagram
8. Deployment diagram
9. Package diagram

**Class diagram :** It contains the set of classes, interfaces, collaborations and their relationships. The class diagram represents the static design view of the system. These diagrams are commonly drawn in UML.

**Object diagram :** These diagrams show the relationship among the different objects. Objects are the instances of the classes, hence the object diagrams represent real or prototypical cases. These diagrams represent the static design view of the system.

**Component diagram :** This diagram contains the collection of components and connectors. It represents the static design view of the system. It shows the encapsulated classes, interfaces, ports and internal structure of the system.

**Use case diagram :** These diagrams are drawn to represent the static design view of the system. The use cases represent the behaviour of the system. It consists of collection of use cases, actors(special type of classes) and their relationships.

**Sequence diagram :** These are kind of interaction diagrams consisting of set of objects, roles along the messages. They are specially useful for modeling the behaviour of the system.

**State diagram :** The state diagrams represent the states, transitions, activities and events. The event -ordered behaviour of the object is emphasized.

**Activity diagram :** This diagram represents the flow of control or data step by step occurring in computations. These diagrams emphasize on representation of functionality. These diagrams represent the dynamic view of the system.

**Deployment diagram :** It addresses the static deployment view of the architecture. The run time processing of the components is represented by these diagrams.

**Package diagram :** These diagrams show the organizational units and their dependencies.

## 8.9 : Class Diagrams

**Q.27 What is class diagram ? Explain different elements of class diagram ?** [JNTU : Part B, Marks 10]

**Ans. : The UML class diagram is used to illustrate classes, interfaces and their associations. The class diagram represents static object modeling.**

- Various elements of the class diagram are -

1. Classifier
2. Attributes and association lines
3. Methods or operations
4. Generalization
5. Dependency
6. Composition and aggregation

Let is discuss these concepts with the help of examples.

### Classifier

The **classifier** represents the behavioural and structural features of the system. It can be specialized. The various elements in classifier are classes, interfaces, use cases and actors. In class diagram the most commonly used classifiers are classes and interfaces.

### Attributes and Association Lines

The attributes of the classifier represent the structural properties. The attributes can be represented in various ways such as -

- Attribute text
- Association line
- Both together

For example -

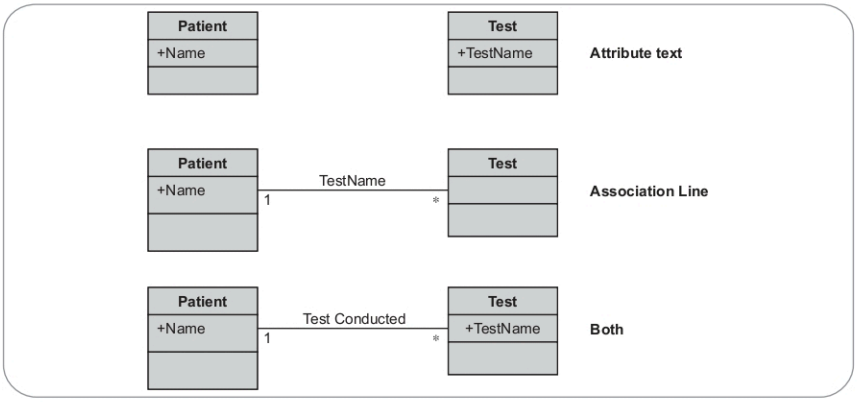



**Fig. Q.27.1**

- The full format of attribute text is

  **Visibility Name: type multiplicity=default{property-string}**

- The visibility means public(+), private(-) or protected(#). It denotes the modes of accessibility of the attribute. Usually, the attributes have the private visibility mode.

- The **multiplicity** can be 1 ..1, 1..* and so on. It is denoted at the end of association line.
- While using the association relationship the **rolename** is mentioned which denotes the association (relationship) between the two objects.
- The **navigation arrow** is used to represent the association between two classes.
- The relationship between two classes can be represented by either attributes or by associations. But when an object can be described by certain properties which are associated with **primitive data types** then such properties are denoted by the attributes of that object. When there is a relationship between two objects that has visual emphasis then those objects are connected by the **association line.** For example -

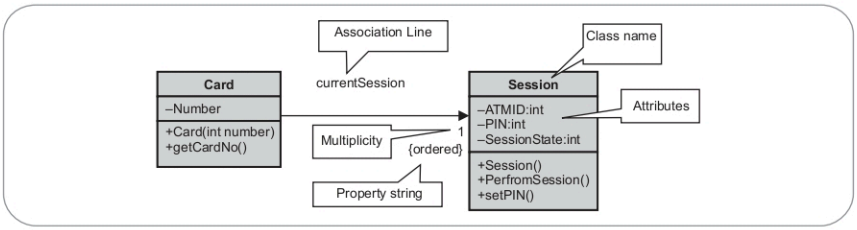



**Fig. Q.27.2 Class**

### Methods and Operations

For denoting the implementation the methods are used in the class. These methods can be accessed by the object of a class. Refer Fig. Q.27.2 in which the entities in the third compartment of the class are all methods.

### Generalization

**Generalization** is a relationship between the superclass and subclass. This is **"is a"** relationship. A **generalization** is shown as a line with a hollow triangle as an arrowhead between the symbols representing the involved classifiers. The arrowhead points to the symbol representing the general classifier.

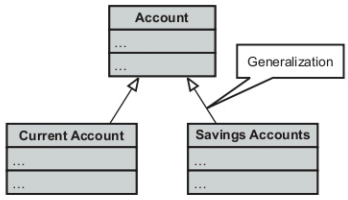



**Fig. Q.27.3 Generalization relationship**

For example - In banking system the generalization relationship is represented as

By the software perspective this relationship implies the **inheritance** concept of object oriented programming.

### Dependency

The dependency link is most commonly used in class and package diagram. In this relationship the **client** element must have the knowledge of another **supplier** element.

The dependency is shown using the dashed line having arrow at one end. The arrow line is from client to supplier.

For example -

The dependency relationship indicates that the client class performs one of the following functions -

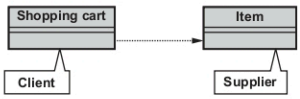



Fig. Q.27.4 Dependancy relationship

The dependency relationship indicates that the client class performs one of the following functions -

- Sends a message to a supplier class
- Temporarily uses a supplier class that has global scope
- Temporarily uses a supplier class as a parameter for one of its operations
- Temporarily uses a supplier class as a local variable for one of its operations

**Composition and Aggregation**

Aggregation is used to represent the whole-part relationship. It normally posses the Has-a relationship. It is denoted as follows -



Fig. Q.27.5

**For example :** The aggregate relationship is


Fig. Q.27.6 Aggregation

**Composition :** Composition is a special kind of aggregation which represents the whole-part relationship. To be more specific, a **restricted** aggregation is called composition. When an object contains the other object, if the contained object cannot exist without the existence of container object, then it is called composition. It is denoted as follows -



Fig. Q.27.7

**For example :** The composite relationship for ATM system is

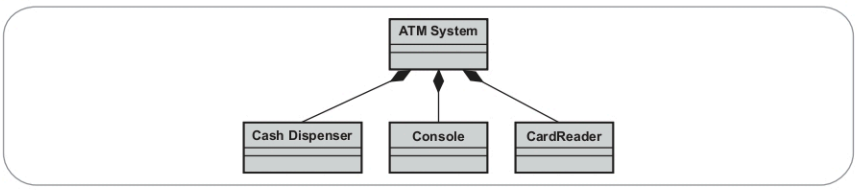



Fig. Q.27.8 Composition

**Q.28 Draw a class diagram for ATM system.**

**Ans. :**

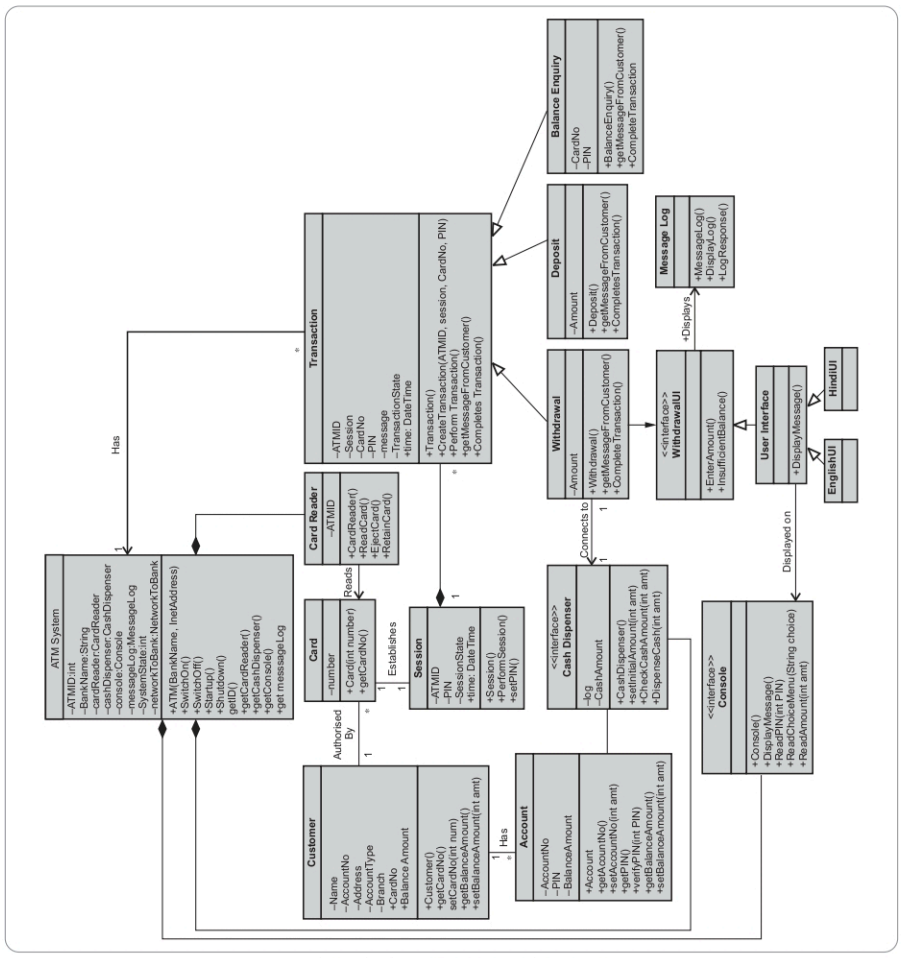



**Fig. Q.28.1 Class diagram for ATM system**

**Q.29 Draw a class diagram for an inventory system**

**Ans. :**

**Problem Description :**

- The Inventory management system is handled by the Inventory manager.
- He first logs in to the system. For checking the availability of the item he checks the inventory.
- He can purchase the items for the stock and then updates the database accordingly.

- If any expired items are present in the stock then he removes those items.
- An order processing clerk can demand for the required items from the stock and then inventory manager processes this order.
- The billing of the items is prepared and is sent to the Accounts department. The accountant pays the bills for the purchased stock. Similarly he deposits the money received when the order gets fulfilled for the customer.
- The stock summary report is generated periodically by the inventory manager.
- The class diagram is as shown below -

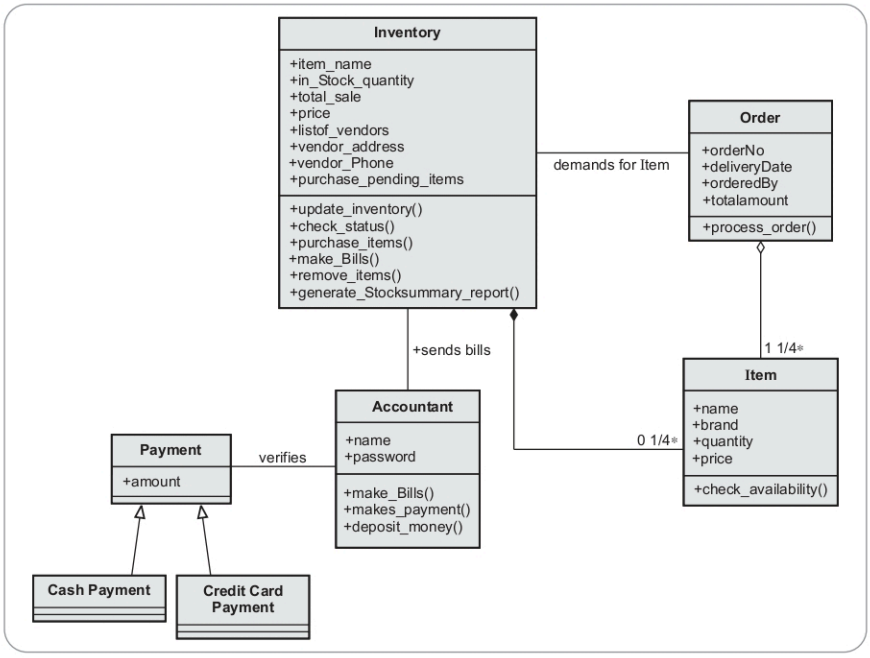



Fig. Q.29.1

**Q.30 Design a class diagram for airline reservation system.**

**Ans. :**

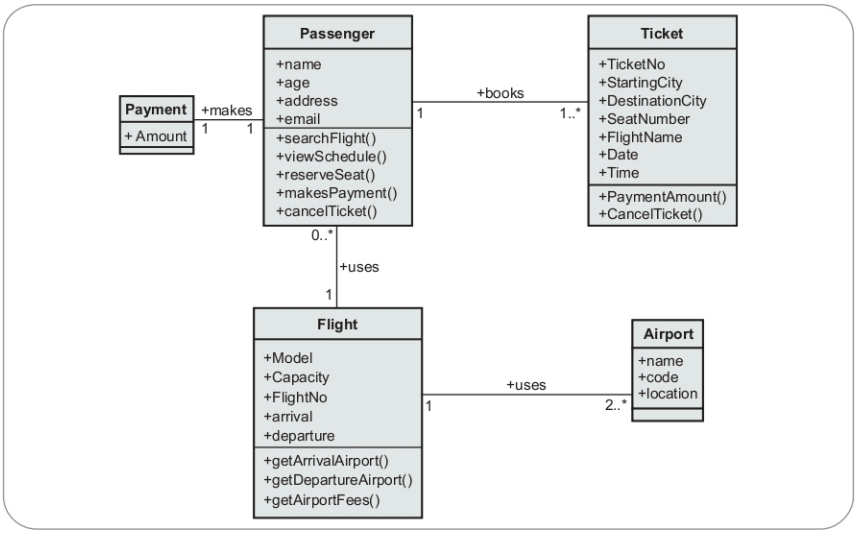



**Fig. Q.30.1**

**Q.31 Draw class diagram for library management system.** ☞ [JNTU : Dec.-19, Marks 5]

**Ans. :**

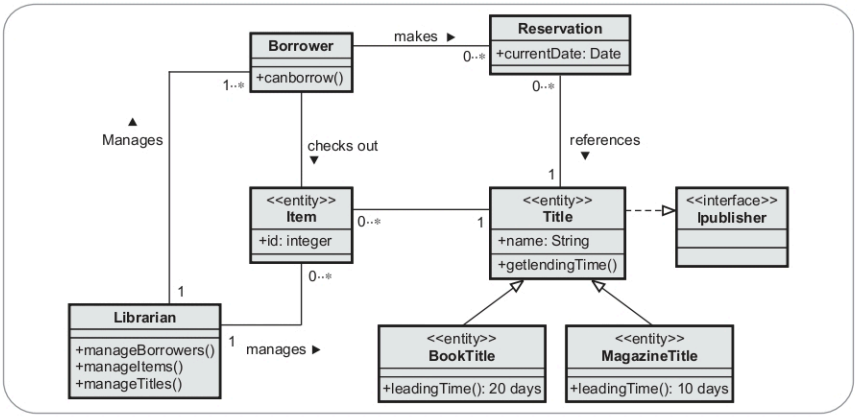



**Fig. Q.31.1 Class diagram for unified library application**

## 8.10 : Sequence Diagram

**Q.32 Explain sequence diagram with example.** ☞[JNTU : Part B, Dec.-19, Marks 5]

**Ans. :** • The sequence diagram is a graphical representation that shows the sender and receiver objects and sequence of messages. Thus interaction between the objects is represented by the sequence diagram.

• The object are represented at the top within the rectangle.

• The vertical line represents the **lifeline**. The messages are represented on the horizontal rows.

• The time proceeds from top to bottom.

• For example -

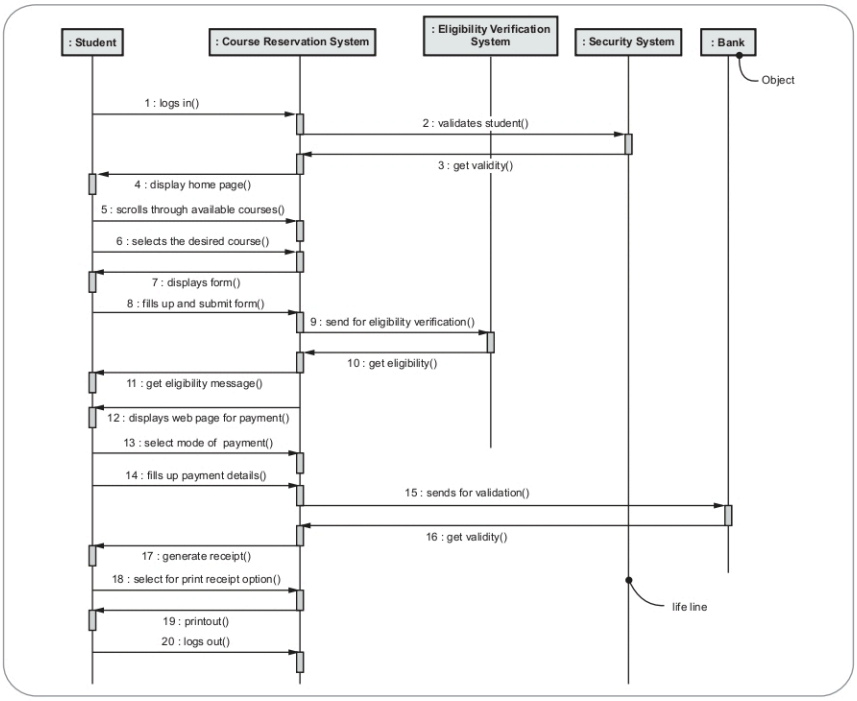



**Fig. Q.32.1 Sequence diagram for online course reservation system**

• Each use case can be represented by one or more sequence diagrams.

• A large scale interaction can be shown using the sequence diagram.

• A separate sequence diagram can be prepared to represent the exceptional conditions of the use cases.

• The sequence diagram is normally created to cover the basic behavior of the system.

**Q.33 Draw sequence diagram for an inventory management system.**

**Ans. :**

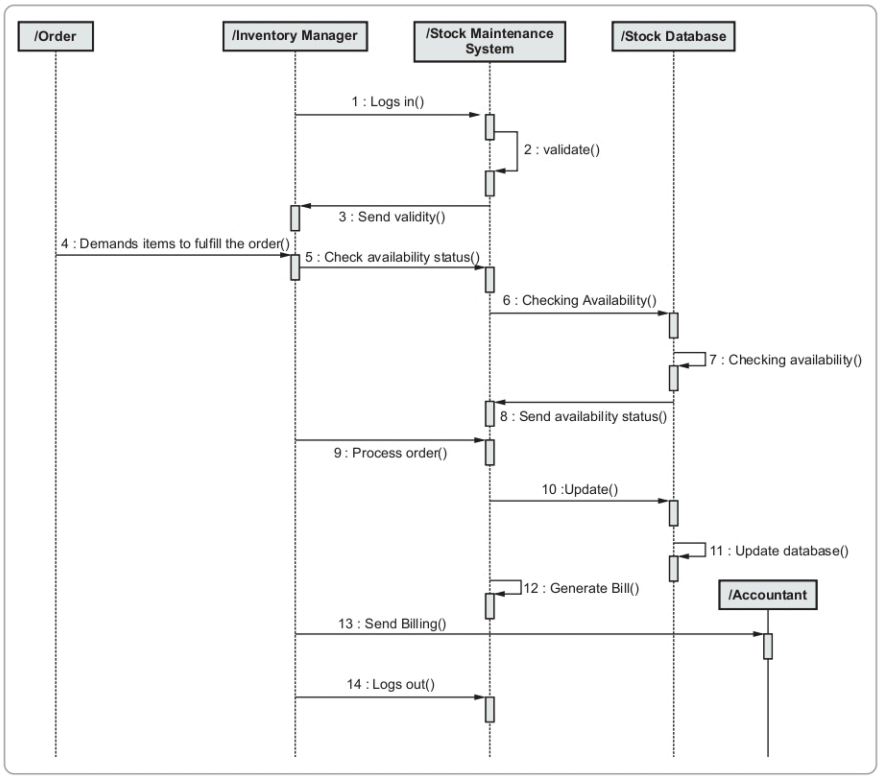



**Fig. Q.33.1 Sequence diagram for inventory system**

**Q.34 Draw a sequence diagrams that specifies the flow of control involved in initiating a simple, two party phone call.**

**Ans. :**

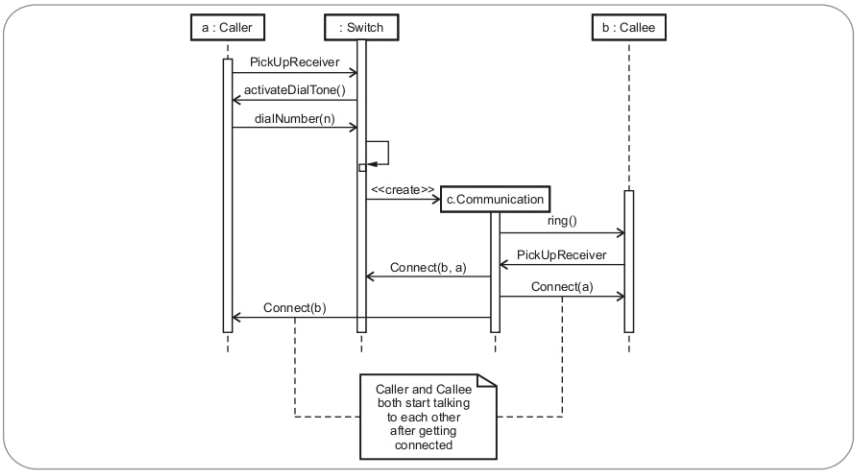



**Fig. Q.34.1 Sequence diagram for two party phone call**

## 8.11 : Collaboration Diagrams

**Q.35 Explain collaboration diagram with example.**

**Ans. :** • For representing the **structural organization** of objects the collaboration diagram is used.

- The **communicating objects** are placed as vertices in graph. The **links** that connect these vertices are the arcs of this graph. These links have the role name to identify them.
- The links can be adorned using **messages** that are sent and received by the objects.
- This gives the **flow of control** in when the objects collaborate with each other.

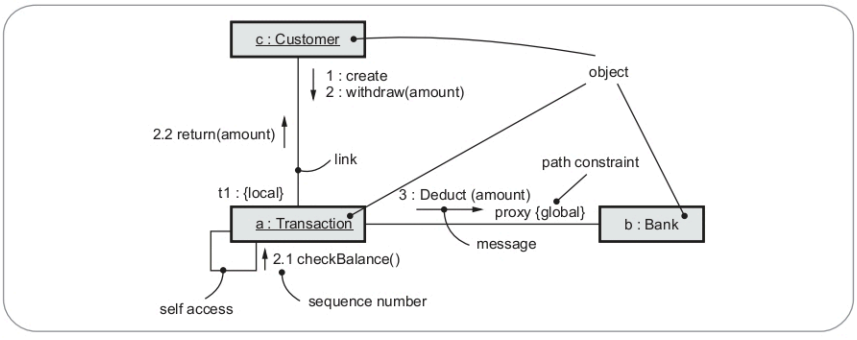



**Fig. Q.35.1 Collaboration diagram**

**Q.36 Difference between Sequence diagram and Collaboration diagram.**

**Ans. :**

| Sequence diagram | Collaboration diagram |
|---|---|
| The sequence diagram models the **lifeline of the objects.** The lifeline denotes the existence of the object between its creation and destruction. | The collaboration diagram represents **structural links** among the objects during there interactions. |
| The particular object can be destroyed explicitly in sequence diagram. | There is no facility to destroy some object. |
| The linking of the objects with other object can not be represented as local or global. | The {local} or {global} scope of the objects with other objects can be represented in this diagram. |

**Q.37 For a railway reservation system, draw possible sequence diagram and convert the same into a collaboration diagram.**

**Ans. :**

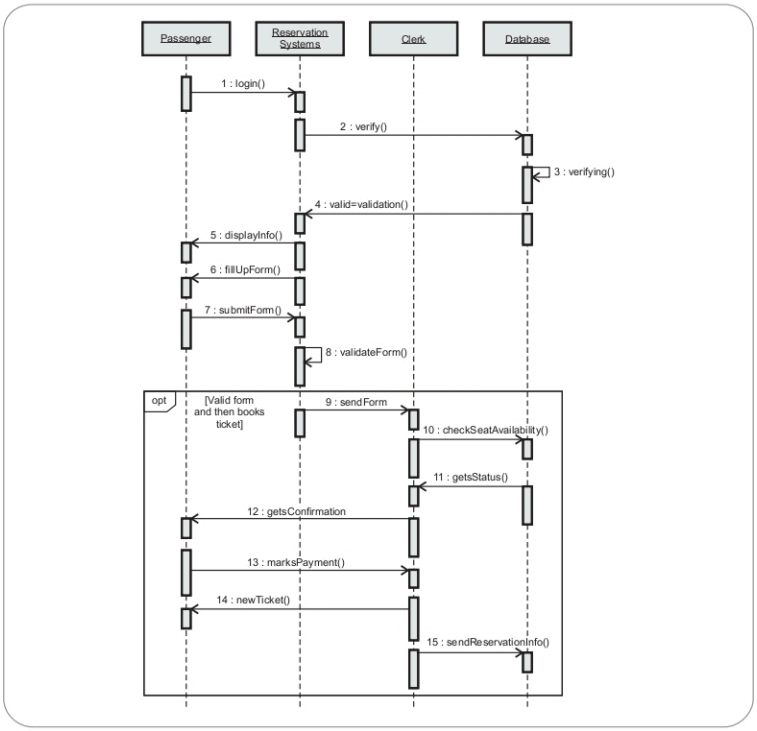



**Fig. Q.37.1 (a) Sequence diagram for railway reservation system**

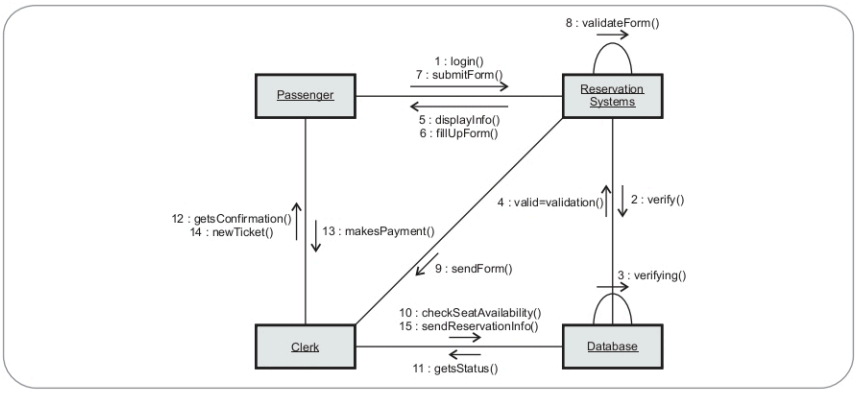



**Fig. Q.37.1 (b) Collaboration diagram for railway reservation system**

**Q.38 Draw a sequence diagram to show how a GUI interacts with other objects and also draw corresponding collaboration diagram.**

**Ans. :**

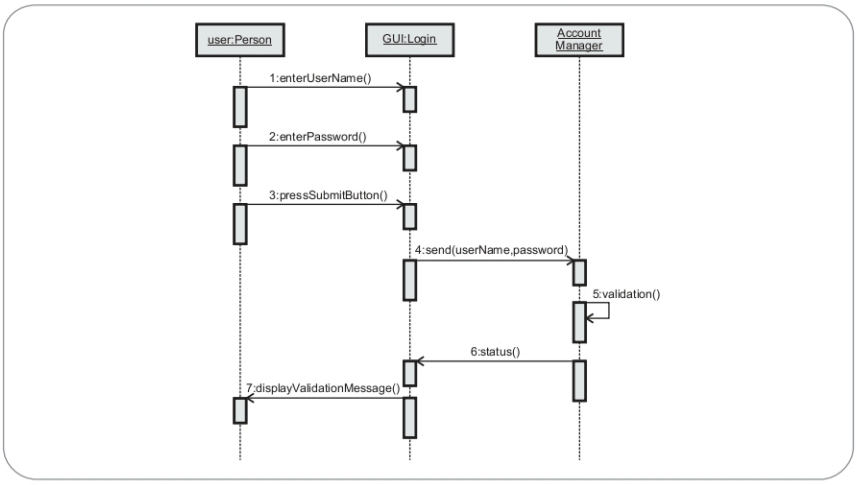



**Fig. Q.38.1 (a) Sequence diagram for interaction with GUI**

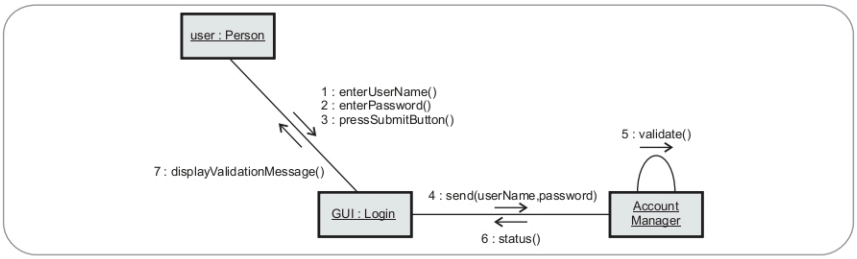



**Fig. Q.38.1 (b) Collaboration diagram for interaction with GUI**

**Q.39 Draw a collaboration diagram that specifies the flow of control involved in registering a new student at a school.**

**Ans. :**

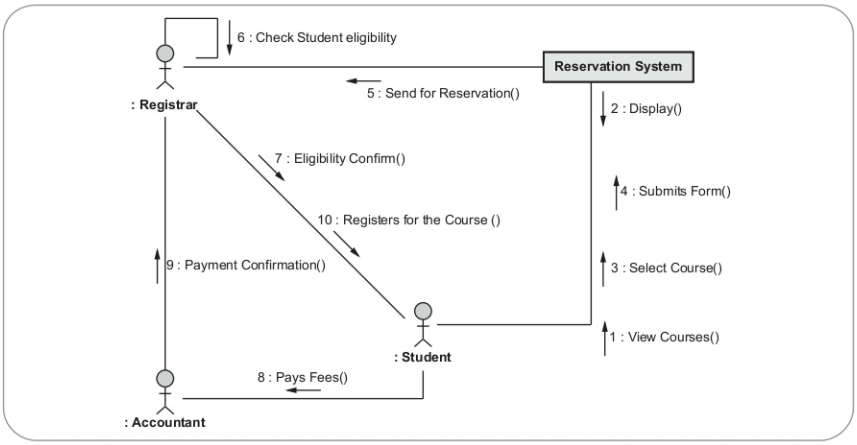



**Fig. Q.39.1 Collaboration diagram**

## 8.12 : Use Case Diagram

**Q.40 Explain Use case diagram with example.**

**Ans. :** Use case model focus on behavior of the system. It exposes functionalities that can be provided by the system to its users.

- The system is represented using the set of use cases and the set of actors.
- The **set of use cases** show the complete functionality of the system.
- The **set of actors** represent the objects that interact with the system in order to exploit the behavior.
- Use case diagram is a graphical notation used to represent the set of use cases and set of objects.

- The rectangular box is used to represent the **boundry** of the system.
- The oval is used to represent each use case.
- The stick man is used to denote the actor. The solid lines connect use cases to participating actors. For example -

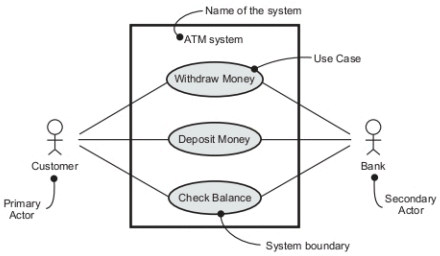



**Fig. Q.40.1 Use case diagram for ATM system**

**Q.41 Draw use case diagram for vending machine**

**Ans. :**

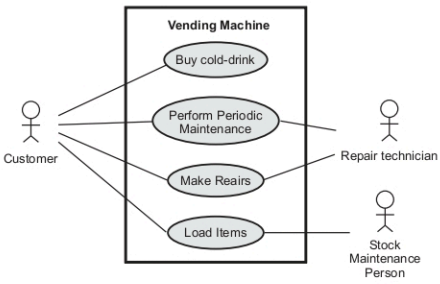


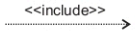

**Fig. Q.41.1 Use case diagram for vending machine**

**Q.42 Explain different types of relationships used in use case diagram.**

**Ans. :**

**(1) Include Relationship**

- The include relationship is used to show that the base case is obtained by the behavior of other use cases.
- Using the stereotype **<<include>>** the include relationship can be represented.
- When there are multiple steps to carry out a single task then that task is divided into the sub-functions and these sub-functions can be denoted by the use cases. It is denoted as -




**Example -**

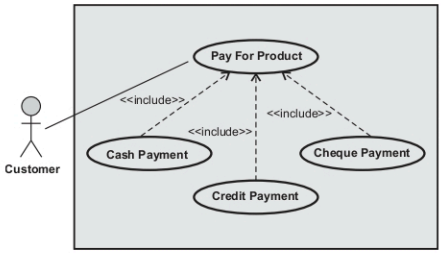



**Fig. Q.42.1 Use case for payment for the product (Include- relationship)**

- While designing the use case diagrams, using the **include** relationship, much fine details of the behavior must not be drawn.
- The factoring of use cases into more detailed use cases must be done to some appropriate level. That means each individual sub-use case must represent at least one functionality.

**(2) Extend Relationship**

- The extend relationship is used when an extended use case is connected to the **base use case.**
- The extend relationship adds incremental behavior to use case. It represents frequent situations in which some initial conditions occurs and when new features are added in the module.
- The extension relationship is often a fragment that means it cannot occur alone; rather it will occur with the base use case.
- It can be denoted as -


- The base use case can specify an insert location within the base class. These extension are called extension points. In most of the cases, an extend behavior occurs only when condition is true. Following Fig. Q.42.2 represents the extend relationship -

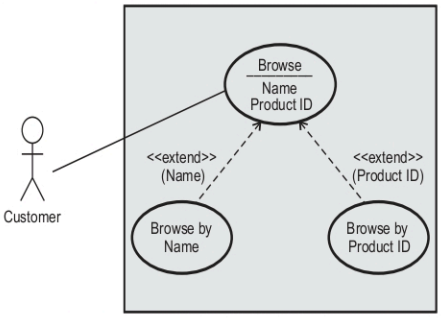



**Fig. Q.42.2 Use case for browsing product (Extend - relationship)**

**(3) Generalization Relationship**

- Generalization is a kind of relationship in which the child class inherits the properties of its parent class. Sometimes the behavior of the parent class can be overridden by the child class.
- Generalization can be represented by a solid directed line with large triangular arrowhead. It is denoted as



- The arrow head must be towards the parent class or base class.
- **Example :** In hospital management system, the patient gets admitted to the hospital. He can be OPD patient or IPD patient. The admission process for these both types can be different. Hence it can be represented using the generalization relationship.

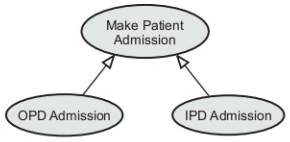



**Fig. Q.42.3 Example for generalization**

**Q.43 Describe use case "Validate User" in modeling an ATM system.**

**Ans. :** There are two actors for the Bank ATM system. The **Customer** and the **Bank**. There are two types of customers - current account holder and savings account holder.

Following are the steps by which the customer interacts with the ATM system -

1. Customer inserts the card.
2. The ATM system validates the card.
3. If there is valid card entered by the customer then the ATM system requests the customer for entering PIN, The customer enters the PIN.
4. The PIN entered by the customer is validated.
5. If the valid PIN is entered then only the customer is prompted for performing the transactions.
6. Customer can select the desired transaction such as Withdrawal of money, Deposition of fund or simply enquires for the available balance amount.

The use case for Validate User is as given below -

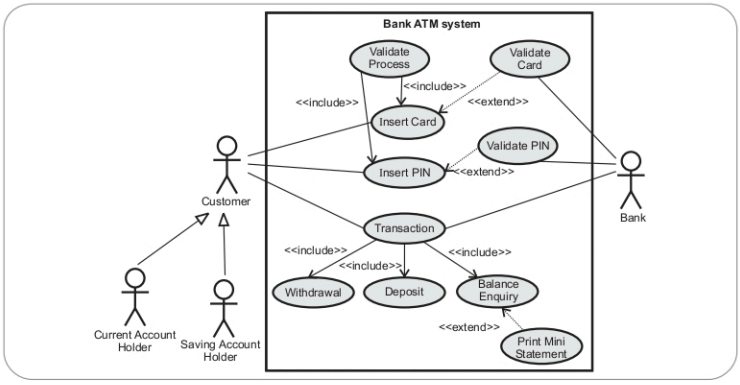



**Fig. Q.43.1 Validate User use case**

**Use case Name : Validate Process**

This use case is for validating the user interacting with the system. It consists of two use cases -

1. Insert Card 2. Enter PIN

For the card insertion the extended use case will be activated. By which the card is validated.

For the Enter PIN use case, the extended use case Validate PIN will be activated.

These validations can be performed by the Bank or by the authentication system which is associated with the bank.

**Q.44 Give use case diagram for hospital management system.**

**Ans. :**

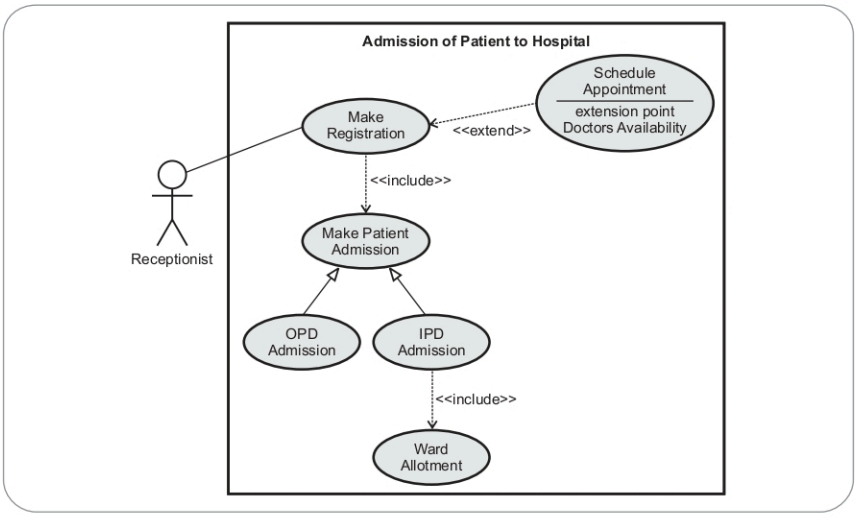



**Fig. Q.44.1 (a) Use case diagram for hospital management system**

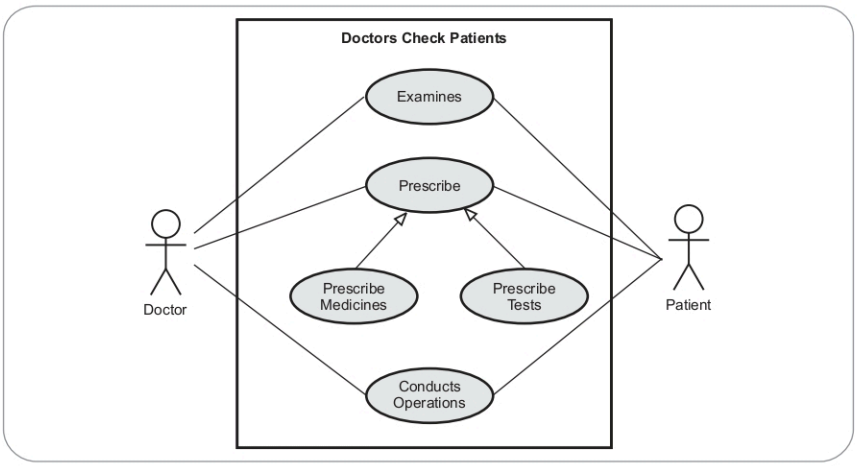



**Fig. Q.44.1 (b) Use case diagram for hospital management system**

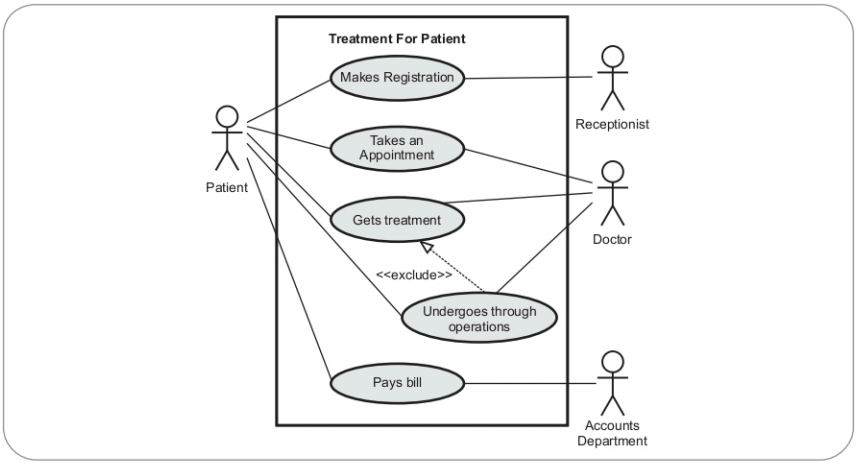



**Fig. Q.44.1 (c) Hospital management system**

**Q.45 Draw a use case diagram that depict the context of a credit card validation system. Explain briefly.**

**Ans. :**

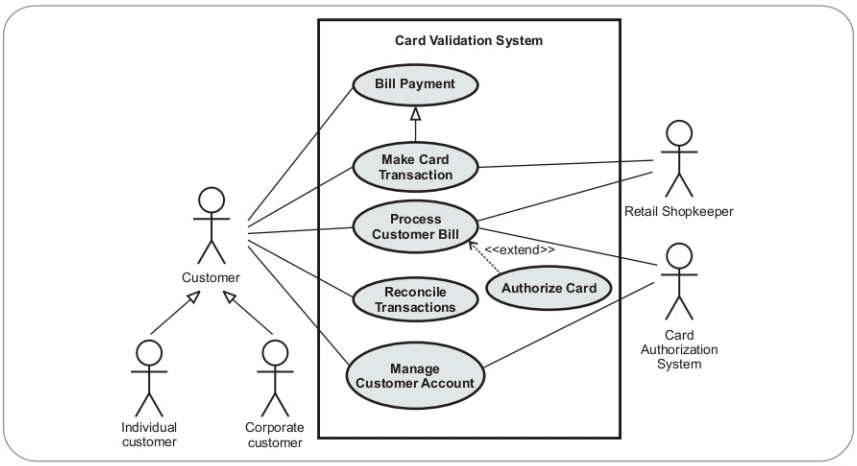



**Fig. Q.45.1 Use case diagram**

In the credit card validation system, there are various actors that surrounds the system. The Customer can be of Individual customer and Corporate Customer. There two more customers that are included in this transactions and those are Retail Shopkeeper and Card authorization system. Various functionalities involved are making card transaction, processing of bill, reconciling of transactions and customer account management.

**Q.46 Draw use case diagram to model the behavior of a cellular phone. Explain briefly.**

**Ans. :**

In above use case diagram, there are two important actors - user who operates the Cell phone and the Cellular network. The three important use cases are -

**Use case Name :** Make a phone call.

In this use case the activity of user makes a call is represented. While making a call user might make a conference call for some group communication.

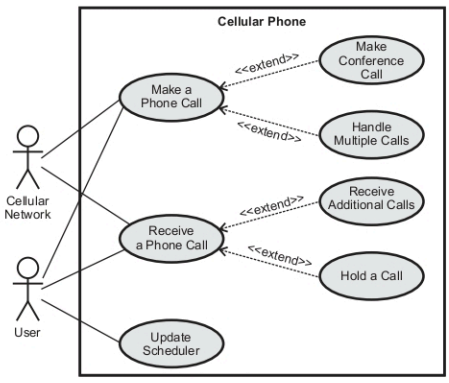



**Fig. Q.46.1 Use case model for cellular phone**

**Use case Name :** Receive a phone call

The receiving call can be attended in normal scenario. But user can receive some additional calls or he/she can hold some receiving call. These two activities are represented using the extended relationship.

**Use case Name :** Update scheduler

The user can update his/her schedule using calendar.

**Q.47 Draw Use case diagram for library management system.** ☞[JNTU : Dec.-19, Marks 5]

**Ans. :** Use cases demonstrate the functionalities of the system. Various use cases in the unified library application are -

- Login
- Search for items
- Browse the items
- Make Reservation
- Cancel Reservation
- Return Items
- Manage Titles
- Manage Items
- Mange Borrowers
- Validate Identity of browser

Following is the main control flow in the unified library system

1. The registered user logs in the system.
2. The borrower searches for the desired item,
3. He browses for the desired item.
4. If the desired title is not available then he reserves the item.
5. If the item is available then borrower checks out the item.
6. The borrower may cancel his reservation
7. The borrower returns the items already taken.

Following is a use case model that demonstrates the unified library system.

(See Fig. Q.47.1 on next page)

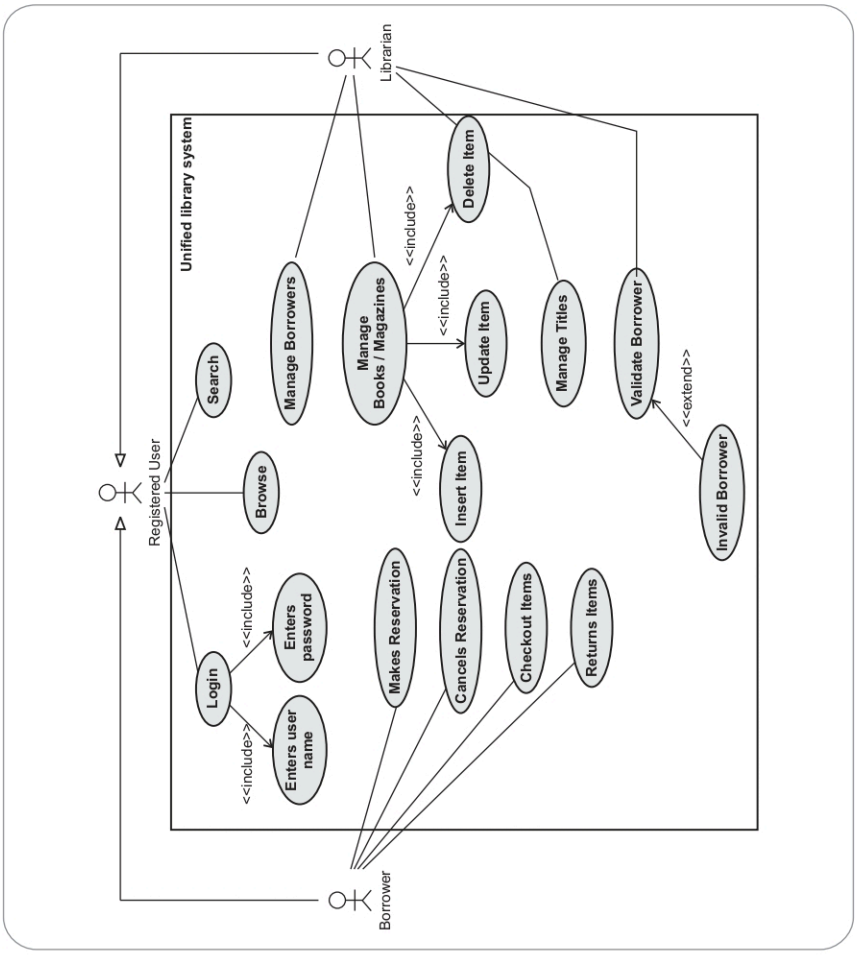



Fig. Q.47.1 Use case model for unified library application

## 8.13 : Component Diagram

**Q.48 Explain the component diagram with example.** ☞ [JNTU : Dec.-19, Marks 5]

**Ans. :** • Component diagram and deployment diagram are the two types of models that focus on the physical aspect of the object oriented system.

- The component diagram represents the organization and dependencies among the set of components.
- The component diagram is used to model the static implementation view of the system.
- It contains the physical things(components) such as executables, libraries, files, tables and documents.

- The **component diagrams are basically the class diagrams** which mainly focus on system's **component.**
- For example

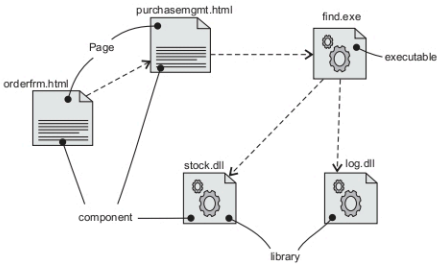



**Fig. Q.48.1 Component diagram**

# For Mid Term Exam



## Fill in the Blanks for Mid Term Exam

1. ___ is a process of translating analysis model into design model.
2. A _____ is a named collection of data that describes a data object. [JNTU : August-16]
3. In object-oriented design, the modules in the design represent ____
4. _____ is the process of changing a software system. [JNTU : August-16]
5. ____ and ___ are two qualitative criteria used for functional independence.
6. The weakest coupling is ____
7. The dashed line in design model represents the ___ between analysis and design
8. The type of cohesion in which task within a module need to be executed in some specific time span is called ___
9. The goal of good design is ____ cohesion and ___ coupling
10. Archetype describes the pattern which is used in designing the ____ system.
11. UML is mainly used for ___ design.
12. The Mean Time To Failure(MTTF) is a metric that is widely used to measure the ____.
13. The modularity criteria proposed by Meyer are - modular decomposability modular composability, understandability, ___ and ___.
14. Various elements of Data design are ___, ____ and ___.
15. A data flow diagram is mapped into program structure using transform mapping and/or _____.
16. _____ type of mapping is applied to an information flow that exhibits distinct boundaries between incoming and outgoing data.
17. The constructs fundamental to structured programming are sequence, _____ and repetition.
18. PDL stands for _____.

## Multiple Choice Questions for Mid Term Exam

**Q.1** In ___ coupling the global variables are used.

[a] Stamp [b] data
[c] common [d] content

**Q.2** Which of the following is a tool in design phase ?

[a] Abstraction
[b] Refinement
[c] Information hiding
[d] All of the above

**Q.3** Information hiding is to hide information from user ____.

[a] that is relevant to him,
[b] that is irrelevant to him
[c] that can be maliciously handled by him
[d] none of these

**Q.4** Design phase includes _____.

[a] data, architectural, and procedural designs only
[b] architectural, procedural and interface design only
[c] data, architectural and interface design only
[d] data, architectural interface and procedural design only

**Q.5** What incorporates data, architectural, interface, and procedural representations of the software ?

a design model
b user's model
c system image
d all of these

**Q.6** What is the meaning of functional cohesion ?

a Operations are part of single functional task and are placed in same procedures.
b All operations that access the same data are defined within one class.
c All operations that access the data from outside the module.
d None of the above

**Q.7** Which is the worst type of coupling ?

a Control coupling
b Data coupling
c Content coupling
d Stamp coupling

**Q.8** The purpose of user interface is ___

a to help users to communicate using windows, icons with computer system
b to convert program to machine language form
c transmit data to remote location
d all of these

**Q.9** The rules regarding the user interface design are known as _____.

a golden rule
b silver rule
c rule of thumb
d universal rule

**Q.10** Which of the following is golden rule for interface design ?

a Place the user in control
b Reduce the user's memory load
c Make the interface consistent
d All of the mentioned

**Q.11** In which type of interface users provide commands selecting from a menu ?

a command language interface
b Graphical user interface
c voice recognition user interface
d none of these

**Q.12** Stepwise refinement is _____ strategy suggested by Niklaus WIRTH.

a top down
b bottom up
c either top down or bottom up

**Q.13** Commonly used architectural styles are

a data centered architecture
b data flow architecture
c call and return architecture
d all of these

**Q.14** During architectural design at the initial stage ___ is prepared

a use case diagram
b context model
c component diagram
d none of these

**Q.15** Control systems may make use of the Environmental Control pattern, which is a general control pattern that includes _____processes.

a sensor
b actuator
c pipeline
d both sensor and actuator

**Q.16** A _____ view shows the system hardware and how software components are distributed across the processors in the system.

a physical
b logical
c process
d all of the mentioned

**Q.17** Which view in architectural design shows the key abstractions in the system as objects or object classes?

a physical
b development
c logical
d process

**Q.18** _____ is an indication of the relative functional strength of a module.

**[JNTU : August-16]**

a Coupling
b Cohesion
c Coordination
d Quality

**Q.19** Coupling is a measure of _____.

a Relative functional strength
b Interdependence among module
c Both of the above
d None of the above

**Q.20** _____ classes represent data stores.

**[JNTU : August-16]**

a Process
b User interface
c System
d Persistent

**Answer Keys for Fill in the Blanks :**

| | | | |
|---|---|---|---|
| 1. | Software designing | 2. | Data abstraction |
| 3. | Data abstraction | 4. | Refactoring |
| 5. | Cohesion, Coupling | 6. | data coupling |
| 7. | boundary | 8. | temporal cohesion |
| 9. | high, low | 10. | target |
| 11. | object oriented | 12. | reliability |
| 13. | modular continuity and modular protection | 14. | Data object, Databases and Data Warehouses |
| 15. | transaction mapping | 16. | Transform mapping |
| 17. | condition | 18. | program design language |

**Answer Keys for Multiple Choice Questions**

| | | | |
|---|---|---|---|
| 1. | c | 2. | d |
| 3. | c | 4. | d |
| 5. | a | 6. | a |
| 7. | c | 8. | a |
| 9. | a | 10. | a |
| 11. | b | 12. | a |
| 13. | d | 14. | b |
| 15. | d | 16. | a |
| 17. | c | 18. | b |
| 19. | b | 20. | d |

*END...*

UNIT - IV

# 9 Testing Strategies

## 9.1 : A strategic Approach to Software Testing

**Q.1 What is testing ?**
[JNTU : Part A, Dec.-17, Marks 2]

**Ans. : Definition : Software testing** 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.

**Q.2 Enlist three objectives of testing process.**
[JNTU : Part A, Marks 3]

**Ans. :** According to **Glen Myers** the testing objectives 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.

**Q.3 What is the criteria for completion of testing ? Explain.** [JNTU : Part B, May-13, Marks 5]

**Ans. :** Testing is a complex activity in the software systems. It is said that testing is an endless process and complete testing not possible for almost all the projects. But there are some common factors that are required to decode when to stop testing.

1. When the testing cost is increasing and if it is more than the project cost then it is enough to test.
2. If the project deadline and testing deadline is already crossed.
3. After completion of critical or key test cases one can stop testing.
4. If the project is meeting functional coverage, code coverage or satisfying the client requirements at some point.
5. When high priority bugs are resolved and defect rates fall below certain specified level.
6. When project progresses through alpha and beta testing.

**Q.4 Discuss the need for testing a developed software.** [JNTU : Part B, Dec.-13, Marks 8]

**Ans. :** • Generally, testing is a process that requires more efforts than any other software engineering activity.

- Testing is a set of activities that can be planned in advance and conducted systematically.
- If it is conducted haphazardly, then only time will be wasted and more even worse errors may get introduced.
- This may lead to have many undetected errors in the system being developed. Hence performing testing by adopting systematic strategies is very much essential in during development of software.

**Q.5 What are the characteristics of good tester ?**
[JNTU : Part A, Marks 2]

**Ans. :** Following are the characteristics of good tester -

1. The tester must be able to understand the software. He should be in a position to find out high probability errors.
2. The tester must not conduct two different tests for the same purpose.
3. The tester must be able to write simple test cases.
4. The tester should conduct the tests which should have highest likehood of uncovering errors.

**Q.6 What is meant by testing ? Discuss about software testing objectives and testing phases**
[JNTU : Part B, Dec.-14, Marks 15]

**Ans. : Testing -** Refer Q.1.

**Testing Objective -** Refer Q.2.

**Testing Phases -** Various testing activities are

**1. Test planning :** The test plan or test script is prepared. These are generated from requirements analysis document (for black box) and program code (for white box).

**2. Test case design :** The goal of test case design is to create a set of tests that are effective in testing.

**3. Test execution :** The test data is derived through various test cases in order to obtain the test result.

**4. Data collection :** The test results are collected and verified.

**5. Effective evaluation :** All the above test activities are performed on the software model and the maximum number of errors are uncovered.

**Q.7 What is software testing ? Explain test characteristics.** [JNTU : Dec.-19, Marks 5]

**Ans. :** Refe Q.1 and Q.5.

## 9.2 : Testing Strategies For Conventional Software

**Q.8 What is the objective of unit testing ?**
[JNTU : Part A, Marks 2]

**Ans. :** The objective of unit testing is to test individual components independently to ensure their quality. Thus the focus is to uncover the errors in design and implementation.

**Q.9 List out the data structure errors identified during the unit testing.** [JNTU : Part A, Marks 2]

**Ans. :** Various data structure errors that can be identified during the unit testing are -

1. Incorrect arithmetic precedence.
2. Mixed mode operations.
3. Precision inaccuracy.
4. Comparison of different data types.

**Q.10 What are the testing strategies for conventional software ? Explain them in detail.**
[JNTU : Part B, Dec.-12, Marks 15, Dec.-17, Marks 5]

**Ans. :**

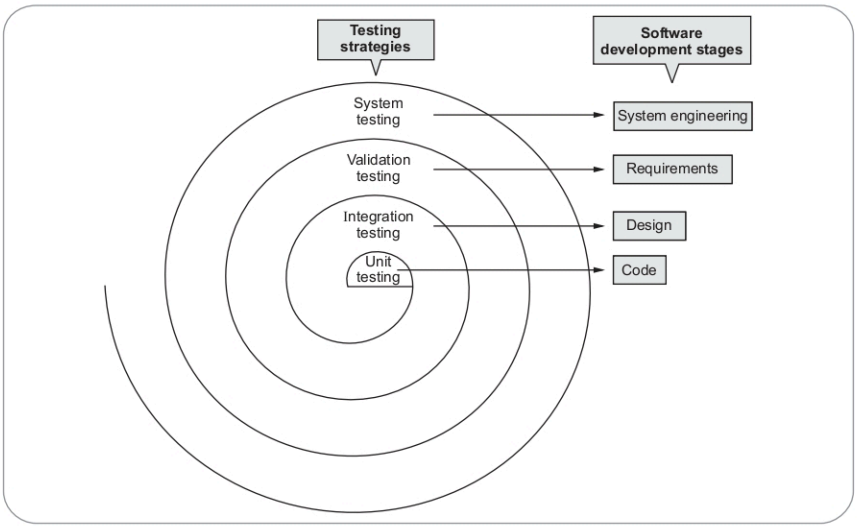



**Fig. Q.9.1 Testing strategy**

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

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.

**Q.11 Briefly discuss about unit testing strategy**
**[JNTU : Part B, Marks 8]**

**OR Discuss clearly the unit test procedures and environment.** **[JNTU : Part B, April-11, Marks 8]**

**Ans. :** • 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 minimal data manipulation if required. This is illustrated by following Fig. Q.11.2.
  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.

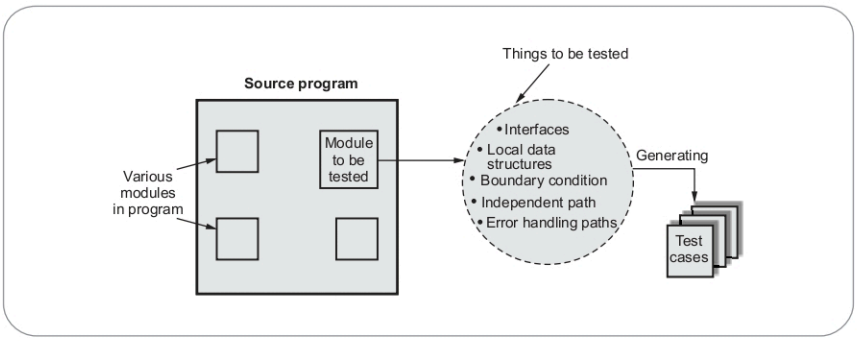



**Fig. Q.11.1 Unit testing**

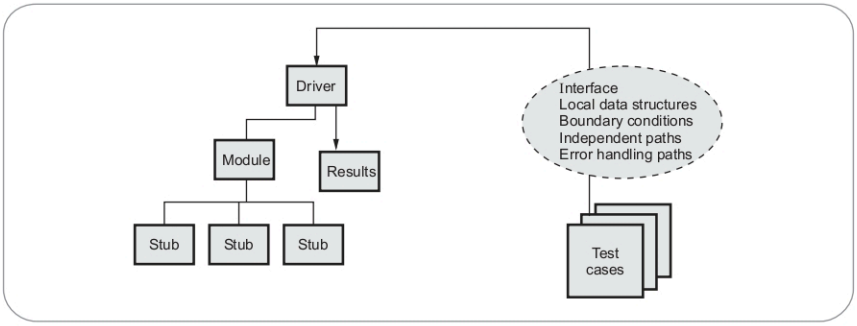



**Fig. Q.11.2 Unit testing environment**

**Q.12 Briefly discuss about integration testing strategies.** **[JNTU : Part B, Dec.-16, Marks 5]**

**Ans. :** • A group of dependent components are tested together to ensure their quality of their integration unit.

- The objective is to take unit tested components and build a program structure that has been dictated by software design.
- The focus of integration testing is to uncover errors in :
  - Design and construction of software architecture.
  - Integrated functions or operations at subsystem level.
  - Interfaces and interactions between them.
  - Resource integration and/or environment integration.
- The integration testing can be carried out using two approaches.
  1. The non-incremental integration
  2. Incremental integration

**Advantage of big-bang** : This approach is simple.

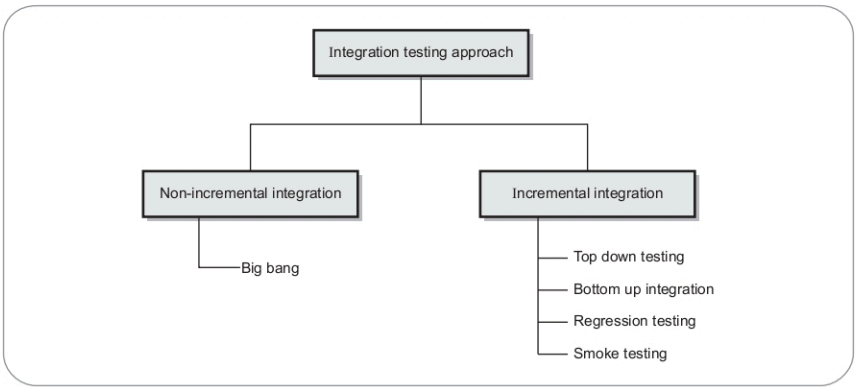



**Fig. Q.12.1 Integration testing approach**

**Disadvantages :**

1. It is hard to debug.
2. It is not easy to isolate errors while testing.
3. In this approach it is not easy to validate test results.
4. After performing testing, it is impossible to form an integrated system.

- An incremental construction strategy includes
  - Top down integration
  - Bottom up integration
  - Regression testing
  - Smoke testing

**Q.13 What is big-bang approach of integration testing ?** ☞[JNTU : Part A, Marks 3]

**Ans. :** The non-incremental integration is given by the "**big bang**" approach. All components are combined in advance. The entire program is tested as a whole. And chaos usually results. A set of errors is tested as a whole. Correction is difficult because isolation of causes is complicated by the size of the entire program. Once these errors are corrected new ones appear. This process continues infinitely.

**Q.14 What is integration testing ? Discuss about top-down integration.**
☞[JNTU : Part B, April-11, Marks 8]

**Ans. : Integration testing -** Refer Q.12.

**Top down integration -**

- Top down testing is an incremental approach in which modules are integrated by moving down through the control structure.
- Integration process can be performed using following steps.
  1. The main control module is used as a test driver and the stubs are substituted for all modules directly subordinate to the main control module.
  2. Subordinate stubs are replaced one at a time with actual modules using either depth first or breadth first method.
  3. Tests are conducted as each module is integrated.
  4. On completion of each set of tests, another stub is replaced with the real module.
  5. Regression testing is conducted to prevent the introduction of new errors.

**For example :**

In top down integration if the depth first approach is adopted then we will start integration from module M1 then we will integrate M2 then M3, M4, M5, M6 and then M7.

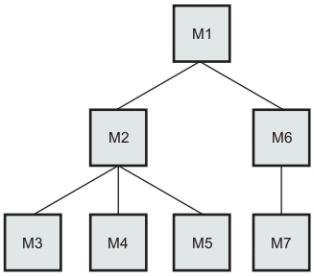



**Fig. Q.14.1 Program structure**

**Q.15 Discuss the steps in bottom up integration.**
☞[JNTU : Part B, Dec.-10, Marks 6]

**Ans. :** In bottom up integration the modules at the lowest level are integrated first, then integration is done by moving upward through control structure.

The bottom up integration process can be carried out using following steps.

1. Low-level modules are combined into clusters that perform a specific software subfunction.
2. A driver program is written to co-ordinate test case input and output.
3. The whole cluster is tested.
4. Drivers are removed and clusters are combined moving upward in the program structure.

**For example :**

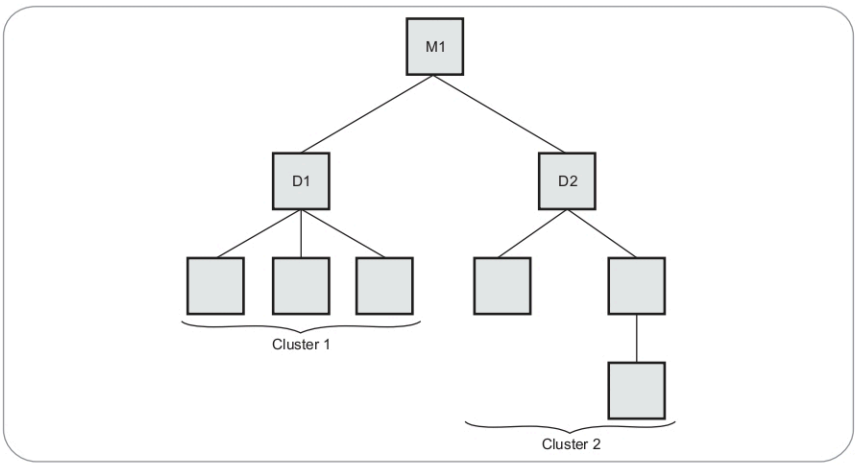



**Fig. Q.15.1 Bottom up integration testing**

**Q.16 Compare top down and bottom up integration testing.** [JNTU : Part A, Marks 3]

**Ans. :**

| Sr. No. | Top-down integration | Bottom-up integration |
|---|---|---|
| 1 | The major controls or decisions are verified at early stage itself. | After integrating all the components at the bottom level, major controls or decisions can be verified. |
| 2 | Testing can be performed from the early stage. | Individual components can be tested at higher level, but after integration of low level components into cluster testing is required. |
| 3 | In top-down testing, testing stubs are created. | In bottom-up testing, test drivers are required. |
| 4 | Working system can be available at early stage. | Working system is available only after integrating all the components. |

**Q.17 Describe in detail - Validation testing.** [JNTU : Part B, Feb.-10, Marks 7]

**Ans. :** • The integrated software is tested based on requirements to ensure that the desired product is obtained.

- In validation testing the main focus is to uncover errors in
  - System input/output
  - System functions and information data
  - System interfaces with external parts
  - User interfaces
  - System behaviour and performance
- Software validation can be performed through a series of **black box tests**.
- After performing the validation tests there exists two conditions.
  1. The function or performance characteristics are **according** to the **specifications** and are accepted.
  2. The requirement specifications are derived and the deficiency list is created. The **deficiencies** then can be **resolved** by establishing the proper communication with the customer.

- Finally in validation testing a review is taken to ensure that all the elements of software configuration are developed as per requirements. This review is called configuration review or audit.

**Q.18 Describe in detail - System testing.**
**[JNTU : Part B, Feb.-10, Marks 9, Dec.-17, Marks 5]**

**Ans. :** • The system test is a series of tests conducted to fully the computer based system.

- Various types of system tests are -
  - **Recovery testing :** Recovery testing is intended to check the system's ability to recover from failures.
  - **Security testing :** Security testing verifies that system protection mechanism prevent improper penetration or data alteration.
  - **Stress testing :** Determines breakpoint of a system to establish maximum service level.In stress testing the system is executed in a manner that demands resources in abnormal quantity, frequency or volume.
  - **Performance testing :** Performance testing evaluates the run time performance of the software, especially real time software.
- The main focus of such testing is to test -
  1. System functions and performance.
  2. System reliability and recoverability (recovery test).
  3. System installation (installation test).
  4. System behaviour in the special conditions (stress test).
  5. System user operations (acceptance test/alpha test).
  6. Hardware and software integration and collaboration.
  7. Integration of external software and the system.

**Q.19 Write short note on : Validation test criteria.**
**[JNTU : Part B, Feb.-10, Marks 4]**

**Ans. :** Validation test criteria is achieved through series of black box testing. Test plan and test procedure are designed to check -

(1) Requirements are satisfied or not.

(2) All behavioral characteristics are achieved or not.

(3) All performance requirements are attainable or not.

(4) Document is correct or not.

**Q.20 Why does software fail after it has passed from acceptance testing ?** **[JNTU : Part A, Marks 2]**

**Ans. :** (1) During acceptance testing, the random input is used for testing. This may lead to the situation that some input values that may cause failure go unhandled.

(2) The practical problem with acceptance testing is that it is time consuming. Hence in order to keep testing cost low, there are restricted number of test cases.

Hence there are high chances of software failure even after passing the acceptance testing.

**Q.21 Write short note on : Alpha and Beta testing.**
**[JNTU : Part B, Feb.-10, Marks 4]**

**Ans. : 1. Alpha test** - The alpha testing is a testing in which the version of complete software is tested by the customer under the supervision of developer. This testing is performed at developer's site. The software is used in natural setting in presence of developer. This test is conducted in controlled environment.

2. **Beta test** - The beta testing is a testing in which the version of software is tested by the customer without the developer being present. This testing is performed at customer's site. As there is no presence of developer during testing, it is not controlled by developer. The end user records the problems and report them to developer. The developer then makes appropriate modification.

**Q.22 Write short note on : Recovery testing.**
**[JNTU : Part B, Feb.-10, Marks 4]**

**Ans. :** • Recovery testing is intended to check the system's ability to recover from failures.

- In this type of testing the software is forced to fail and then it is verified whether the system recovers properly or not.
- For automated recovery then reinitialization, checkpoint mechanisms, data recovery and restart are verified.

**Q.23 Write short note on : Security testing.**
[JNTU : Part B, Feb.-10, Marks 4]

**Ans. :** • Security testing verifies that system protection mechanism prevent improper penetration or data alteration.

- It also verifies that protection mechanisms built into the system prevent intrusion such as unauthorized internal or external access or willful damage.
- System design goal is to make the penetration attempt more costly than the value of the information that will be obtained.

**Q.24 Write a note on : Regression testing.**
[JNTU : Part B, Dec.-10, Marks 4]

**Or Regression testing is an important strategy for reducing "Side Effects". Discuss.**
[JNTU : Part B, Dec.10, Marks 4]

**Ans. :** • Regression testing is used to check for defects propagated to other modules by changes made to existing program. Thus regression testing is used to reduce the side effects of the changes.

- There are three different classes of test cases involved in regression testing -
  - Representative sample of existing test cases is used to exercise all software functions.
  - Additional test cases focusing software functions likely to be affected by the change.
  - Tests cases that focus on the changed software components.
- After product had been deployed, regression testing would be necessary because after a change has been made to the product an error that can be discovered and it should be corrected. Similarly for deployed product addition of new feature may be requested and implemented. For that reason regression testing is essential.

**Q.25 Write a note on : Smoke testing.**
[JNTU : Part B,Dec.-10, Marks 4]

**Ans. :** • The smoke testing is a kind of integration testing technique used for time critical projects wherein the project needs to be assessed on frequent basis.

- Following activities need to be carried out in smoke testing -
  1. Software components already translated into code are integrated into a "build". The "build" can be data files, libraries, reusable modules or program components.
  2. A series of tests are designed to expose errors from build so that the "build" performs its functioning correctly.
  3. The "build" is integrated with the other builds and the entire product is smoke tested daily.

**Smoke testing benefits**

1. Integration risk is minimized.
2. The quality of the end product is improved.
3. Error diagnosis and correction are simplified.
4. Assessment of progress is easy.

**Q.26 Bottom up integration eliminates the need for complex stubs. Discuss.**
[JNTU : Part B, Dec.-10, Marks 6]

**Ans. :** Refer Q.15.

## 9.3 : Black Box Testing

**Q.27 Define Black box testing.**
[JNTU : Dec.-19, Marks 5]

**Ans. :** • The black box testing is used to demonstrate that the software functions are operational. As the name suggests in black box testing it is tested whether the input is accepted properly and output is correctly produced.

- The major focus of black box testing is on functions, operations, external interfaces, external data and information.

**Q.28 Enlist different black box testing methods.**
[JNTU : Part A, Marks 3]

**Ans. :** Various techniques of black box testing are as follows -

(1) Equivalence testing method

(2) Boundary value analysis method

(3) Graph based testing method

**Q.29 Explain the equivalence testing method in detail.** [JNTU : Part B, Marks 5]

**OR Explain black box testing. Give an account of equivalence partitioning and boundary value analysis techniques.** [JNTU : Dec.-19, Marks 5]

**Ans. :** • It is a black box technique that divides the input domain into classes of data. From this data test cases can be derived.

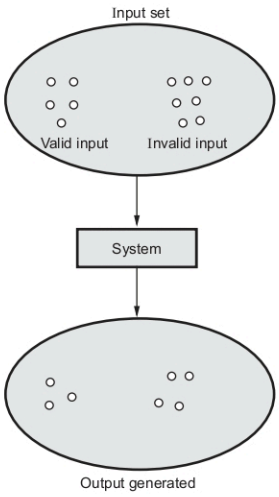



**Fig. Q.29.1**

- An ideal test case uncovers a class of errors that might require many arbitrary test cases to be executed before a general error is observed.
- In equivalence partitioning the equivalence classes are evaluated for given input condition. Equivalence class represents a set of valid or invalid states for input conditions.
- Equivalence class guidelines can be as given below :
  - If input condition specifies a range, one valid and two invalid equivalence classes are defined.
  - If an input condition requires a specific value, one valid and two invalid equivalence classes are defined.
  - If an input condition specifies a member of a set, one valid and one invalid equivalence class is defined.
  - If an input condition is Boolean, one valid and one invalid equivalence class is defined.

**For example :**

**Area code :** Input condition, Boolean - The area code may or may not be present.

**Input condition, range -** Value defined between 200 and 700.

**Password : I**nput condition, Boolean - A password may or may not be present.

**Input condition, value -** Seven character string.

**Command :** Input condition, set - Containing commands noted before.

**Q.30 What is boundary value analysis method ? Explain.** **[JNTU : Part B, Dec.-19, Marks 5]**

**Ans. :** • Boundary value analysis is done to check boundary conditions.

- A boundary value analysis is a testing technique in which the elements at the edge of the domain are selected and tested.
- Using boundary value analysis, instead of focusing on input conditions only, the test cases from output domain are also derived.
- Boundary value analysis is a test case design technique that complements equivalence partitioning technique.

**For example :**

Integer D with input condition [– 2, 10],

Test values : – 2, 10, 11, – 1, 0

If input condition specifies a number values, test cases should developed to exercise the minimum and maximum numbers. Values just above and below this min and max should be tested.

Enumerate data E with input condition :
{2, 7, 100, 102}

Test values : 2, 102, – 1, 200, 7

**Q.31 What are the guidelines used for boundary value analysis technique ?**

**Ans. :** Guidelines for boundary value analysis technique are

1. If the input condition specified the range bounded by values x and y, then test cases should be designed with values x and y. Also

test cases should be with the values above and below x and y.

2. If input condition specifies the number of values then the test cases should be designed with minimum and maximum values as well as with the values that are just above and below the maximum and minimum should be tested.
3. If the output condition specified the range bounded by values x and y, then test cases should be designed with values x and y. Also test cases should be with the values above and below x and y.
4. If output condition specifies the number of values then the test cases should be designed with minimum and maximum values as well as with the values that are just above and below the maximum and minimum should be tested.
5. If the internal program data structures specify such boundaries then the test cases must be designed such that the values at the boundaries of data structure can be tested.

**Q.32 What are merits and demerits of BVA ?**

**Ans. : Advantages :**

1) This method provides clear guideline on determining test cases.
2) It is a good testing technique for exposing potential user interface problems.

**Disadvantages :**

1) It cannot test all possible inputs.
2) It does not test dependencies between combinations of inputs.

**Q.33 Write short note on - Graph based testing**

☞[JNTU : Part B]

**Ans. :** • In the graph based testing, a graph of objects present in the system is created.

- The graph is basically a collection of nodes and links. Each node represents the object that is participating in the software system and links represent the relationship among these objects.
- The node weight represents the properties of object and link weight represents the properties or characteristics of the relationship of the objects.

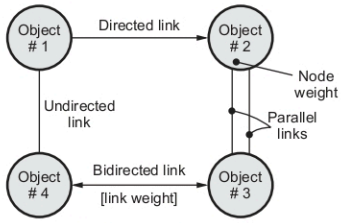



**Fig. Q.33.1 Graph notations**

- After creating the graph, important objects and their relationships are tested.

## 9.4 : White Box Testing

**Q.34 Explain white box testing with an example.**

☞[JNTU : Part B]

**OR Give an overview of white-box testing techniques with help of flow graph.**

☞[JNTU : Dec.-19, Marks 5]

**Ans. : 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.
- Thus major focus of white box testing is on internal structures, logic paths, control flows, data flows, internal data structures, conditions, loops, etc.

**Example :** Refer Q.35.

**Q.35 Explain Basis Path testing using flow graph analysis in detail.** ☞[JNTU : Part B]

**Ans. : Basis Path Testing :** Path testing is a structural testing strategy. This method is intended to exercise every independent execution path of a program atleast once.

Following are the steps that are carried out while performing path testing.

**Step 1 : Design the flow graph for the program or a component.**

Flow graph is a graphical representation of logical control flow of the program. Such a graph consists of circle called a ***flow graph node*** which basically represents one or more procedural statements and arrow called as ***edges*** or ***links*** which basically represent control flow. In this flow graph

the areas bounded by nodes and edges are called ***regions***. Various notations used in flow graph are -

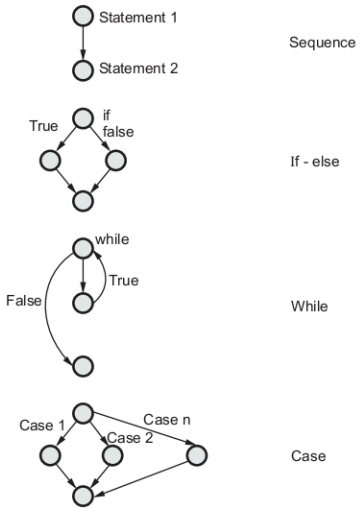



**Fig. Q.35.1**

**For example :** Following program is for searching a number using binary search method. Draw a flow graph for the same.

```
void search (int key, int n, int a [ ])
  {
     int mid;
1)   int bottom = 0;
2)   int top = n – 1;
3)   while (bottom <=top)
4)   { mid = (top + bottom) / 2;
5)   if (a [mid] = = key)
     {
6)   printf ("Element is present");
7)   return;
     } // end of if
     else
     {
8)   if (a [mid] < key)
9)   bottom = mid + 1;
     else
10)     top = mid – 1;
     } // end of else
     } // end of while
11)  } // end of search
```

The flow graph will be

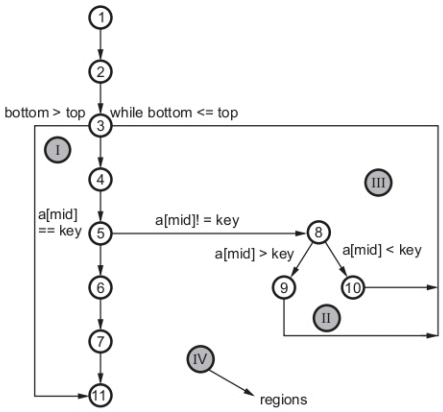



**Fig. Q.35.2**

**Step 2 : Calculate the cyclomatic complexity.**

The cyclomatic complexity can be computed by three ways.

1) Cyclomatic complexity = Total number of regions in the flow graph = **4** (note that in above flow graph regions are given by shaded roman letters).

2) Cyclomatic complexity = E – N + 2

   = 13 edges – 11 nodes + 2

   = 2 + 2 = **4**

3) Cyclomatic complexity = P + 1 = 3 + 1 = **4**. There are 3 predicate (decision making) nodes : Nodes 3, 5 and 8.

**Step 3 : Select a basis set of path**

The basis paths are

Path 1 : 1, 2, 3, 4, 5, 6, 7, 11

Path 2 : 1, 2, 3, 11

Path 3 : 1, 2, 3, 4, 5, 8, 9, 3 …

Path 4 : 1, 2, 3, 4, 5, 8, 10, 3 …

**Step 4 : Generate test cases for these paths.**

After computing cyclomatic complexity and finding independent basis paths, the test cases has to be executed for these paths.

**Q.36 What is cyclomatic complexity ? Define steps to find the cyclomatic complexity using flow graph ?** [JNTU : Part B]

**Ans. : Cyclomatic Complexity :** • Cyclomatic complexity is a software metric used to indicate the complexity of a program.

- It is a quantitative measure of the number of linearly independent paths through a program's source code.
- It is calculated using following formula -

Cyclomatic Complexity = E – N + 2 where

E is total number of edges

N is total number of nodes or vertices.

Refer Q.37.

**Q.37 Consider the program given below**

```
void main()
int i,j,k;
readln (i,j,k);
if( (i < j) || (i > k))
{
writeln("then part");
if(j < k)
writeln("j less then k");
else writeln ("j not less then k");
}
else writeln("else Part");
}
```

**i) Draw the flow graph.**
**ii) Determine the cyclomatic complexity.**
**iii) Arrive at all the independent paths.**
[JNTU : Part B]

**Ans. :** i) We will number the source code statements

```
    void main()
    {
1.  int i,j,k;
2.  readln (i,j,k);
3,4 if( (i < j) || (i > k))
    {
5.  writeln("then part");
6.  if(j < k)
7.  writeln("j less then k");
8.  else writeln ("j not less then k");
9.  }
10. else writeln("else Part");
11. }
```

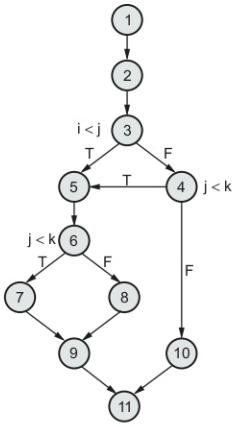



**Fig. Q.37.1**

ii) Cylomatic complexity = E – N + 2 = 13 – 11 + 2
= 4

iii) The four independent paths are

Path 1 :1, 2, 3, 5, 6, 7, 9, 11

Path 2 :1, 2, 3, 5, 6, 8, 9, 11

Path 3 :1, 2, 3, 4, 5, 6, 7, 9, 11

Path 4 :1, 2, 3, 4, 10, 11

**Q.38 What is meant by white box testing ? What are the aspects to be considered while generating of white box test cases**
[JNTU : Part B, April-11, Marks 10]

**Ans. :** Refer Q.34, Q.35 and Q.36.

**Q.39 Differentiate between Black box and white box testing.** [JNTU : Part B, Dec.- 16, Marks 5]

**Ans. :**

| Sr. No. | Black Box Testing | White Box Testing |
|---|---|---|
| 1. | Black box testing is the software testing method which is used to test the software without knowing the internal structure of code or program. | White box testing is the software testing method in which internal structure is being known to tester who is going to test the software. |
| 2. | This type of testing is carried out by testers. | Generally, this type of testing is carried out by software developers. |
| 3. | Implementation knowledge is not required to carry out Black Box Testing. | Implementation knowledge is required to carry out White Box Testing. |
| 4. | Programming knowledge is not required to carry out Black Box Testing. | Programming knowledge is required to carry out White Box Testing |
| 5. | Testing is applicable on higher levels of testing like System Testing, Acceptance testing. | Testing is applicable on lower level of testing like Unit Testing, Integration testing |
| 6. | Black box testing means functional test or external testing. | White box testing means structural test or interior testing. |
| 7. | Black Box testing can be started based on Requirement Specifications documents. | White Box testing can be started based on Detail Design documents. |
| 8. | The Functional testing, Behavior testing, Close box testing is carried out under Black Box testing. | The Structural testing, Logic testing, Path testing, Loop testing, Code coverage testing, Open box testing is carried out under White Box testing. |

## 9.5 : Validation Testing

**Q.40 Differentiate between verification and validation.** [JNTU : Part A, Dec.-16, Marks 2]

**Ans. :**

| Verification | Validation |
|---|---|
| **Verification** refers to the set of activities that ensure software **correctly implements** the specific function. | **Validation** refers to the set of activities that ensure that the software that has been built is **traceable to customer requirements.** |
| Are we building the product right ? | Are we building the right product ? |
| After a valid and complete specification the verification starts. | Validation begins as soon as project starts. |
| The verification is conducted to ensure whether software meets specification or not. | Validation is conducted to show that the user requirements are getting satisfied. |

## 9.6 : System Testing

**Q.41 What is regression testing ? Give example..** [JNTU : Part A, Dec.-16, Marks 3]

**Ans. :** Refer Q.24.

## 9.7 : Art of Debugging

**Q.42 Explain in detail about the art of debugging.** [JNTU : Part B, Dec.-10, Marks 16]

**OR What is debugging ? Discuss the debugging strategies.** [JNTU : Part B, April-11, Marks 8]

**Ans. :** • Debugging is a process of removal of a defect. It occurs as a consequence of successful testing.

- Debugging process starts with execution of test cases.
- The actual test results are compared with the expected results.
- The debugging process attempts to find the lack of correspondence between actual and expected results.
- The suspected causes are identified and additional tests or regression tests are performed to make the system to work as per requirement.

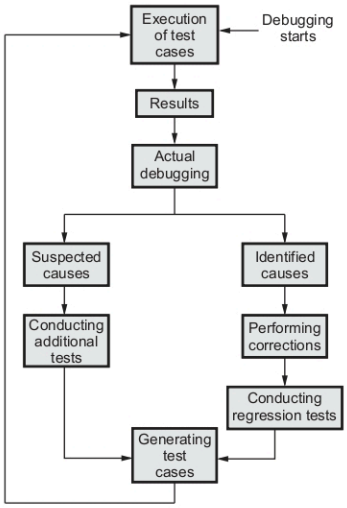



**Fig. Q.42.1 Debugging process**

- Common approaches in debugging are :
  - **Brute force method** - The memory dumps and run-time traces are examined and program with write statements is loaded to obtain clues to error causes.
  - In this method "Let computer find the error" approach is used.
  - This is the least efficient method of debugging.
  - **Backtracking method** - This method is applicable to small programs.
  - In this method, the source code is examined by looking backwards from symptom to potential causes of errors.
  - **Cause elimination method** - This method uses binary partitioning to reduce the number of locations where errors can exist.

**Q.43 Why debugging is so difficult ?**
**[JNTU : Part B, Marks 5]**

**Ans. :** Following are some reasons that reveal why **debugging is so difficult** -

1. The **symptoms** of bug may be present at some part(module) of the program and the its effect might be seen in some other module of the program. Hence **tracing out the location** of symptom becomes difficult.
2. Symptoms may be caused by software developers during the development process. Such symptoms are difficult to trace out.
3. The symptom may appear due to **timing problems** instead of processing problem
4. If the developer **corrects** some error then the symptom may **disappear temporarily**.
5. The symptom can appear if some **in-accuracies** in the program are simply **rounded off**.
6. In **real time systems**, it is not possible to accurately reproduce the input conditions and this may lead to symptoms of bugs.
7. The symptom may appear periodically. Such things normally occur in embedded systems in which hardware and software is coupled.
8. In distributed systems number of tasks are running on several distinct processors which may lead to symptoms.

**Q.44 Differentiate between testing and debugging ?**
**[JNTU : Part A, Marks 3]**

**Ans. :**

| Sr. No. | Testing | Debugging |
|---|---|---|
| 1. | Testing is a process in which the bug is identified. | Debugging is the process in which the bug or error is corrected by the programmer. |
| 2. | In testing process, it is identified where the bug occurs. | In debugging the root cause of error is identified. |
| 3. | Testing starts with the execution results from the test cases. | Debugging starts after the testing process. |

***END...***

UNIT - IV

# 10 Product Metrics

## 10.1 : Software Quality

**Q.1 Define software quality.**
**[JNTU : Part A, May-09, Marks 3]**

**Ans. :** Software quality can be defined as "the conformance to explicitly stated functional and performance requirements, explicitly documented development standards and implicit characteristics that are expected of all professionally developed software".

**Q.2 What are the points that software quality emphasis on ?** **[JNTU : Part A, May-09, Marks 3]**

**OR Why software quality gets failed ?**

**Ans. :** 1. Software **requirements** must be well understood before the software development process begins.

2. Similar to **explicit requirements** it is also essential to understand the implicit requirements of the software. If the software confirms the explicit requirements but not satisfying the implicit requirements then surely quality of software being developed is poor.
3. The set of **development criteria** has to be decided in order to specify the standards of the product. This will ultimately help the software engineer during development. If such a criteria is not been fixed then definitely the software product will lack the quality.

**Q.3 Write short notes on : a) Direct metrics b) Indirect metrics.** **[JNTU : Part B, May-09, Marks 8]**

**Ans. :** Generally, there are two classes of quality factors that affect the software quality

- Directly measured
- Indirectly measured

Direct metric includes the measurement of cost and effort applied. It includes lines of code produced, execution speed, memory size and the defects reported over some set of time.

Indirect metric includes the measurement of functionality, quality, complexity, efficiency, reliability, maintainability, usability, correctness and other abilities.

**Q.4 Write short note on : a) Correctness b) Maintainability.**

**Ans. : (a) Correctness :** It is an ability to fulfil the specification and customer requirements.

**(b) Maintainability :** It is an ability required to locate or fix the bugs in the software.

**Q.5 What is FURPS ? Explain.**
**[JNTU : Part B, April-11, Marks 8]**

**Ans. : 1. Functionality :** This is an ability by which the software satisfies the needs of the software denoted by suitability, accuracy, interoperability, compliance and security.

2. **Reliability :** The degree by which the software will be available. The sub categories are maturity, fault tolerance, recoverability.
3. **Usability :** The ability that indicates the usefulness of the software. The sub-categories are understandability, operability, learnability.
4. **Efficiency :** The measure of computing resources and time required by the program to perform. The sub categories are time behaviour and resource behaviour.
5. **Maintainability :** The ability required to locate or fix the bugs in software. The sub-attributes are analyzability, changeability, stability, testability.
6. **Portability :** The ability of the software to work properly even if the environment gets changed

(i.e. change in hardware or software). The sub attributes are adaptability, installability, conformance, replaceability.

## 10.2 : Framework for Product Metrics

**Q.6 Describe the framework for software product metrics.**
**[JNTU : Part B, Dec.-13, Marks 8, Dec.-16, Marks 5]**

**Ans. :** Software measurement means deriving a numeric value for an attribute of a software product or process.

**Measure :** It is a quantitative indication of the extent, amount, dimension, or size of some attribute of a product or process.

**Metrics :** It is the degree to which a system, component, or process possesses a given attribute. The software metrics relate several measures. For example - average number of errors found per review.

**Indicators :** Indicators mean combination of metrics that provides insight into the software process, project or product.

**Direct Metrics :** It refers to immediately measurable attributes. For example - line of code, execution speed.

**Indirect Metrics :** It refers to the aspects that are not immediately quantifiable or measurable. For example-functionality of the program.

**Faults :** Faults are of two types and those are errors and defects.

**i) Errors :** These are type of faults that are found by the practitioners during software development.

**ii) Defects :** Defects mean faults found by the customers after release.

**Q.7 Explain the principles for measurement.**
**[JNTU : Part B, Marks 5]**

**Ans. : Roche** suggested following principles for the measurement process

1. **Formulation :** The software measures and metrics should be appropriately represented for the software process being measured.
2. **Collection :** There should be appropriate mechanism to collect the results or data obtained from the formulated metrics.
3. **Analysis :** The analysis should be made on the computation of metrics and application of mathematical tool.
4. **Interpretation :** The evaluation metrics provides the insight for the software quality.
5. **Feedback :** The interpretations obtained from the product metrics must be submitted to the software team for review and feedback.

**Q.8 What is goal oriented software measurement ?**
**[JNTU : Part A, Marks 3]**

**Ans. :** Goal/Question Oriented Software Measurement (GQM) is a technique for identifying meaningful metrics for any part of software process.

For applying this technique following are the requirements -

1. The **explicit measurement goals** must be established which is based on process activity or process characteristics.
2. Prepare a **set of questionnaire** which will help to find out the measurement goals.
3. Identify **well formulated metrics** that will help to answer the prepared set of questions.

**Q.9 What are the attributes of effective software metrics ?** **[JNTU : Part B, Marks 5]**

**Ans. :** The effective software metrics should have following attributes.

1. **Simple and computable -** The derivation of metric should be easy to compute and should not be a time consuming activity.
2. **Empirically and intuitively persuasive -** It should be immediate and can be derived based on observations.
3. **Consistent and objective -** The metric should produce unambiguous results. Anybody should get the same result by using these metrics when same set of information is used.

4. **Consistent in its use of units and dimensions -** The mathematical units and dimensions used for the metric should be consistent. And there should not be intermixing of units.
5. **Programming language independent -** The metric should be based on analysis model, design model and program structure. It should be independent of programing languages, syntax or semantic of any programing language.
6. **Metric should be effective mechanism for high quality feedback -** The metric should provide a way to produce high quality software product.

## 10.3 : Metrics for Analysis Model

**Q.10 Discuss the metrics for specification quality.**
**[JNTU : Part B, April-11, Marks 6, Dec.-17, Marks 5]**

**Ans. :** • To assess the quality of analysis model and corresponding requirements **Davis and his colleagues** has suggested some **characteristics**. These characteristics are

- Completeness
- Correctness
- Understandability
- Verifiability
- Internal and external consistency
- Achievability
- Concision
- Traceability
- Modifiability
- Precision
- Reusability
- The total requirements in the specification can be specified as $n_r$

where $n_r = n_f + n_{nf}$

$n_r$ = Total number of requirements

$n_f$ = Total number of functional requirements

$n_{nf}$ = Total number of non functional requirements

- Davis has suggested the metric for specificity of requirements as

$$Q1 = n_{ui}/n_r$$

Where $n_{ui}$ is the number of requirements that have unique interpretation and $n_r$ is the total number of requirements.

- Completeness of functional requirements is given by

Where $Q2 = n_u/[n_i * n_s]$

$n_u$ = Number of unique function requirements

$n_i$ = Number of input given in the specification

$n_s$ = Number of states specified

Thus Q2 gives the percentage of necessary functions but here it does not consider the non functional requirements

- The overall metric(even by considering the non functional requirements) for completeness can be obtained by validating the requirements

$$Q3 = n_c/[n_c + n_{nv}]$$

Where

$n_c$ is number of requirements that are validated as correct.

$n_{nv}$ is number of requirements that are not been validated.

## 10.4 : Metrics for Design Model

**Q.11 Why do we need metrics for design model ? Describe in detail the architectural design metrics.**
**[JNTU : Part B, May-09, Dec.-10, Marks 16, April-11, Marks 10]**

**Ans. :** The design model metrics is required for determining the measurement of design quality. This metric guides the software design activity as design evolves.

**Architectural design metrics :** Two scientists Card and Glass has suggested three design complexity measures as

- *Structural complexity*
  Structural complexity depends upon the **fan-out** for modules. It can be defined as :

$$S(k) = f^2_{out}(k)$$

Where $f_{out}$ represents fan-out for module k[fan out means number of modules that are subordinating module k]

- *Data complexity*

  Data complexity is the complexity within the interface of internal module. For some module k it can be defined as

$$D(k) = \frac{tot\_var(k)}{[f_{out}(k)+1]}$$

Where *tot_var* is total number of input and output variables going to and coming out of the module.

- *System complexity*

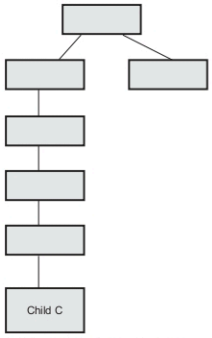
  System complexity is the combination of structural and data complexity. It can be denoted as

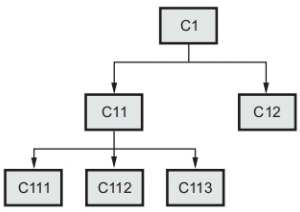
$$Sy(k) = S(k)+D(k)$$

When structural, data and system complexity get increased the overall architectural complexity also gets increased.

**Q.12 What is CK metrics suit ? State and explain the six class based design metrics for object oriented systems.** **[JNTU : Part B, Dec.-10, Marks 10]**

**Ans. :** The metric for object oriented design is mainly based on the fundamental unit called 'classes'. Chidamber and Kemerer has proposed OO software metrics which are popularly known as CK metrics suite. These are six class based design metrics for OO systems. Let us discuss them.

| Metric | Abbreviation | Description |
|---|---|---|
| **W**eighted **M**ethods per **C**lass | WMC | If in a class there are n methods and these methods have complexities C1,C2…Cn. Then overall complexity of the class is defined as a collection of all the complexities of methods in a class. i.e. WMC – Sum(Ci)<br>If number of methods in a class gets increased then the overall complexity of the class gets increased. It is necessary to have WMC value **minimum** in order to make the design **less complicated**. |
| **D**epth of **I**nheritance **T**ree | DIT | Assume that there exists an inheritance tree in OO design, then Depth of inheritance tree (DIT) is the maximum length from node to root of the tree.<br>For example :<br>  <br>In above given class hierarchy the longest path from root to node C indicates the depth of inheritance tree. In this tree predicting the behaviour of the child class C is a difficult task(because it has got inherited through many levels). Such a situation makes the complexity of design very very large. But at the same time we can many methods get reused in such a situation. |

| | | |
|---|---|---|
| **N**umber **O**f **C**hildren | NOC | The NOC is a count for number of children appearing as subclass in the class hierarchy.<br>**For example :**<br>  <br><br>In above given class hierarchy, C11 has three children C111, C112 and C113. As number of children increases the NOC value gets **incremented**. The **advantage** of this is that method reusability will be increased but the **disadvantage** is that it will require more amount of testing (i.e. *each child need to be tested*). |
| **C**oupling **B**etween **O**bject classes | CBO | CBO is the **number of collaborations** between the classes. There are two disadvantages of having large CBO value : i) The reusability of class will decrease and ii) the *modifications* and *testing after modification* activities become complicated. Hence the **CBO value** should be as **minimum** as possible.<br>This also implies that number of couplings in the software should be very less. |
| **R**esponse **F**or a **C**lass | RFC | RFC is number of methods that are executing for responding to a class. Normally when a message is received by an object of corresponding class then set of methods may get executed. RFC gives the count form these executing methods.<br>As **RFC increases** the testing as well as OO design becomes more and **more complicated**. |
| **L**ack of **CO**hesion in **M**ethods | LCOM | The LCOM is the count for set of methods accessing common attributes of the same class. For example : If there are 10 methods for a particular class C and seven methods have access to one or more attributes in common then LCOM is 7. Increase in LCOM means more and more classes should be connected with each other via a set of attributes. In other words cohesion in software must be high and LCOM should be **low**. |

**Q.13 Discuss the MOOD metrics suit.** ☞[JNTU : Part B, Dec.-10, Marks 6]

**Ans. :** **MOOD** metrics suite is proposed by Harrison, Counsell and Nithi for object oriented design. It includes two metrics MIF and CF. Let us discuss them

| Metric | Abbreviation | Description |
|---|---|---|
| **M**ethod **I**nheritance **F**actor | MIF | The MIF can be computed as<br>$MIF = \frac{\sum M_i(C_i)}{\sum M_a(C_i)}$<br>Where i varies from 1 to n and n denotes the total number of classes in the architecture.<br>$M_a(C_i) = M_a(C_i) + M_i(C_i)$<br>where,<br>$\mathbf{C_i}$ is a class within the architecture<br>$\mathbf{M_i(C_i)}$ is number of methods inherited in $C_i$<br>$\mathbf{M_d(C_i)}$ is number of methods declared in class $C_i$<br>MIF represents the impact of inheritance on Object Oriented system. |

| | | |
|---|---|---|
| **C**oupling **F**actor | CF | In software systems coupling represents the connection between elements of object oriented design. The coupling factor can be denoted as follows –<br>CF = $\Sigma_i \Sigma_j IsClient(C_i, C_j)/(T_C^2 - T_C)$<br>where,<br>i and j varies from 1 to total number of classes in the architecture $T_C$.<br>IsClient is a Boolean function, it is = 1 if there exists a relationship between client and server classes.<br>IsClient = 0 if there is no relationship between client class and server class.<br>As CF **increases** the complexity of object oriented design gets **increased.** |

## 10.5 : Metrics for Source Code

**Q.14 What is the metrics for source code ? Explain.** **[JNTU : Part B, Marks 5]**

**Ans. :** Halstead [1977] has proposed in "software science" some software science metrics. These metrics are based on -

- Common sense
- Information theory
- Psychology

In the proposed metrics the used measures are.

$n_1$ = The number of distinct operators in the program

$n_2$ = The number of distinct operands in the program

The paired block such as {…}, or begin…end or repeat…until are treated as single operator. The program length N can be defined as.

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

The program volume can be defined as

$$V = N \log_2 (n_1 + n_2)$$

The program volume is heavily dependent upon the program volume.

Let,

$N_1$ = Total count for all the operators in the program

$N_2$ = Total count for all the operands in the program

The program volume ration L can be defined as -

$$L = 2/n_1 X\ n_2/N_2$$

## 10.6 : Metrics for Testing

**Q.15 Define black box testing. Explain the metrics for testing.**

**[JNTU : Part B, May-13, Marks 5]**

**Ans. :** Black box testing is the software testing method which is used to test the software without knowing the internal structure of code or program.

Halstead's metrics for estimating the testing efforts are as given below.

The Halstead effort can be defined as

$$e = V/PL$$

Where V is the program volume and PL is the program level. The program level can be computed as

$$PL = 1/[(n_1/2)\ X\ (N_2/n_2)]$$

The percentage of overall testing effort =

testing effort of specific module/testing efforts of all the modules.

## 10.7 : Metrics for Maintenance

**Q.16 What is the metrics for source code and maintenance ? Explain.**

**[JNTU : Part B, May-13, Dec.-16, Marks 5]**

**Ans. : Metrics for source code -** Refer Q.14.

**Metrics for maintenance -** The stability of software product is given by an IEEE standard which suggests a metrics software maturity index(SMI) for that matter. It is given as follows -

$$SMI = (M - (A + C + D))/M$$

Where,

M = Number of modules in current version

A = Number of added modules in current version

C = Number of changed modules in current version

D = Number of deleted modules in current version compared to the previous version

When SMI reaches to the value 1.0 the product becomes more and more stabilized. This SMI metrics is used for planning the software maintenance activities.

## 10.8 : Software Measurement

**Q.17 Explain 3 p's in software metrics.**

☞[JNTU : Part B, Dec.-10, Marks 8]

**Ans. :** Effective software project management focuses on three P's - People, Product and Process.

**(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 Software Engineering Institute (SEI) has developed the People Management Capability Maturity Model (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 -
  1. Recruitment
  2. Selection
  3. Performance management
  4. Training compensation
  5. Career development
  6. Organization and work design
  7. Culture development.

**(2) The product**

- Before planning the project three important tasks need to be done -
- Product objectives and scope must be established.
- Alternative solutions should be considered.
- 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.

**(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.

**Q.18 Describe the metrics in the process and project domains.**

☞[JNTU : Part B, May-13, Marks 5]

**Ans. :** Process metrics are the set of process indicators that are used to improve the software processes. Process metrics is collected over the complete software life cycle. The software process can be improved with the help of process metrics.

**Project metrics** enables a software project manager to

(1) Assess the status of ongoing project.

(2) Track potential risks.

(3) Uncover problem areas before they go critical.

(4) Adjust workflow or tasks.

(5) Evaluate project team ability to control quality of software work products.

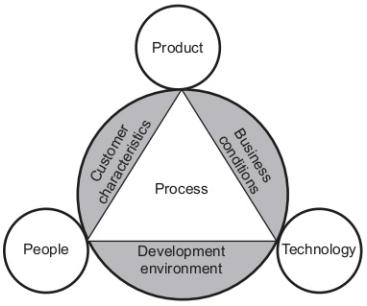



**Fig. Q.18.1**

- In making improvements to any software system, there are three basic quality factors to consider: **product, people, and technology**. These three are the major determinants of software cost, schedule, productivity, and quality.
- The factor **people** includes hiring the best people you can find, motivating them to do the best job, and training them on the skills needed to perform their jobs effectively.
- The **technology** factor includes acquiring and installing tools that help automate. The technology also includes use of new software languages to develop the desired quality product (e.g., Java, Visual C++, Oracle).
- The complexity of the factor **product** has great impact on quality and team performance.
- Out of these three components if one component is altered then it will impact other two factors. For example if an organization makes use of new testing tool(***technology***) then the staff(***people***) must be trained to work with this tool. The organization must consider if this new tool helps in generating the desired **software product**.
- This process triangle resides within the circle. This **circle** specifies the **environmental conditions** such as
  - Customer characteristics(communication and collaboration between user and developer).
  - Business conditions(Organizational policies, Business rules).
  - Development environment(use of new technologies, use of automated tools).

**Q.19 Explain size oriented metrics with an example.** **[JNTU : Part B, Feb.-10, Marks 8]**

**Ans. :** • 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 Q.19.1 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.

**Q.20 What are the advantages and disadvantages of size oriented metrics ?** [JNTU : Part A, Marks 3]

**Ans. : 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.

**Q.21 Explain function oriented metrics with example.**

[JNTU : Part B, Marks 5, Part A, Dec.- 19, Marks 2]

**Ans. :** • The function point 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.
- 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.

Example - Refer Q.22.

**Q.22 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.**

**Ans. : 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 |

Function point = Count total $\times [0.65 + 0.01 \times \Sigma(F_i)]$

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

$= 233 \times [0.65 + 0.32]$

$= 233 \times 0.97$

FP = 226.01

Hence adjusted function point is **226.01**

**Q.23 What are the advantages and disadvantages of function point metrics ?**

**Ans. : 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.

**Q.24 Give difference between size oriented metrics and function oriented metrics.**

**Ans. :**

| 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. |

**Q.25 Discuss the relationship between the lines of code and function points.**

☞[JNTU : Part B, Dec.-10, Marks 8]

**Ans. :** • The relationship between the Lines of Code and Function Points depend upon the programming language which is used for implementing the software and quality of design.

• Following table shows the rough estimates for some programming languages.

| Programming Language | LOC/FP |
|---|---|
| Assembly Language | 320 |
| C | 128 |
| C++ | 66 |
| COBOL | 105 |
| FORTRAN | 105 |
| PASCAL | 90 |
| ADA | 70 |
| php | 67 |
| Java | 31 |
| Forth Generation Languages | 30 |
| spreadsheets | 6 |
| Graphical languages | 4 |

• Using this information content it is possible to know from existing software, the number of function points once the total number of programming language statements are known.

• The FP and LOC are considered to be relatively accurate predicators of software development effort and cost.

**Q.26 What is software measure ? Explain the importance of it in detail.**

☞[JNTU : Part B, Dec.-11, Marks 7]

**Ans. :** • Software measure is quantitative indication of the extent, amount, dimension or size of some attribute of a product or a process.

• Thus software measurement is for obtaining a value for an attribute of software product or process.

• The software measurement is important because it allows -

1. Production planning, from a qualitative and quantitative viewpoint,
2. Production monitoring and control, from a qualitative and quantitative viewpoint,
3. Decision making,
4. Cost/benefit analysis, especially when new techniques are proposed or introduced,
5. Post-mortem analysis of projects.

**Q.27 What is meant by Defect Removal Efficiency (DRE) ? How it can be accessed ?**

☞[June 14, Marks 7]

**Ans. :** • While developing the software project many work products such as SRS, design document, source code are being created. Along with these work products many errors may get generated. Project manager has to identify all these errors to bring quality software.

• Error tracking is a process of assessing the status of the software project.

• The software team performs the formal technical reviews to test the software developed. In this review various errors are identified and corrected. Any errors that remain uncovered and are found in later tasks are called defects.

• The defect removal efficiency can be defined as

$$DRE = E/(E+D)$$

Where DRE is the defect removal efficiency,
E is the error
and D is defect.

• The DRE represents the effectiveness of quality assurance activities. The DRE also helps the project manager to assess the progress of software project as it gets developed through its scheduled work task.

• During error tracking activity following metrics are computed

1. Errors per requirements specification page : denoted by $E_{req}$
2. Errors per component - design level : denoted by $E_{design}$
3. Errors per component - code level : denoted by $E_{code}$
4. DRE - requirement analysis

5. DRE - architectural design
6. DRE - component level design
7. DRE - coding

## 10.9 : Metrics for Software Quality

**Q.28 What are the metrics for software quality ? Explain.** ☞[JNTU : Part B, May-13, Dec.-17, Marks 5 Dec.- 19, Marks 2 ]

**Ans. :** Following are the measure of the software quality -

1. **Correctness :** Correctness is a degree to which the software produces the desired functionality. The correctness can be measured as

   Correctness = Defects per KLOC

Where defect means lack of conformance to requirements. Such defects are generally reported by the user of the program.

Thus it is expected that the program must work correctly otherwise it is of no use.

2. **Integrity :** Integrity is basically an ability of the system to withstand against the attacks. Typically attacks are on programs, data and documents. There are two attributes that are associated with integrity: threat and security.

   **Threat** is the probability that specific type of attacks may occur. And **security** is the probability that the system will repel against the specific attack. Hence integrity can be measured as :

   $$\text{Integrity} = \Sigma((1-\text{Threat})\times(1-\text{Security})$$

System integrity is now a days has got great importance.

3. **Usability :** Usability means *user friendliness* of the system or ability of the system that indicates the usefulness of the system. Following are the characteristics that are useful for measuring the usability. The user friendliness is measured using following four characteristics -
   - The time required to make the system efficient.
   - The skill required to learn the system
   - The net increase in productivity after regular use of the system.
   - The user attitude towards the system.
4. **Maintainability :** Maintainability is an ability of the system to accommodate the corrections made after encountering errors, adapting the environment changes and adapt the changes made in the system in order to satisfy the user. The metric used for maintainability is MTTC i.e. mean time to change. The MTTC can be defined as the time required to analyse the change request, design an appropriate modification and implementation of those desired changes. Lower the value of MTTC means the software is more maintainable.

**Q.29 Differentiate measures, metrics and indicators. Explain software quality metrics.**

☞[JNTU : Dec.-19, Marks 5]

**Ans. : Measure :** Measure can be defined as quantitative indication of amount, dimension, capacity, or size of product and process attributes.

**Metric :** Metrics can be defined as quantitative measures that allow software engineers to identify the efficiency and improve the quality of software process, project, and product.

**Indicator :** An indicator is a qualitative or quantitative factor that provides a simple means to express achievement and the attainment of a goal.

**Software quality metrics :** Refer Q.28.

# For Mid Term Exam



## Fill in the Blanks for Mid Term Exam

1. Testing of individual components by the developers are comes under ____ type.
2. The goal of integration testing is to find defects in communication between two modules rather than seeing if modules are working correctly.( True or false)
3. Stubs and drivers are used in big bang integration. (True/False)

**Explanation :** In big bang integration, all the modules are required to be completed before integration testing starts. So, it does not require any dummy programs.

4. Alpha and Beta testing are forms of ___ testing.
5. Cyclomatic complexity is computed in ___.
6. During ____ testing the source code, statements, loops,branches are checked.
7. ___ testing is also known as functional testing.
8. _____ is a software measure, that provides a quantitative measure of logical complexity of a program.
9. The two types of integration testing are ___ and ___.
10. _____ is mainly used for testing real-time and embedded systems.
11. The important metrics used during testing is ___.
12. _____ refers to a different set of tasks ensures that the software that has been built is traceable to customer requirements.
13. _____ refers to the set of tasks that ensures the software correctly implements a specific function.
14. The metrics ___ is based on functionality of the application.
15. ____ measure is derived by considering the size of software in terms of lines of code.
16. DRE stands for ___.
17. ____ testing technique are applied to detect the errors from each software component individually.

## Multiple Choice Questions for Mid Term Exam

**Q.1** Testing can be applied to ____.

- [a] requirements
- [b] design
- [c] coding

**Q.2** What are various testing strategies ?

- [a] unit testing
- [b] integration testing
- [c] validation testing
- [d] system testing
- [e] all of the above

**Q.3** Unit testing is done by _____.

- [a] developers
- [b] designer
- [c] user

**Q.4** What is the objective of integration testing ?

- [a] To verify that system meets user expectation and needs
- [b] To verify that system separately testable modules are functioning properly

c To verify that interfaces between different parts of system

d All of the above

**Q.5** Choose the correct option
I. Stubs are dummy calling programs
II. Drivers are dummy called programs

a Only I

b Only II

c Both I and II

d Neither I nor II

**Explanation :** Stubs are dummy called programs and drivers are dummy calling programs

**Q.6** Which traditional order in Software Testing is organized ?
A) Integration Testing B) System Testing
C) Unit Testing D) Validation Testing

a A, D, C, B

b B, D, A, C

c C, A, D, B

d D, B, C, A

**Q.7** Regression testing is performed _____.

a every week

b after software has changed.

c as many times as possible

d when the software gets completely ready

**Q.8** When should you stop testing ?

a When time for testing has run out.

b When the test completion criteria have been met

c When all planned tests have been run

d When no faults have been found by the tests run

**Q.9** Which of the following is a form of functional testing ?

a Usability testing

b Boundary value analysis

c Performance testing

d Security testing

**Q.10** Which of the following testing is performed by user ?

a Acceptance testing

b Unit testing

c Compatibility testing

d Integration testing

**Q.11** Which of the following are advantages of using LOC (lines of code) as a size oriented metric ?

a LOC is easily computed

b LOC is a language dependent measure

c LOC is a language independent measure

d LOC can be computed before a design is completed

**Q.12** Beta testing is done by _____.

a developer b user

**Q.13** Before handing over the software to the client, which testing is to be done in-house ?

a Alpha b Beta

c Gamma d Theta

**Q.14** Which of the following is the black box testing ?

a Basis path testing

b Structural testing

c Boundary value analysis

d Both (a) and (b)

**Q.15** White box testing can be started _____.

a after SRS b after designing

c after coding d any time

**Q.16** White-Box Testing is also known as _____ .

a structural testing

b code-based testing

c clear box testing

d all of the above

**Q.17** The amount of time that the software is available for use is known as _____.

a reliability b usability

c efficiency d functionality

**Answer Keys for Fill in the Blanks :**

| | | | |
|---|---|---|---|
| **1.** | unit testing | **2.** | True |
| **3.** | False | **4.** | acceptance testing |
| **5.** | white box testing | **6.** | white box |
| **7.** | Black box | **8.** | Cyclomatic complexity |
| **9.** | incremental, non incremental | **10.** | Performance testing |
| **11.** | reliability | **12.** | Validation |
| **13.** | Verification | **14.** | function point model |
| **15.** | Size oriented | **16.** | Defect Removal Efficiency |
| **17.** | Unit testing | | |

**Answer Keys for Multiple Choice Questions**

| | | | |
|---|---|---|---|
| **1.** | c | **2.** | e |
| **3.** | a | **4.** | c |
| **5.** | d | **6.** | c |
| **7.** | b | **8.** | b |
| **9.** | b | **10.** | a |
| **11.** | a | **12.** | b |
| **13.** | a | **14.** | c |
| **15.** | c | **16.** | d |
| **17.** | a | | |

*END...*

UNIT - V

# 11 Risk Management

## 11.1 : Reactive Vs. Proactive Risk Strategies

**Q.1 What is risk and risk management ?**

☞[JNTU : Part A, Dec.-17, Marks 5]

**OR Explain various steps in risk management.**

☞[JNTU : Dec.-19, Marks 5]

**Ans. : • 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.

**Q.2 Differentiate between reactive Vs. proactive risk strategies.**

☞[JNTU : Part B, Dec.-13,16,19, Marks 5]

**Ans. :** The difference is as follows -

| Sr. No. | Reactive Risk Management | Proactive Risk Management |
|---|---|---|
| 1 | Reactive risk management is a risk management strategy in which when project gets into trouble then only **corrective action** is taken. | Proactive risk management strategy **begins before the technical activity** by considering the probable risk. |
| 2 | When reactive risks can not be managed and new risks come up one after the other, the software team flies into action in an attempt to correct problems rapidly. These activities are called **"firefighting" activities.** | In this strategy potential **risks are identified first then their probability and impact is analyzed.** Such risks are then specified according to their priorities (i.e. high priority risks should be managed first!). Finally the software team prepares a plan for managing these risks. |
| 3 | In this strategy **no preventive care** is taken about the risks. They are handled only on their occurrences. | In this strategy the **occurrence of risks is prevented.** But it is not possible to avoid all the risks, hence team prepares the risk management plan in such a manner that risk controlling can done efficiently. |
| 4 | It is an **older approach** of risk management. | The proactive risk management techniques are **used now a days** by most of the IT industries. |
| 5 | Reactive risk management solely depends on **past accidental analysis and response.** | Proactive risk management **combines a mixed method of past, present and future prediction before finding solutions to avoid risks.** |

## 11.2 : Software Risks

**Q.3 What types of risk occur during software development ?** ☞[JNTU : Part B, Dec.-16, Marks 5, Feb.-10, Dec.-14,Marks 8]

**Ans. : 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.

## 11.3 : Risk Identification

**Q.4 What is risk identification ?** [JNTU : Part A, Marks 2]

**Ans. :** 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.

**Q.5 Explain the two approaches used during the risk identification.** [JNTU : Part A, Marks 2]

**Ans. :** 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.

**Q.6 Discuss about risk identification and risk item checklist.** [JNTU : Part B, Feb.-10, Marks 8]

**OR Discuss in detail about risk identification.** [JNTU : Part B, May-13, Dec.-17, Marks 5, May-09, Marks 8]

**Ans. :** 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.**

**Q.7 What are the components of risks ?** [JNTU : Part B, Dec.-10, Marks 8]

**Ans. :** Following are commonly used components of risk -

1. ***Performance risk*** **-** It is the degree of uncertainty that the product will satisfy the requirements
2. ***Cost risk*** **-** It is the degree of uncertainty that the project will maintain the budget.
3. ***Support risk*** **-** It is the degree of uncertainty that the software project being developed will be easy to correct, modify or adapt.
4. ***Schedule risk*** **-** It is the degree of uncertainty that the software project will maintain the schedule and the project will be delivered in time.

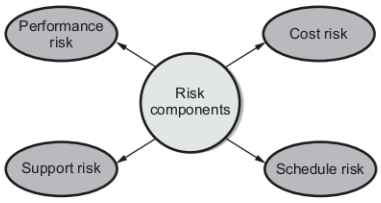



**Fig. Q.7.1 Components of risk**

## 11.4 : Risk Projection

**Q.8 Explain about risk projection.** [JNTU : Part B, May-09, Dec.-13, Marks 8]

**OR What are the four steps for risk projection ? What is the intention of these steps ?** [JNTU : Part B, Dec.-10, 17, Marks 5]

**Ans. :** 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.

**Q.9 Develop a risk table and explain it.** [JNTU : Part B, Dec.-10, Marks 8, April-11, Marks 10]

**Ans. :** The sample risk table is as given below -

| Risk | Category | Probability | Impact | RMMM |
|---|---|---|---|---|
| Is the skilled staff available | Staff | 50 % | Catastrophic | |
| Is that the team size sufficient | Staff | 62 % | Critical | |
| Have the staff received sufficient training | Staff | 25 % | Marginal | |
| Will technology meet the expectations | Technology | 30 % | Critical | |
| Is the software management tool available | Environment | 40 % | Negligible | |
| How much amount of reused software is required? | Project size | 60 % | Marginal | |
| Will customer change the requirement ? | Customer | 20 % | Critical | |

While building the risk table

- The project team first of all enlists all probable risks with the help of risk item checklist.
- Each risk is then categorized. As we know various categories of risk can be a) Project size b) Technology c) Customer d) Staff e) Business f) Developing environment.
- Probability of occurrence of each risk is then estimated by each team member individually.
- Then impact of each risk is assessed. While calculating the impact of each risk, each using the cost drivers each component of risk (*performance, cost, support, and schedule*) is assessed and it then averaged to quote the overall impact of particular risk.

1. After building this table it is then sorted by probability and impact. The high probability and high impact risks will be at the top of the table. And low probability and low impact risk will be at the bottom of the table. This arrangement of the table is called **first-order prioritization**.
2. Then the project manager goes through this first-order prioritized risk table and draws a horizontal line at some point in the table. This line is called **cut off line**. The risks table above the cut off line is now considered for further risk analysis.
3. The risk table below the cut off line is again sorted and a **second-order prioritization** is applied on this table.
4. The risk table above the cut-off line is having the risks with high probability and high impact and such risks should occupy the significant amount of management time.
5. All the risks that lie above the cut off line should be managed. Using Risk mitigation, monitoring and management plan the last column of the risk table is filled up.

**Q.10 Explain how to access risk impact ?**
**[JNTU : Part B, Dec.-10, Marks 6]**

**Ans. :** While assessing the risks impact three factors are considered

- Nature of risk • Scope of the risk • Timing at which risk occurs.

**Nature** of risk denotes the type or kind of risk. For example if software requirement is poorly understood, the software processes gets poorly designed and ultimately it will create a problem in unit testing. **Scope** of the risk means severity of the risk. And **timing** of risk means determining at which phase of software development life cycle the risk will occur and how long it will persist.

U.S. Air Force has suggested following steps in order to determine the impact of risk -

1. The probability of all the components of risk (*performance, cost, support and schedule*) is calculated and averaged.
2. Using risk drivers (*catastrophic, critical, marginal, negligible)* the impact of risk on each components is determined.
3. Build the risk table and analyse the high impact, high probability risks.

## 11.5 : Risk Refinement

**Q.11 Explain the concept risk refinement.**
**[JNTU : Part A,Dec.-17, Marks 2]**

**Ans. :** • Risk refinement is a process of specifying the risk in more detail. The risk refinement can be represented using CTC format.

- The CTC stands for *condition-transition-consequence.*
- The condition is first stated and then based on this condition sub conditions can be derived. Then determine the effects of these sub conditions in order to refine the risk.
- This refinement helps in exposing the underlying risks.
- This approach makes it easier for the project manager to analyse the risk in greater detail.

## 11.6 : RMMM

**Q.12 What does RMMM stand for ?**

**Ans. :** RMMM stands for - Risk mitigation, monitoring and management.

**Q.13 What are the objectives of risk monitoring ?**
☞[JNTU : Part A, Marks 3]

**Ans. :** The objectives of risk monitoring are -

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.

**Q.14 Describe about RMMM.**
☞[JNTU : Part B, Dec.-10, Marks 8]

**OR What is RMMM. Explain various methods followed to mitigate, monitor and manage risks.**
☞[JNTU : Dec.-19, Marks 5]

**Ans. :** RMMM stands for **risk mitigation**, **monitoring** and **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.

**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.

**Q.15 Discuss about risk management.**
☞[JNTU : Part B, Dec.-10, Marks 8]

**Ans. :** Refer Q.14.

**Q.16 What are the risks associated with delayed projects ? How do project managers manage such risk ?**

**Ans. :** With delayed project the project risk occurs. Following are the problems that may get occur due to delayed projects -

1. The cost of overall project will get increased.
2. The resources need to be engaged for a long time

3. The technology may get changed over a period. And project may no longer be compatible with the current trends and technology.
4. The requirements might get changed by the period of completion of the period.
5. Due to delayed project work the development staff may loose interest in the project and hence there will be dilution of effort.

The project managers assist the project team in developing the strategy for dealing with delayed project risk. An effective strategy considers three important issues

1. Risk avoidance
2. Risk monitoring
3. Risk monitoring and contingency planning.

The first and foremost activity is to avoid the risk of getting the project a delayed one. This is called risk mitigation. To mitigate this risk project management must develop a strategy for handling the delay in projects. The possible steps can be

1. Meet the current staff and determine the causes of delay in the project.
2. Communicate with the customer to convince him/her about elimination/reduction of unrealistic requirements.
3. Mitigate those causes that are under the control of project manager.
4. Organize the project team in such a way that development activity is widely dispersed.
5. Develop a technique to ensure the continuity when people leave the ongoing project in-between.
6. Conduct peer reviews of all work periodically.
7. Assign backup staff member for critical activities.
8. Define documentation standards and develop the documents in timely manner. The documents will be useful for all the team members during the development process.

As project proceeds risk monitoring activities start and then it could be identified whether or not the risk is becoming more or less likely. During this phase following factors can be monitored -

1. The general attitude of team members in high project pressures.
2. Interpersonal relationship among the team members
3. The degree to which the team has jelled.

If the risk mitigation efforts are failed and if the risk of delayed project becomes the reality then if backup is available, information is documented and knowledge is dispersed across the team then project manager can add-up the new-comers to speed up the work. Some members then can transfer the knowledge to the new-comers and amount of work can be shared with them.

By following above discussed strategies, the delay in the project can be handled to some extent.

## 11.7 : RMMM Plan

**Q.17 Write short note on RMMM Plan.**
**[JNTU : Part B, Dec.-12, 14, Marks 5]**

**Ans. :** • The RMMM plan is a document in which all the risk analysis activities are described.

• Typical template for RMMM plan or Risk information sheet can be,

| **Risk information sheet** | | | |
|---|---|---|---|
| **Project name** *<enter name of the project for which risks can be identified>* | | | |
| **Risk id** *<#>* | **Date** *<date at which risk is identified >* | **Probability** *<risk probability>* | **Impact** *<low/medium/ high>* |
| **Origin** *<the person who has identified the risk>* | | **Assigned to** *<who is responsible for mitigating the risk>* | |
| **Description** *<Description of risk identified>* | | | |
| **Refinement/Context** *<associated information for risk refinement>* | | | |

| | |
|---|---|
| **Mitigation/Monitoring**<br>*<enter the mitigation/monitoring steps taken>* | |
| **Trigger/Contingency plan**<br>*<if risk mitigation fails then the plan for handling the risk>* | |
| **Status**<br>*<Running status that provides a history of what is being done for the risk and changes in the risk. Include the date the status entry was made>* | |
| **Approval**<br><name and signature of person approving closure>. | **Closing date**<br>*<date>* |

- The risk information sheet can be maintained by database systems.
- After documenting the risks using either RMMM plan or Risk information sheet the risk mitigation, monitoring and analysis activities are stopped.

***END...***

UNIT - V

# 12 Quality Management

## 12.1 : Quality Concepts

**Q.1 What is software quality ?**

**[JNTU : Part A,Marks 2]**

**Ans. :** Software quality can be defined as "the conformance to explicitly stated functional and performance requirements, explicitly documented development standards, and implicit characteristics that are expected of all professionally developed software".

**Q.2 What are the points that software quality emphasis on ?** **[JNTU : Part B, Dec.-10, Marks 8]**

**Ans. :** *Quality of design* is the characteristics of the item which is specified for the designer. For example if a temperature control system is designed, then it should display the temperature with Maximum limit of 100 degree centigrade. This is what the basic characteristic of that system is. And at the time of design of the product this issue must be focused.

*Quality of conformance* is the degree to which the design specifications are followed during manufacturing. If the degree of conformance is **more** then it indicates **higher quality**.

**Q.3 Explain - Cost of quality.**

**[JNTU : Part B, May-13, Marks 5, Part A, Dec.-19, Marks 2]**

**Ans. :** The cost of quality can be defined as the total cost required to obtain the quality in the product and to conduct the quality related activities.

The cost of quality has various components such as

1. **Prevention cost -** This is the cost of quality required for conducting quality planning, formal technical reviews, test equipments and training.
2. **Appraisal cost -** This is the cost of quality required for gaining the insight into the product.

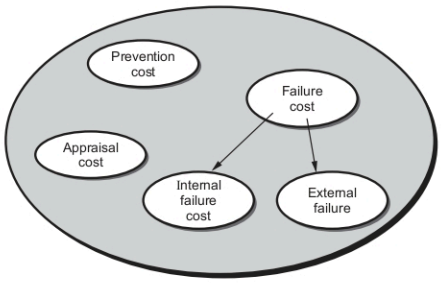



**Fig. Q.3.1 Cost of quality**

It includes the cost required for in-process and inter process inspection, maintenance and testing.

3. **Failure cost -** Failure cost means the cost required to remove the defects in the software product before delivering it to customer. There are two types of failure costs

   a. *Internal failure cost* - Internal failure is nothing but the cost of defects occurred in the product before delivering it to customer e.g. repair in networking, repairing of communication network.

   b. *External failure cost* - External failure is the cost of defects occurred in the product after delivering it to the customer e.g. product repair/replace, complaint processing, warranty work.

## 12.2 : Software Quality Assurance

**Q.4 Give the three reasons why software quality gets failed ?** **[JNTU : Part A, Marks 3]**

**Ans. :** There are three main reasons for ***why software quality gets failed ?***

1. Software **requirements** must be well understood before the software development process begins.
2. Similar to **explicit requirements** it is also essential to understand the implicit requirements of the software. If the software confirms the explicit requirements but not satisfying the implicit requirements then surely quality of software being developed is poor.
3. The set of **development criteria** has to be decided in order to specify the standards of the product. This will ultimately help the software engineer during development. If such a criteria is not been fixed then definitely the software product will lack the quality. Software quality assurance is the process in which conformance to the requirements of the product is made. This shows how important software quality assurance is.

**Q.5 Explain about software quality assurance.**

**Ans. : 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.

**Q.6 What are the activities of SQA ?**

☞[JNTU : Part B, May-09, Marks 10]

**OR Define software quality assurance. State various SQA activities.** ☞[JNTU : Dec.-19, Marks 5]

**Ans. :** The activities of SQA are enlisted as follows -

**1. Prepare an SQA plan for a project.**

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 the development of the project's software process description.**

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

**3. Reviews software engineering activities to verify compliance with the defined software process**

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

**4. Audits designated software work products to verify compliance with those defined as part of the software process.**

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. These work products are documented and handled according to documented procedure.**

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

**6. Records 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.

**Q.7 Discuss the importance of quality assurance.**
**[JNTU : Part B, Dec.-10, Marks 6, Dec.-13, Marks 7]**

**Ans. :** • Software Quality Assurance (SQA) is defined as a well planned and systematic approach to evaluate the quality of software.

- It checks the adherence to software product standards, processes, and procedures.
- SQA includes the systematic process of assuring that standards and procedures are established and are followed throughout the software development life cycle and test cycle as well.
- The compliance of the built with agreed-upon standards and procedures is evaluated through process monitoring, product evaluation, project management etc.
- The major reason of involving software quality assurance in the process of software product development is to make sure that the **final product built is as per the requirement specification** and comply with the standards.

## 12.3 : Software Reviews

**Q.8 What is the need of software review ?**
**[JNTU : Part A, May-09, Dec.-16, Marks 3, Dec.-17, Marks 5]**

**Ans. :** • Software reviews are filter to software engineering process. Such reviews are applied at various points during software development life cycle.

- The **objective** of software reviews is to uncover errors and defects that can be removed.
- The software reviews are conducted for following reasons -
  i) Point out needed improvements in the product of a single person or team.
  ii) Confirm those parts of the product in which improvement is not desired.

**Q.9 What are the different types of reviews ?**
**[JNTU : Part B, May-09, Marks 5]**

**Ans. :** There are three different ways by which software review can be conducted.

- **Informal meeting** an informal meeting can be conducted outside the working environment and an informal discussion of technical issues can be held.
- **Formal presentations** can be conducted for customer, management and technical staff.
- **Formal Technical Reviews (FTR)** sometimes called as **walkthrough** or an inspection is the most effective way of software review. This helps a lot for uncovering the software errors and to improve the quality of software.

**Q.10 Write short note on Software reviews.**
**[JNTU : Part B, Dec.-12, May-13, Marks 5]**

**Ans. :** Refer Q.8 and Q.9.

**Q.11 Explain defect removal amplification with an example.** **[JNTU : Part B, Feb.-10, Marks 8]**

**Ans. :** • A defect amplification model can be used to illustrate the generation and detection of errors during the steps in the software engineering process.

- This model is as given below -

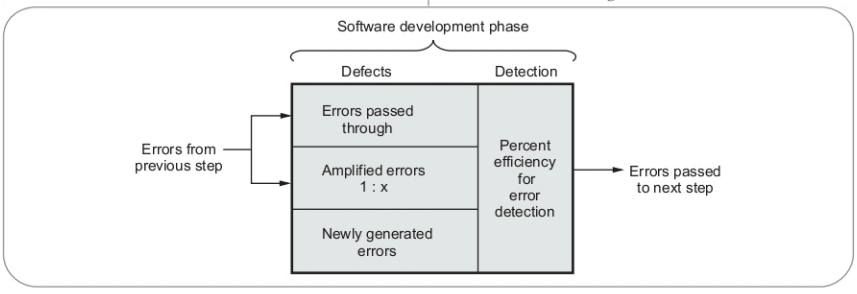



**Fig. Q.11.1 Defect amplification model**

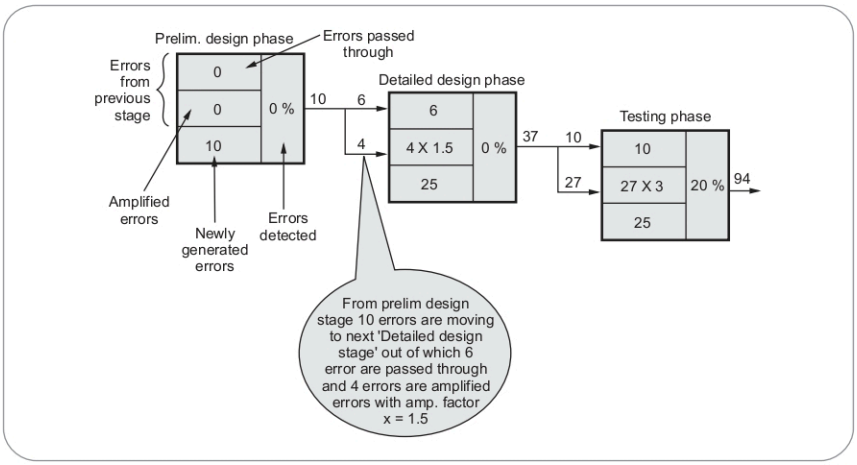



**Fig. Q.11.2**

- In the defect amplification model the outer box indicates the software development stage. This box is partitioned into two categories, defects and detection of errors.
- Under the defect column we list out number of errors. Errors can be those coming from previous stage or it could be newly generated errors.
- The previous errors can be those that are passed through errors as well as the amplified errors. The error amplification factor is denoted by x.

## 12.4 : Formal Technical Reviews

**Q.12 What is the significance of Formal Technical Review ?** ☞[JNTU : Part B, Dec.-16, 19, Marks 5]

**Ans. :** Formal Technical Review is a software quality assurance activity performed by software engineer.

**Objectives of FTR**

1. FTR is useful to uncover errors in logic, function and implementation for any representation of the software.
2. The purpose of FTR is to ensure that software meets specified requirements.
3. It also ensures that the software is represented according to predefined standards.
4. It helps to review the uniformity in software development process.
5. It makes the project more manageable.

**Q.13 Explain about (a) The review meeting. (b) Review reporting and record keeping (c) Review guideline (d) Sample driven reviews.**

☞[JNTU : Part B, Feb.-10, Marks 16, April-11, Marks 8, Dec.-19]

**Ans. : (i) The review Meeting :**

- Every review meeting should be conducted by considering the following constraints -
  - **Involvement of people -** Between 3 and 5 people should be involved in the review.
  - **Advance preparation -** Advance preparation should occur but it should be very short i.e. at the most 2 hours of work for each person can be spent in this preparation.
  - **Short duration -** The duration of the review meeting should be less than 2 hours.
- Rather than attempting to review the entire design, **walkthroughs** are conducted for modules or for small groups of modules.
- The **focus** of the FTR is on a **work product** (a software component to be reviewed).

The review meeting is attended by the review leader, all reviewers and the producer.

- The *review leader* is responsible for evaluating the product for its readiness. The copies of product material is then distributed to reviewers.
- The *producer* organizes a "walkthrough" the product, explaining the material, while the *reviewers* raise issues based on their advance preparation.
- One of the reviewers becomes *recorder* who records all the important issues raised during the review. When errors are discovered, the recorder notes each.
- At the end of the review, the attendees decide whether to accept the product or not, with or without modifications.

**(ii) Review reporting and record keeping :**

- During the FTR, the reviewer actively records all issues that have been raised.
- At the end of meeting these all raised issues are consolidated and *review issues list* is prepared.
- Finally, a *formal technical review summary report* is produced.
- Purpose of review list
  1. *It helps in identifying problematic areas within the product.*
  2. *From this issue list, a check list can be prepared which guides the producer for making the corrections.*
- This review issue list is normally attached to formal technical summary report.

**(iii) Review Guideline :** Guidelines for the conducting of formal technical reviews must be established in advance. This guideline must be distributed to all reviewers, agreed upon, and then followed. For example - Guideline for review may include following things.

1. Concentrate on work product only. That means review the product, not the producer.
2. Set an agenda of review and maintain it.
3. When certain issues are raised then debate or arguments should be limited. Reviews should not ultimately result in some hard-feelings.
4. Find out problem areas, but don't attempt to solve every problem noted.
5. Take written notes (it is for the recorder).
6. Limit the number of participants and insist upon advance preparation.
7. Develop a checklist for each product that is likely to be reviewed.
8. Allocate resources and time schedule for FTRs in order to maintain time schedule.
9. Conduct meaningful trainings for all reviewers in order to make the reviews effective.
10. Review earlier reviews which servers as the base for the current review being conducted.

**(iv) Sample driven reviews :**

- In real world situation, for conducting the software projects the resources are limited and time span for completion of project is very short. Hence many times the reviews are skipped. But this may affect the overall quality of software product.
- **Thelin and his colleagues** addressed this issue by suggesting the **sample driven review process.**
- In this process samples of all software engineering work products are inspected to determine which work product **are most error prone.**
- The FTR resources then focus on only those work products that are error-prone.

**Q.14 Which is the effective method of software review and explain.** **[JNTU : Part B, May-09, Marks 8]**

**Ans. :** Refer Q.12.

**Q.15 Explain - Review Guidelines.**
**[JNTU : Part A, May-13, Marks 3]**

**Ans. :** Refer Q.13 (iii)

## 12.5 : Statistical Software Quality Assurance

**Q.16 What are the steps to be followed by statistical quality assurance ?**
**[JNTU : Part A, May-09, Marks 4]**

**Ans. :** • 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

**Q.17 Explain with an example data collection for statistical software quality assurance.**

☞[JNTU : Part B, May-09, Marks 6]

**Ans. :** Consider that a well known software firm has collected some information about the defects that are occurring in the software product.

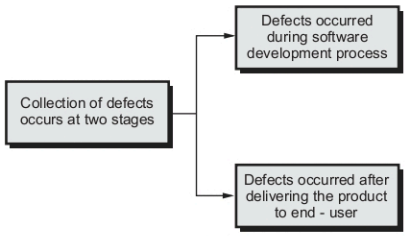



**Fig. Q.17.1**

The causes of defects are tabularized as below -

| Causes of defects | Abbreviations |
|---|---|
| Error in data representation | EDR |
| Error in design logic | EDL |
| Unclear human computer interface | HCI |
| Incomplete and/or erroneous specifications | IES |
| Intentional deviations from specifications | IDS |
| Inconsistent component interface | ICI |
| Incomplete and/or erroneous testing | IET |
| Incomplete or inaccurate documentation | IID |
| Error in programming language translation of design | PLT |
| Violation of programming standards | VPS |
| Misinterpretation of customer communication | MCC |
| Miscellaneous | MIS |

Suppose that some sample representative data of defect causes is collected then one can build a statistical SQA from it. The sample data about the defect causes is as given below -

| Causes of defect | Number of errors | Percentage |
|---|---|---|
| IES | 200 | 21 |
| MCC | 158 | 17 |
| IDS | 48 | 5 |
| VPS | 20 | 2 |
| EDR | 128 | 14 |
| ICI | 60 | 6 |
| EDL | 45 | 5 |
| IET | 94 | 10 |
| IID | 35 | 4 |
| PLT | 60 | 6 |
| HCI | 30 | 3 |
| MIS | 58 | 6 |
| **TOTAL** | **936** | **100%** |

**Table Q.17.1 Sample data collection for statistical SQA**

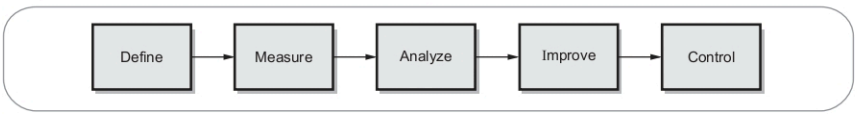



**Fig. Q.18.1 Six sigma framework**

From above table it is clear that the major defect cause is IES+MCC+EDR which generates 21+17+14 = 52 % of total errors. We can say that these errors are *few vital errors* that cause the major defect in the software product.

**Q.18 Explain six sigma strategy for statistical software quality assurance.**
**[JNTU : Part B, May-09, Marks 6]**

**Ans. :** Six sigma is widely used statistical software quality assurance strategy. It is a business driven approach to process improvement, reduced costs and increased profit.

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.

**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.

**Q.19 Explain clearly the statistical software quality assurance.** **[JNTU : Part B, Dec.-13, Marks 8]**

**Ans. :** Refer Q.16 and Q.17.

## 12.6 : Software Reliability

**Q.20 Define software reliability.**
**[JNTU : Part A, Dec.-16, Marks 2, Dec.-17, Marks 3]**

**Ans. :** • Software reliability is defined as the probability of failure free operation of a computer program in a specified environment for a specified time.

• The software reliability can be measured, directed and estimated.

**Q.21 Explain the measures of software reliability.**
**[JNTU : Part B, Dec.-10, 19, Marks 5]**

**OR Write short note on - Software reliability.**
**[JNTU : Part B, Dec.-14, Marks 5]**

**OR Explain the measures of software realibility and availability.** **[JNTU : Dec.-19, Marks 3]**

**Ans. :** Normally there are two measures of software reliability.

1. ***MTBF*** Mean-Time-Between-Failure is a simple measure of software reliability which can be calculated as

   MTBF = MTTF + MTTR

where MTTF means mean-time-to-failure

and MTTR stands for mean-time-to-repair.

Many software researchers feel that MTBF is more useful measure of software reliability than defects/KLOC or defects/FP.

2. ***Availability***

   It's another measure of software reliability software availability is defined as the probability that the program is working according to the requirements at a given points in time. It is measured as

   Availability = (MTTF/(MTTF+MTTR))* 100 %

MTBF is equally sensitive to MTTF and MTTR but *availability* is more sensitive to MTTR.

**Q.22 Explain the software safety in terms of software quality assurance.**
**[JNTU : Part B, Dec.-10, Marks 6]**

**Ans. :** • Software safety is a quality assurance activity in which **potential hazards** are identified and assessed.

- These hazards may bring the total failure of the system. If such hazards are identified and specified in early stage of software development then such hazards can be eliminated or controlled in order to make the software safe.
- Modeling and analysis process is conducted as a part of software safety.
- For example : In a computer based automobile system software hazards are
  1. Uncontrolled acceleration that can not be stopped.
  2. Does not respond to slow the system when breaks are applied.
  3. Slowly gains the speed.
- **How to handle the system level hazards ?**

Following are the steps that can be applied to preserve the software safety.

**Step 1** : The hazards are identified.

**Step 2** : Analysis techniques are used to assign severity of these hazards. The probability of occurrence of such hazards is also analyzed with the help of analysis techniques. The commonly used analysis techniques are fault-tree analysis, real-time logic and Petri-net models. These techniques basically predict the chain of events that can cause hazards.

**Step 3** : Once hazards are identified, safety related requirements can be specified for the software. This specification basically includes list of undesirable events and desired system response.

**Q.23 What is the difference between software reliability and software safety ?**

**Ans. :** • Software reliability and software safety are closely related to each other. However, the difference between them lies in degree and not the type.

- Software reliability uses statistical analysis made to determine the occurrence of software failure. These failures will cause simply dissatisfaction of customer requirements. But the software safety examines the ways in which failure results in conditions that can lead to hazards.
- Software reliability does not detect the failures in depth. But the software safety detects the failures in context of an entire computer based system.

## 12.7 : The ISO 9000 Quality Standards

**Q.24 Write short note on ISO 9000 quality standard.**

**[JNTU : Part B, Dec.-10, 12, 14,16, May-13, Marks 5]**

**Ans. :** • ISO 9000 is a family of *quality assurance system.* It can be applied to all types of organizations.

- 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 900, company's quality system and operations are scrutinized by third-party auditors for a compliance 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*

# For Mid Term Exam

## Unit - V

### Fill in the Blanks for Mid Term Exam

1. ____ activity refers to the process of making decisions based on an evaluation of the factors that threat to the business
2. Four activities that are carried out for risk management are Risk identification, Risk Projection, ____ and ____
3. _____risks threaten the quality and timeliness of the software to be built.
4. Firefighting activities are associated with _____.
5. The risks that affect the project budget, schedule, resources and staffing is called ___.
6. Market risk belongs to ___ category.
7. _____is an activity that consists of auditing, reporting functions of management.
8. FTR stands for ___.
9. FTR includes _____inspections, reviews.
10. The quality standard used in software engineering process is ___.
11. RMMM stands for ___.
12. Changes in Government policies is of ____ type of risk(predictable/unpredictable)
13. MTBF means___.
14. _____ are a "filter" to software engineering processes.

### Multiple Choice Questions for Mid Term Exam

**Q.1** What is Risk ?

- a Negative consequence that could occur
- b Negative consequence that will occur
- c Negative consequence that must occur
- d Negative consequence that shall occur

**Q.2** What is the second stage of risk management process ?

- a Risk planning
- b Risk monitoring
- c Risk analysis
- d Risk identification

**Q.3** Risk Management is a responsibility of the _____.

- a customer
- b investor
- c developer
- d project team

**Q.4** The goal of quality assurance is to provide management with the data needed to determine which software engineers are producing the most defects.

- a True
- b False

**Q.5** The problem that threatens the success of a project but which has not yet happened is a ____.

- a bug
- b error
- c risk
- d failure

**Q.6** If P is risk probability, L is loss, then Risk Exposure (RE) is computed as _____.

- a RE = P/L
- b RE = P + L
- c RE = P*L
- d RE = 2* P *L

**Q.7** Software risk always involves two characteristics. What are those characteristics ?

- a Uncertainty and Loss
- b Certainty and Profit
- c Staff size and Budget
- d Project Deadline and Budget

**Q.8** When senior management or responsible staff leaves the organization then following type of risk occurs _____.

a management risk b technical risk
c project risk d all of these

**Q.9** When risk is associated with hardware or software technology then following type of risk occurs _____.

a management risk
b technical risk
c project risk
d all of these

**Q.10** RE represents _____ .

a risk Expense
b related Expense
c risk Exposure
d risk Evaluation

**Q.11** Non-conformance to software requirements is known as _____.

a software availability
b software reliability
c software failure
d all of the above

**Q.12** What is Six Sigma ?

a It is the most widely used strategy for statistical quality assurance
b The "Six Sigma" refers to six standard deviations
c It is the most widely used strategy for statistical quality assurance AND The "Six Sigma" refers to six standard deviations
d None of these

**Q.13** Which of the following is not a core step of Six Sigma ?

a Define b Control
c Measure d Analyse

**Q.14** The ISO quality assurance standard that applies to software engineering is _____.

a ISO 9000 b ISO 9001
c ISO 9002 d ISO 9003

**Answer Keys for Fill in the Blanks :**

| | | | |
|---|---|---|---|
| **1.** | Risk Management | **2.** | risk refinement, risk mitigation, monitoring and management |
| **3.** | Technical risks | **4.** | reactive risk strategy |
| **5.** | project risk | **6.** | business risk |
| **7.** | Quality assurance | **8.** | Formal Technical Review |
| **9.** | walkthroughs | **10.** | ISO 9001 |
| **11.** | Risk Mitigation, Monitoring and Management | **12.** | unpredictable |
| **13.** | Mean Time Between Failure | **14.** | Software reviews |

**Answer Keys for Multiple Choice Questions**

| | | | |
|---|---|---|---|
| **1.** | a | **2.** | c |
| **3.** | d | **4.** | b |
| **5.** | c | **6.** | c |
| **7.** | a | **8.** | a |
| **9.** | b | **10.** | c |
| **11.** | c | **12.** | c |
| **13.** | b | **14.** | b |

***END...***

# DECEMBER - 2019

**Software Engineering 135BM (R-16)**

**Solved Paper**
**III[rd] Year [B.Tech]**
**Sem - I (CSE/IT)**

**Time : 3 Hours]** **[Maximum Marks : 75**

**Note :** This question paper contains two parts A and B.

Part A is compulsory which carries 25 marks. Answer all questions in Part A. Part B consists of 5 Units. Answer any one full question from each unit. Each question carries 10 marks and may have a, b, c as sub questions.

**PART - A** **(25 Marks)**

**Q.1** **a)** *What is legacy software ? Explain.* **(Refer Q.8 of Chapter 1)** [2]

**b)** *Define software and its characteristics.* **(Refer Q.4 of Chapter 1)** [3]

**c)** *Discuss software scope and feasibility study.* **(Refer Q.5 of Chapter 5)** [2]

**d)** *Classify various functional and non-functional requirements.* **(Refer Q.5 of Chapter 4)** [3]

**e)** *Explain in brief the taxonomy of various architectural styles.* **(Refer Q.8 of Chapter 8)** [2]

**f)** *Express the golden rules in performing user interface design.* **(Not in new syllabus)** [3]

**g)** *Explain function point metrics.* **(Refer Q.21 of Chapter 10)** [2]

**h)** *What are software quality metrics ?* **(Refer Q.3 of Chapter 12)** [3]

**i)** *What are software risks ? Explain various types of software risks.* **(Refer Q.2 of Chapter 11)** [2]

**j)** *Explain the measures of software realibility and availability.* **(Refer Q.21 of Chapter 12)** [3]

**PART - B** **(50 Marks)**

**Q.2** **a)** *Compare and contrast between waterfall model and spiral model with neat diagrams.*
**(Refer Q.26 of Chapter 3)**

**b)** *Analyze the importance of the unified process in software development.*
**(Refer Q.31 and Q.32 of Chapter 3)** [5 + 5]

**OR**

**Q.3** **a)** *Give an overview of capability maturity model integration. Which level of organizations as a customer you would prefer and why ?* **(Refer Q.5 of Chapter 2)**

**b)** *What are various software myths prevalent in industry ? Why do the stakeholders believe them ? Contradict the myths with reality.* **(Refer Q.11 of Chapter 1)** [5 + 5]

**Q.4** **a)** *Discuss various steps in requirements engineering. What are the work products of engineering the requirements ?* **(Refer Q.3 of Chapter 5)**

**b)** *What is context model ? Describe the importance of context model.* **(Refer Q.5 of Chapter 6)** [5 + 5]

**OR**

**Q.5** **a)** *Describe desirable characteristics of a good software requirement specification document. What is the role of SRS in software engineering ?* **(Refer Q.21 of Chapter 4)**

**b)** *Describe the problem of library management system using two UML diagrams.* **(Refer Q.31 and Q.47 of Chapter 8)** **[5 + 5]**

**Q.6** **a)** *Explain various types of cohesion and coupling. List in the order of level of cohesion and coupling preferred for component level design.* **(Refer Q.16 of Chapter 7)**

**b)** *Explain various steps in the user interface design and evaluation.* **(Not in new syllabus)** **[5 + 5]**

**OR**

**Q.7** **a)** *Illustrate with neat diagrams the process of mapping data - flow into a software architecture.* **(Refer Q.22 of Chapter 8)**

**b)** *Give an overview of steps in conducting component level design.* **(Refer Q.48 of Chapter 8)** **[5 + 5]**

**Q.8** **a)** *What is software testing ? Explain test characteristics.* **(Refer Q.7 of Chapter 9)**

**b)** *Give an overview of white-box testing techniques with help of flow graph.* **(Refer Q.34 of Chapter 9)** **[5 + 5]**

**OR**

**Q.9** **a)** *Explain black box testing. Give an account of equivalence partitioning and boundary value analysis techniques.* **(Refer Q.27, Q.29 and Q.30 of Chapter 9)**

**b)** *Differentiate measures, metrics and indicators. Explain software quality metrics.* **(Refer Q.29 of Chapter 10)** **[5 + 5]**

**Q.10** **a)** *Explain various steps in risk management.* **(Refer Q.1 of Chapter 11)**

**b)** *Define software quality assurance. State various SQA activities.* **(Refer Q.6 of Chapter 12)** **[5 + 5]**

**OR**

**Q.11** **a)** *What is RMMM. Explain various methods followed to mitigate, monitor and manage risks.* **(Refer Q.14 of Chapter 11)**

**b)** *What is role of format technical reviews in quality control ? Discuss various steps and procedure to conduct a FTR.* **(Refer Q.12 and Q.13 of Chapter 12)** **[5 + 5]**

***END...***

# SOLVED MODEL QUESTION PAPER

**Software Engineering [As Per R-18 Pattern]**

**III[rd] Year B. Tech., Sem - I [CSE/IT]**

**Time : 3 Hours]** **[Maximum Marks : 75**

**Note :** This question paper contains two parts A and B.

Part A is compulsory which carries 25 marks. Answer all questions in Part A. Part B consists of 5 Units. Answer any one full question from each unit. Each question carries 10 marks and may have a, b as sub questions.

**PART - A** **(25 Marks)**

**Q.1** **a)** *What is legacy software ? Explain.* **(Refer Q.8 of Chapter-1)** [2]

**b)** *List the task regions in the spiral model.* **(Refer Q.16 of Chapter - 3)** [2]

**c)** *Discuss software scope and feasibility study.* **(Refer Q.5 of Chapter-5)** [2]

**d)** *What are the differences between functional requirements and non-functional requirements ?* **(Refer Q.7 of Chapter - 4)** [2]

**e)** *Explain in brief the taxonomy of various architectural styles.* **(Refer Q.8 of Chapter - 8)** [2]

**f)** *Enlist three objectives of testing process ?* **(Refer Q.2 of Chapter - 9)** [3]

**g)** *Explain function point metrics.* **(Refer Q.21 of Chapter - 10)** [3]

**h)** *What are software quality metrics ?* **(Refer Q.3 of Chapter - 12)** [3]

**i)** *What are software risks ? Explain various types of software risks.* **(Refer Q.2 of Chapter - 11)** [3]

**j)** *Explain the measures of software realibility and availability.* **(Refer Q.21 of Chapter - 12)** [3]

**PART - B** **(50 Marks)**

**Q.2** **a)** *Explain the following : Water fall model* **(Refer Q.4 of Chapter - 3)**

**b)** *Write detailed notes on CMMI.* **(Refer Q.5 of Chapter - 2)** [5 + 5]

**OR**

**Q.3** **a)** *What is software process ? What is need of software process improvement ?* **(Refer Q.2 and Q.3 of Chapter - 2)**

**b)** *What are various software myths prevalent in industry ? Why do the stakeholders believe them ? Contradict the myths with reality.* **(Refer Q.11 of Chapter - 1)** [5 + 5]

**Q.4** **a)** *Discuss various steps in requirements engineering. What are the work products of engineering the requirements ?* **(Refer Q.3 of Chapter - 5)**

**b)** *What is context model ? Describe the importance of context model.* **(Refer Q.5 of Chapter - 6)** [5 + 5]

**OR**

**Q.5** **a)** *Describe desirable characteristics of a good software requirement specification document. What is the role of SRS in software engineering ?* **(Refer Q.21 of Chapter - 4)**

**b)** *Describe the problem of library management system using two UML diagrams.* **(Refer Q.31 and Q.47 of Chapter - 8)** **[5 + 5]**

**Q.6** **a)** *Explain various types of cohesion and coupling. List in the order of level of cohesion and coupling preferred for component level design.* **(Refer Q.16 of Chapter - 7)**

**b)** *What is system modeling ? Explain the process of creating models and the factors that should be considered when building models.* **(Refer Q.1 and Q.2 of Chapter - 6)** **[5 + 5]**

**OR**

**Q.7** **a)** *Illustrate with neat diagrams the process of mapping data - flow into a software architecture.* **(Refer Q.22 of Chapter - 8)**

**b)** *Give an overview of steps in conducting component level design.* **(Refer Q.48 of Chapter - 8)** **[5 + 5]**

**Q.8** **a)** *What is software testing ? Explain test characteristics.* **(Refer Q.7 of Chapter - 9)**

**b)** *Give an overview of white-box testing techniques with help of flow graph.* **(Refer Q.34 of Chapter - 9)** **[5 + 5]**

**OR**

**Q.9** **a)** *Explain black box testing. Give an account of equivalence partitioning and boundary value analysis techniques.* **(Refer Q.27, Q.29 and Q.30 of Chapter - 9)**

**b)** *Differentiate measures, metrics and indicators. Explain software quality metrics.* **(Refer Q.29 of Chapter - 10)** **[5 + 5]**

**Q.10** **a)** *Explain various steps in risk management.* **(Refer Q.1 of Chapter - 11)**

**b)** *Define software quality assurance. State various SQA activities.* **(Refer Q.6 of Chapter - 12)** **[5 + 5]**

**OR**

**Q.11** **a)** *What is RMMM. Explain various methods followed to mitigate, monitor and manage risks.* **(Refer Q.14 of Chapter - 11)**

**b)** *What are different types of reviews* **(Refer Q.9 of Chapter - 12)** **[5 + 5]**

***END...***